# What Do You Need for Diverse Trajectory Composition in Diffusion Planning?

**Quentin Clark**
Department of Computer Science
University of Toronto
qtcc@cs.toronto.edu

**Florian Shkurti**
Department of Computer Science
University of Toronto
florian@cs.toronto.edu

## Abstract

In planning, stitching is an ability of algorithms to piece together sub-trajectories of data they are trained on to generate new and diverse behaviours. While stitching has historically been a strength of offline reinforcement learning, recent generative behavioural cloning (BC) methods have also shown proficiency at stitching. However, the main factors behind this are poorly understood, hindering the development of new algorithms that can reliably stitch. Focusing on diffusion planners trained via imitation, we find two properties are key contributors for composition: *shift equivariance* and *local receptiveness*. We use these two properties to explain architecture, data, and inference choices in existing generative BC methods based on diffusion planning, including replanning frequency, data augmentation, and data scaling. Experimental comparisons show that (1) while locality is more important than shift equivariance in creating a diffusion planner capable of composition, both are crucial (2) enabling these properties through relatively simple architecture choices can be competitive with more computationally expensive methods such as replanning or scaling data, and (3) simple inpainting-based guidance can guide architecturally compositional models to enable generalization in goal-conditioned settings.

## 1 Introduction

The recent resurgence in generative diffusion models [58, 30] has inspired a new wave of work investigating the potential for these models to act as planners for robotic agents [34, 13]. One surprising ability these diffusion planning models have is *compositionality*: the ability to take component elements seen in samples of their dataset and compose them in new, kinematically feasible ways [20, 44].

One form of compositionality especially important in diffusion planning is *stitching*, where some diffusion planning methods are capable of composing sub-trajectories into new trajectories distinct from the training data. Historically, stitching has been seen as one of the primary benefits of temporal difference (TD) learning [38], both in offline and online RL. Planning with diffusion models thus potentially offers some of the stitching benefits of TD learning without needing to handle RL-specific challenges.

Despite their recent success in vision, when and why diffusion models compose in the planning domain is not well understood. Contrary to toy experiments in prior works [1, 34], the convolutional neural network (CNN)-based backbones commonly used in diffusion planners often struggle with composition [9], instead regressing to memorization of training examples [2].

This gap between the promise of diffusion planners to enable stitching and reality that they frequently struggle with composition is crucial to address, as diffusion offers an appealing alternative to other

Preprint. Under review.

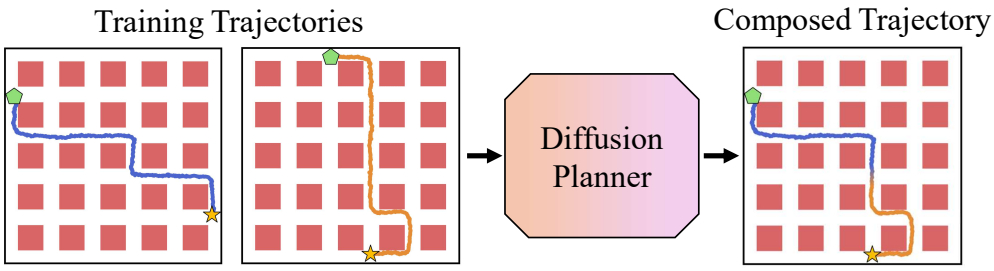



Figure 1: An overview of our problem. We try to understand when and why diffusion planners are able to compose sub-trajectories seen disparately in its training data into new trajectories, without resorting to dynamic programming or TD learning.

stitching approaches. RL suffers from many challenges when stitching such as the deadly triad [33, 22, 64] and slow temporal credit assignment [12], while non-generative BC stitching approaches require nettlesome data augmentation [26]. The capability to sample a diverse set of new trajectories at test time is important property for planners broadly. Reasons for this include being able to select solutions from a candidate set [69], encouraging multi-modal behaviour [40], or generating diverse data to train imitation learning or offline RL algorithms [38].

To address this gap, we analyze a variety of approaches that induce composition in diffusion planners. We find that there are two key ingredients: *local receptiveness* [52] during generation, and *shift equivariance* of states in a trajectory [36]. Through experiments in navigation environments, we show that although most current diffusion planning models do not possess this compositional capacity natively, it can be induced through architecture changes, inference choices, data augmentation strategies, or data scaling without needing dynamic programming or TD-learning. We further show that the diversity of trajectories this compositionality enables can be selectively sampled from using inpainting, allowing the use of diffusion planners only trained on imitation as goal-conditioned planners, similar to offline goal-conditioned RL.

Our primary contributions are as follows:

- Showing that local attention and shift equivariance, previously shown to mechanistically explain creativity and generalization in image diffusion models [36], also function as ingredients for composition in diffusion planners. However, unlike the image diffusion case, they are not the only factors: the inference method used to sample trajectories from diffusion planners also plays a significant role.
- Suggesting that the presence or absence of composition behaviours in past work on diffusion planning [9, 48, 9, 62, 13] can be partially explained by our ingredients
- Demonstrating that when induced through architecture changes, the stitching capabilities of whole-sequence diffusion can be effectively guided through simple inpainting to achieve strong goal-conditioned performance on unseen start-goal pairs.

Our work provides more clarity behind what components are necessary to guide diffusion planners to compose trajectories at test-time successfully. We hope this will inspire future algorithmic and architectural developments in compositional diffusion planners.

## 2 Related Work

**Diffusion Models:** Diffusion models [61, 30] offer an efficient way train models that can sample from complex distributions. Diffusion models train from data points drawn from some target distribution $P$ by taking a sample $x \sim P$ and iteratively adding Gaussian noise, such that $x_{t+1} = x_t + \eta_t, \eta_t \sim \mathcal{N}(0, \sigma^2)$, called the *forward diffusion process*. The diffusion model learns the *reverse diffusion process*, where some model (usually a neural network) $f_\theta$ parameterized by $\theta$ is trained to learn $f(x_t, t) = x_{t-1}$ by minimizing the MSE loss between the model's output and the reverse diffusion step: $\theta^\star = \text{argmin}_\theta \sum_{t=1}^{T} \mathbb{E}||f_\theta(x_t) - x_{t-1}||_2^2$. Sampling from the new distribution is performed by first sampling from the Gaussian and using the model to reverse-sample the diffusion process: $x_{t-1} = f(x_t, t) + \mathcal{N}(0, \sigma^2 \Delta t)$ [50, 7].

**Diffusion Planning:** Introduced by Janner et al. [34], diffusion planners model trajectories with diffusion. A diffusion model predicts an upcoming state or state-action trajectory horizon $H_F$ time-steps long, conditioned on a starting state or state memory of length $H_P$, in an environment with $T$ max steps. From the generated plan, actions are executed for a fixed horizon $H_A$ before generating a new trajectory. In this work, we are mostly interested in the *whole sequence* planning domain, where $H_A = H_F = T$, and no replanning occurs. Diffusion planning should not be confused with the *diffusion policy* [13] formulation where the diffusion model generates a short sequence of actions with high reactivity from frequent re-generation of actions. We elaborate on this distinction further in Appendix A.6.3.

**Stitching in Imitation Learning:** While stitching is often viewed as a strength of TD learning [38], other work analyzes stitching in imitation learning [8, 28]. We were inspired by Ghugare et al. [26], which analyzes stitching through a compositional generalization perspective. They revealed that applying supervised-learning as a form of offline RL [21] fails to stitch, requiring training data augmentation which manually stitches together trajectories. However, their result does not directly translate to ours because the properties we analyze (local receptiveness and shift equivariance) both enforce a form of regularization that creates composition, even when such behaviour is not explicitly built into the data.

**Compositionality in Diffusion Planners:** Many approaches exist to create compositionality in diffusion planners. One common thread is performing hierarchical planning, including using diffusion to stitch existing dataset trajectories [41], generating subsequences jointly conditioned on each-other [48], or using a hierarchical cascade of models to separately create high-level plans and low-level trajectories [10, 42]. Our work differs from these by mainly targeting *whole-sequence* compositionality, where composition does not occur from explicit stitching but from implicit model behaviour, similar to image diffusion. [1]

Several other works have observed that Diffusion models often produce stitched trajectories when frequently replanning during inference [9] and limiting the history the next generation is conditioned on [62]. Our work attempts to formalize these qualitative observations with more robust experiments and explain it under a single conceptual framework (locality for short history and shift equivariance for the replanning process) that also extends to architectural or training choices.

Our focus on shift equivariance and locality was inspired by prior works which observed that these properties explain much of the compositional behaviour of image diffusion models [36, 52]. We extend their work in a few significant ways. One, we connect their image-diffusion observations to sequential decision-making by connecting these properties that enable composition to other methods commonly used to achieve stitching in IL. Two, we show that common diffusion architectures fail to exhibit these properties, preventing stitching from occurring, especially at small data sizes. Three, we analyze the effect each independently has on compositionality. Finally, we show that common guidance strategies enable strong *base-model* compositionality to be *directed* towards certain goals, which is particularly important when using generative models for offline goal-conditioned RL.

# 3 Methodology

## 3.1 Local Receptiveness and shift equivariance in Score-Based Diffusion Models

To unify prior explanations for compositionality in diffusion planners, we rely on two properties previously described for diffusion models in vision [36, 52].

The first is *locality*, which means the size of the region around an individual state in a trajectory that influences the next denoising step for that state is constrained to a small region. A local model only attends to a small patch around part of a sample when denoising, but through multiple denoising steps, information can propagate across a generated sample.

**Definition 1: Locally Receptive Field.** Assume we are interested in generating trjectories $\tau$, and have a generative trajectory model whose output estimates the denoising gradient $M_t[\tau]$. We say that $M_t[\tau]$ has a field locally receptive to a state neighbourhood $\Omega$ if for all trajectories $\tau$ and state positions within those trajectories $x_s$, $M_t[\tau](x_s) = M_t[\tau|\Omega_{x_s}](x_s)$.

[1] See Appendix A.6.2 for more discussion.

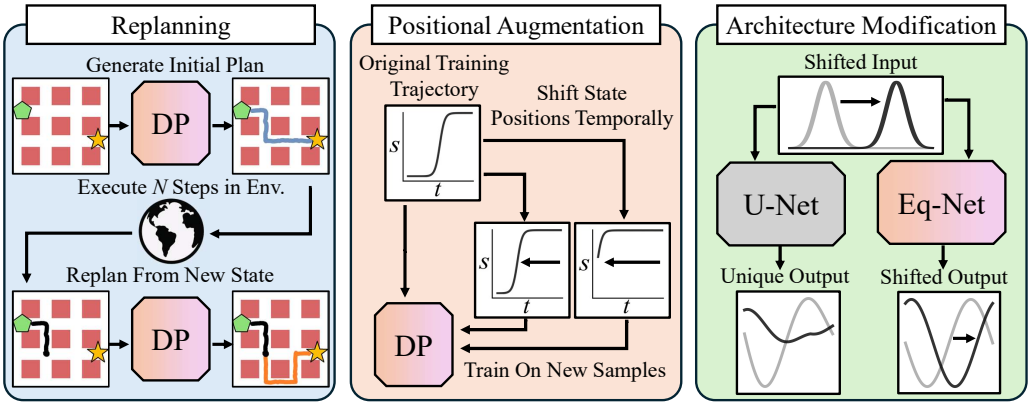



Figure 2: The methods to enable trajectory composition that we examine. The left shows replanning (an inference technique), the middle positional augmentation (a dataset augmentation technique) and the right an architectural modification (an architectural modification). Each of these enables composition, showing that the broad principles of locality and shift equivariance can be applied in different contexts.

The second is *positional equivariance*, which requires that the model not use the absolute position of a state in the sequence as a condition during the denoising process. Convolutional neural networks (CNNs) are typically assumed to have this property, as convolutions by design shift a shared weight kernel and apply it to local patches in a sequence. However, downsampling and pooling operators common in CNNs break this property [70, 6].

**Definition 2: Shift equivariance.** A trajectory score model $M_t[\tau]$ is shift equivariant if for all shifts [2] $U$ on trajectories $\tau$ which uniformly change the position of states within the trajectory, $M_t[U[\tau]] = U[M_t[\tau]]$. [3]

### 3.2 Composition in Diffusion Planners

Our main insight is that these notions of locality and shift equivariance have analogies to other strategies used to increase composition in diffusion planners.

**Increasing Compositionality by Replanning:** The first strategy we address is *replanning*, specifically replanning with a limited history. A trajectory planning model replans when it generates a trajectory $\tau$ of length $H_F$, executes some action horizon $H_A < H_F$, and then generates a new trajectory starting at the current state $s_1$. Because a diffusion planner can be queried to generate plans with a variable length, $H_F$ is often set to some number large enough to ensure temporal consistency far enough into the future to do effective planning. Additionally, a history of past states to condition on with length $H_P$ must be decided.

Formally, given a generative trajectory model $f$ that generates trajectories of length $H_F$ conditioned on a history of length $H_P$, each state in the trajectory will only be directly attentive to a local field of size $l = \max(H_F, H_P)$, i.e, $P(s_t|s_{t-l}, s_{t-l+1}, ...s_{t+l-1}, s_{t+l})$.

It is important to note that locality through replanning and locality in the reverse-diffusion process are not identical, as the reverse diffusion process being several steps long means influence can be spread along a long trajectory through many local denoising steps. We show this visually in Figure 3. However, the similar mechanisms highlight how increasing the influence of nearby states in a given trajectory is important for composition.

Replanning also ensures a form of shift equivariance when replanning with a fixed amount of the past trajectory kept as conditioning for the future trajectory. Diffusion planners can condition their plans on past state observations in two ways: either by including it as an explicit condition to the diffusion

[2] By shift equivariant we mean to discrete shifts and not continuous translations. See McGreivy and Hakim [49].

[3] In theory, circular padding is required for a CNN to have full shift equivariance[36], but in practice this is uncommon.

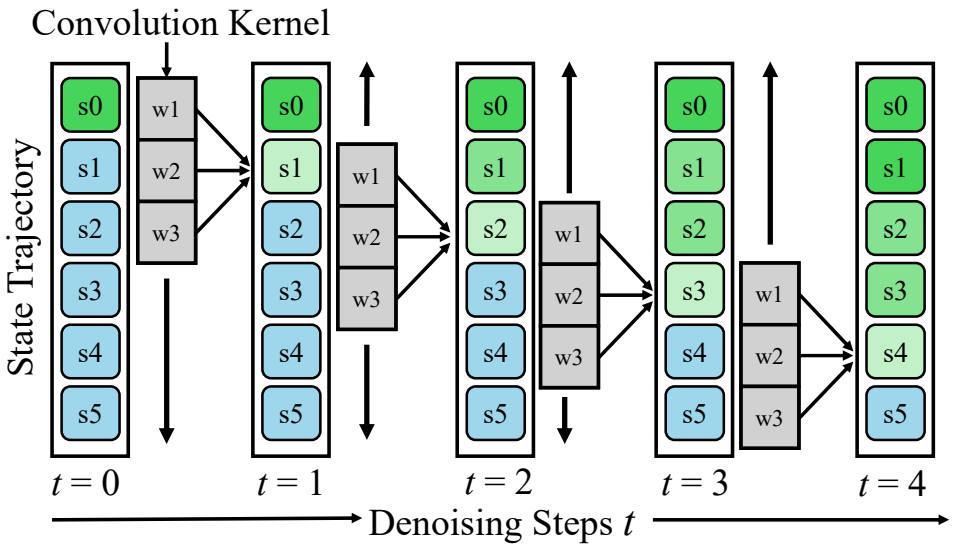



Figure 3: A visualization of how information during diffusion denoising can be propagated by locally receptive models. Here, we imagine a toy setting with a sequence of length 6, being denoised by a model consisting of just a single convolutional kernel with a receptive field of size 3 and stride 1. State s0 has some information that must be considered when denoising the rest of the sequence. Even though the later states in the sequence cannot receive information from s0 during a single denoising step, over several diffusion steps information propagates across the state trajectory. Note that this does not just happen in one direction down the trajectory, but can also happen up it as each convolution happens in parallel.

network, usually combined with classifier-free guidance (CFG) [31], or by inpainting $H_P$ states in the new generated trajectory. Full memory with inpainting does *not* guarantee shift equivariance, as each state in the trajectory can end up only being drawn to the same position in the trajectory that they were seen in during training. However, replanning with a smaller history shifts the viewed positions of a particular segment of the trajectory over by $H_P$ steps. Paradoxically, this means models with strong replanning capability must already have some form of shift equivariance.

**Increasing Compositionality by Training Augmentation:** The next strategy to increase compositionality we address is *training augmentation*, where the data is modified during training to give the model approximate learned equivariance to some transformation [3, 59]. The most common approach to encourage stitching is manually stitching together trajectories in the training set randomly by finding similar states in different trajectories[26, 8, 28, 39]. This augmentation attempts to breaks the corelation between states far away in a trajectory that prevents stitching. However, this strategy requires tuning the distance metric that determines when segments are close, and is potentially infeasible if the combinatorial set of all legal stitches is large enough to make augmenting every possible trajectory expensive.

We instead focus on augmenting the data to provide shift equivariance. Specifically, we examine *positional augmentation*, where the position of specific states within a trajectory are randomized during training such that the model cannot overfit to a state's position. We implement this by sampling a random state from a training trajectory, removing the states before that state, and then padding the end of the trajectory with the final state until it is the same length as the original trajectory, such that the trajectory is randomly "shifted" by a number of states.[4]

Formally, given a training trajectory $\tau$ with states $\tau[1] = s_1, \tau[2] = s_2, ...\tau[T] = s_T$, we sample a position from the uniform distribution $i \sim U_{[1,T]}$. To create our new augmented trajectory $\tau^\star$, we shift the states in $\tau$ by $i$ positions to the left, such that for state indexes $x \in [1, T-i)$, $\tau^\star[x] = \tau[x+i]$.

[4]This assumes that the environment can be kept at the same state by the agent, which is true of our environments. Alternatives include masking out the remainder of the trajectory or allowing for mixed-length trajectories during training.

For the remainder of the states $x \in [T-i, T]$, we set pad the states with the last state in the original trajectory: $\tau^\star[x] = \tau[T]$.This modified trajectory is then used when training our planner.

**Increasing Compositionality via Architecture Design Choices:** The next strategy we address are *architecture modifications*. While equivariance can be learned through data augmentation, another popular strategy is to simply build equivariance to certain transformations through architecture changes[14, 15], especially in limited data settings.

Surprisingly, we found that the common 1D U-Net proposed in Janner et al. [34] has a globally receptive field and is not equivariant to shifts, which we show in Appendix A.4.1. To easily analyze the effect that locality and shift equivariance have on composition, we replace the U-Net with a network which we call *Eq-Net*. Eq-Net simply consists of stacked 1D convolutions without pooling or down-sampling layers, which guarantees equivariance to shifts. We also decrease the size of the kernels in Eq-Net to guarantee locality. We defer further discussion and explanation to Appendix A.4.2.

To test the effects of having a network with local receptiveness but not shift equivariance, we add positional encoding to Eq-Net with sinusoidal embeddings [66], adding each time positions' embedding channel-wise after the first convolutional layer. See Appendix A.4.5 for details.

**Increasing Compositionality via Scaling Training Data:** The last strategy we address is *scaling training data*, which has been frequently observed to increase diffusion model generalization [68] and compositionality [53]. Importantly, this increase in compositionality does not come from memorizing more of the legal space of generations, but from learning the underlying rules governing composition in a particular area. Specifically, Favero et al. [23] finds that as data scales, progressively higher-level grammar rules are learned, starting with low level (local) consistency rules.

Our work is specific to sequential decision making, where data is often sparse. For instance, the toy vision dataset CIFAR-10 [37] has $60,000$ samples, and the more modern ImageNet [17] has $1,281,167$. By contrast, most of the environments in the modern offline RL OGBench dataset [54] have between $1000$ and $5000$ episodes per environment. Similarly, most of the tasks in Chi et al. [13] use $500$ or less trajectories. Given that most sequential decision making datasets have orders of magnitude fewer samples, considering other approaches to generalization becomes more important than relying on data scaling.

**Explaining Composition in Other Diffusion Planners:** Our goal in examining the factors that lead to composition is to help explain why prior works compose (or fail to) so that future diffusion planning practitioners know which choices are important if their domain requires compositionality. Due to space constraints we include a thorough discussion of other works in Appendix A.6.3. Our main finding is that the majority of existing works achieve locality and shift equivariance either through frequent replanning with short-history conditioning [62, 9, 13] or some form of hierarchical decomposition where trajectories are generated explicitly as sub-trajectories and then composed manually [10, 48, 42]. Surprisingly, achieving compositionality through architecture choices alone is rare, with our work to our knowledge being the first to take this approach in the planning domain.

## 4 Experiments

Our hypothesis is that local receptiveness and positional equivariance are key ingredients in explaining compositionality in diffusion planning, and that different ways of incorporating each (training augmentations, replanning, architecture choices, etc.) should increase compositionality. However, it is not known which of the two properties is more important, whether both are needed, or which way of inducting the two properties increases compositionality the most effectively.

To test our hypothesis that local receptiveness and shift equivariance are sufficient for composition in diffusion planners and answer our open questions, we train a variety of models on toy environments designed to test generated sequence diversity and multi-goal completion.[5]

**Environment Description**: Our main environment is a navigation task where an agent must navigate from one point in a grid to another while avoiding obstacles. Figure 1 shows examples of our grid environment, including points navigating to and from different locations in the grid. Additionally, to test training data scaling, we use a separate navigation environment where an agent moves between

[5]Our code can be found here: https://github.com/rvl-lab-utoronto/diffusion-stitching

ten points in euclidean space in different sequences of points. More details are deferred to Appendix A.3.

We chose toy navigation environments over complex benchmarks for a few of reasons. First, popular datasets which claim to test stitching either have testing start-goal pairs contained in the training set [24][6] or a very limited number of start and goal states [54]. Additionally, existing datasets do not report what subsequences are present in the training data, making it difficult to predict stitching behaviour, whereas we can control this in our environment. Finally, the grid layout of our environment makes quantifying diversity easy, as each stitched trajectory will be topologically distinct [57], allowing us to perform careful analysis of a given model's compositional ability.

**Data Collection for Diffusion Model Training**: We look at the diversity of trajectories generated by diffusion planners in two separate planning settings, and collect two seperate datasets in our grid environment.

The first is the *fixed-endpoint* setting, where the goal of the task is to reach a fixed end point from a fixed start point. Every training trajectory is optimal for the task, but the demonstration data includes multi-modal behaviour, as individual trajectories reach the end point via a different route. All trajectories have the same starting and ending state but take different paths through the grid, and the diffusion planner is tasked with generating new paths from the start to the goal composed of sub-trajectories seen in training in new configurations.

The second is the *end-point conditional* setting, where a wider range of trajectories provide sub-trajectories that need to be composed to meet novel start-goal pairs. In our environment, the goal is defined as reaching a goal state from a start state, where each intersection point along the edge of the grid is a legitimate start or goal state. To be precise, we attempt to learn a model that can sample from the distribution $P(\tau|s_0 = A, s_T = B)$. We collect data that includes diverse grid navigation to give the model ample sub-trajectories to compose, while being careful to keep some start-goal pairs unseen. We defer details to Appendix A.3.

**Algorithm Setup:** For our experiments we use a 1D U-Net Diffusion Planner as the base model. We use CleanDiffuser's [18] implementation and hyperparameters. One change we make is only predicting future states and using an inverse dynamics model to recover actions Ajay et al. [1]. [7].

## 5 Results

Here we report our findings on the unconditional and conditional environments, with the environments, data, and algorithms described in Section 4.

In our results, we count diversity by examining *topologically distinct* trajectories. Practically, each trajectory can split the obstacles into two groups: those on its left side, and on its right; two trajectories with different obstacle splits are topologically unique.

### 5.1 Unconditional Environment Results

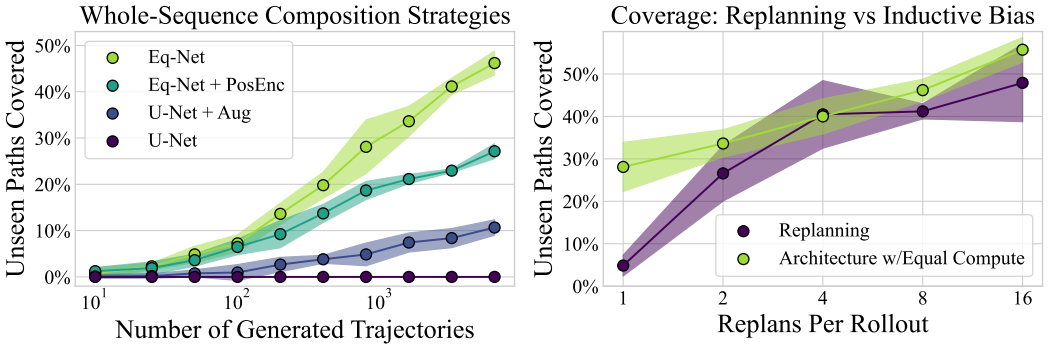
To quantify each model's ability to compose in the unconditional environment, we took our final trained models and sampled them at a range of sample counts. From these samples, we counted what percentage of the possible (but unseen in training) trajectory space each model's generations covered. By giving us a sense of how a model's coverage scales with the number of generations, this gives a sense of how capable each model's composition abilities are without solely relying on a model's behaviour at very high sampling counts.[8].

Figure 4 shows our results for the unconditional navigation environment. Figure 4a shows a comparison of the different whole-sequence composition strategies. The base U-Net fails to *ever* compose on our small dataset, only reproducing examples seen in training. Our model with approximate positional equivariance (granted through training augmentation) performs better, but fails to extrapolate, not

[6] Shown by Ghugare et al. [26].

[7] See Table 2 for details on hyperparameters and inverse dynamics.

[8] In our unconditional results, we filter to only analyze *successful trajectories* (ones that successfully reached the end state without hitting an obstacle) but all methods achieved very high success rates. Full completion rates are reported in Appendix A.2.1




(a) **Whole-Sequence Composition Strategies**: Our results show both locality *and* positional equivariance play a role in encouraging composition, with the model which has both (Eq-Net) outperforming the model with only locality (Eq-Net + PosEnc), only positional equivariance (U-Net + Aug) or neither (U-Net).

(b) **Coverage from Replanning vs Inductive Bias**: We find that while frequent replanning leads to more trajectory diversity, instead using the extra "budget" of diffusion model queries required during replanning on generating more samples from our strongest whole-sequence diffusion model produces similar or superior diversity.

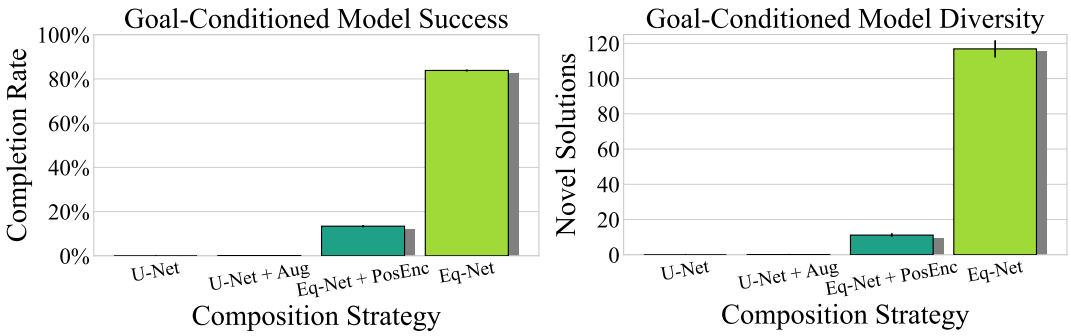
Figure 4: Results from testing our CNN diffusion planners on the unconditional navigation task. Error bars are 2-sigma taken over 5 samples.

covering a significant proportion of the unseen trajectory space even at very high sampling rates. Our equivariant network with added positional encoding (making the model local, but not positionally equivariant) performs much better, indicating locality is a *more important factor* in encouraging composition than positional equivariance. Lastly, our model that is both locally receptive and shift equivariant (Eq-Net) performs the best, exhibiting very strong whole-sequence composition from *architecture choices alone*, without utilizing data scaling, augmentation, or replanning.

Figure 4b analyzes replanning, where we progressively increase the number of times the model replans during a trajectory, and analyze the resulting diversity of generations over 800 independent generations. For our replanning model we used our U-Net trained with positional data augmentation, as we found it necessary to enable the model to generate trajectories from diverse start positions. Assume we wish to replan $N$ times in a task $T$ long. When replanning, we keep only the current state ($H_P = 1$ using our prior notation), generate a new trajectory starting at that point, execute $A = \frac{T}{N}$ steps, and then repeat until the goal is reached. Figure 4b shows that as expected, as $N \to T$, the diversity of generations increases. To compare this to our architectural strategy, we compare it against the best model from Figure 4. To directly compare the computational cost associated with replanning, we compare each agent that replans $N$ times against Eq-Net that gets $N$ times as many parallel samples to generate from, such that each point queries the diffusion model an equal amount of times. For instance, at $N = 4$, the replanning model over 800 independent samples is queried $800 * 4 = 3200$ times, as each trajectory requires 4 queries, so to have a fair comparison we query Eq-Net for 3200 independent samples.

Predictably, our results show that more frequent replanning indeed leads to more trajectory diversity. Interestingly, they also show that the extra budget of diffusion model queries required during replanning might be better spent generating more samples from a high-diversity whole-trajectory model, as Eq-Net performs similarly or slightly better on all samples.

Lastly, to validate that our results are not influenced by regularization induced by under-training of our diffusion planner, we analyze how each model's compositionality evolves throughout training. We find that while early stopping helps compositionality (perhaps acting as a form of regularization that prevents overfitting to the training samples as has been observed in other work [5]), models with stronger inductive biases preserve their compositional abilities as they trained for longer. See Appendix A.5.1 for details.




(a) **Goal-Conditioned Model Success**: Both positional equivariance and locality contribute heavily to strong model success rates. The Eq-Net model much more consistently generates trajectories that achieve the goal, are continuous, and do not hit obstacles.

(b) **Goal-Conditioned Model Diversity**: Both positional equivariance and locality also matter for the diversity of solutions generated, with the Eq-Net model producing far more viable candidate solutions than the other models.

Figure 5: Results in our goal-conditioned environment. Both success rate and successful trajectory candidate diversity are substantially increased when the diffusion planner is both local and equivariant to position. Error bars are 3-sigma taken over 5 samples.

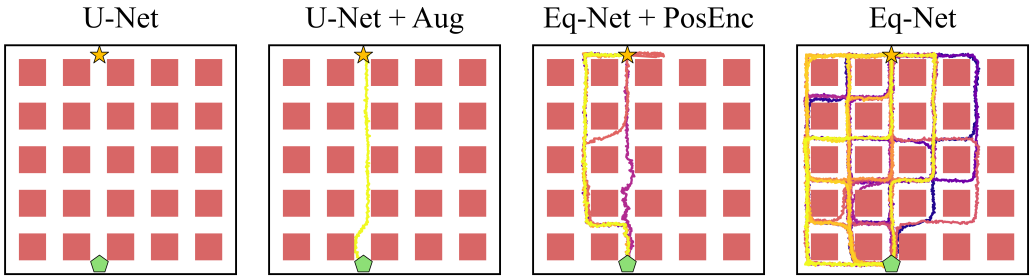



Figure 6: Successful trajectories from a random sample of 50 runs on one of the unseen start-goal pairs in our goal-conditioned environment. Eq-Net is able to produce a far greater diversity of canditate solutions than any other planner. Each colour represents a different generated trajectory.

### 5.2 Conditional Environment Results

In the conditional environment described in Section 4, the goal of the diffusion planner is to take unstructured trajectories in our environment and be able to consistently produce trajectories connecting an arbitrary start and goal location. To ensure that stitching is required at inference time, we examine every pair of edge intersections and filter out any pair where the starting pair and goal pair appear together sequentially, resulting in 11 test-time pairs. We run our whole-sequence planners on each pair 500 times, calculate an average completion rate and number of unique successful plans (where uniqueness is defined as topological distinctness), and take the average over 5 sample runs.

To guide the model to a specific goal, we use inpainting like in prior diffusion works [47, 34, 61]. We found that repeating the inpainted goal state significantly improved goal-reaching performance, so we repeat the goal state 25 times at the end of the trajectory before diffusing the intermediate states.

Our results are shown in Figure 5. The model that is both local and positionally equivariant far outperforms all other models, achieving an average 83.3% completion rate, roughly 7.5× higher than the model with only locality but not positional equivariance. Only having positional equivariance did not substantially help the model in this scenario, with the U-Net with data augmentation only achieving a non-zero result on one of the testing pairs and only a small amount of the time. Lastly, like in the unconditional case, the default U-Net fails to perform stitching entirely if it is not locally receptive or shift equivariant.

To visualize the difference in composition for each model, we plotted the unique successful runs from 50 random generations from each algorithm on one of the hold-out start-goal pairs. Figure 6 shows the relative diversity of each model: the base U-Net fails to ever generate a successful trajectory, the U-Net made approximately positionally equivariant with data augmentation can produce a single

candidate successful trajectory, the Eq-Net with positional equivariance removed produces a handful of plans, and the full Eq-Net produces a large diversity of successful trajectories.

### 5.3 Data Scaling Results

Lastly, we investigate the role of data in inducing composition. We use a different environment in these experiments, consisting of a navigation task where a point must sequentially navigate between five different points of a decagon. A compositional planner should be able to stitch sub-trajectories from individual points to other points. We train models with data ranging from from 25 to 25000 data points, where 25000 points is close to the number of possible combinations. Figure 7 shows that even models without strong inductive biases can learn to compose at large enough data sizes. Past a certain data quantity (roughly 2500 samples in our environment), the U-Net learns to compose, producing new stitched trajectories at test time. We provide evidence that this data scaling induces a smaller receptive field of the diffusion model in Appendix A.5.2. This confirms that the ability to generalize composition of sub-sequences is another property that diffusion models learn from data at scale, adding onto concept composition [53], consistency [68] and grammar [23].

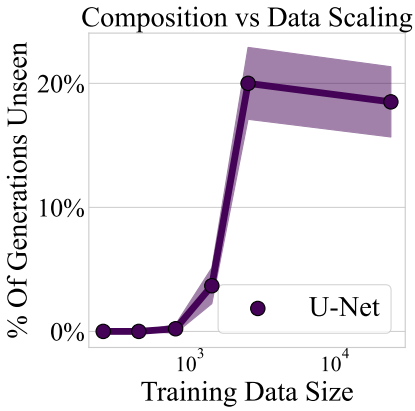



Figure 7: **Scaling training data quantity versus compositional ability**. Our results show that at small data sizes, models without strong inductive biases fail to compose, but gain this ability once a critical amount of training data is reached, generalizing to produce new combinations of sub-trajectories.

## 6 Discussion

### 6.1 What Are the Keys to Composition?

While our work attempts *explain* why existing works in diffusion planning achieve composition, a reasonable question to ask is what engineering insights our findings have. While our results indicate that achieving whole-sequence compositionality through architecture choices may be under-explored, we emphasize that different choices are valid in different contexts. For instance, in environments where reactive control is very effective, generating long-horizon sequences may not be important, so frequent re-planning with a short horizon may work better. In environments where data is abundant, using a higher-capacity model might be appropriate. We provide additional discussion of our findings in Appendix A.6.

### 6.2 Limitations and Future Work

Our work has several limitations. First, we only consider navigation environments with little complexity in the dynamics or observation space. While we motivate this in Section 4, evaluating our findings in a more realistic robotics environment would be useful future work. Second, our guidance strategy was limited to inpainting, which is effective in environments with clear endpoint goals, but less useful when goals are abstract (such as with language-conditioned agents) or multi-modal. Incorporating a way to leverage other guidance strategies like classifier-based [30] or free [31] guidance into our proposed high-compositionality diffusion planners is potentially fruitful future work. This would also let us fit our composition framework into the diffusion policy [13] setting, where composition of atomic skills may still be useful, but where a goal state cannot be inpainted because the diffusion model generates future actions and not future states.

## References

[1] Anurag Ajay et al. "Is Conditional Generative Modeling all you need for Decision Making?" In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=sP1fo2K9DFG.

[2] Giulio Biroli et al. “Dynamical regimes of diffusion models”. In: *Nature Communications* 15.1 (2024), p. 9957.

[3] Valerio Biscione and Jeffrey S Bowers. “Convolutional neural networks are not invariant to translation, but they can learn to be”. In: *Journal of Machine Learning Research* 22.229 (2021), pp. 1–28.

[4] Johan Bjorck et al. “Gr00t n1: An open foundation model for generalist humanoid robots”. In: *arXiv preprint arXiv:2503.14734* (2025).

[5] Tony Bonnaire et al. “Why Diffusion Models Don’t Memorize: The Role of Implicit Dynamical Regularization in Training”. In: *arXiv preprint arXiv:2505.17638* (2025).

[6] Anadi Chaman and Ivan Dokmanic. “Truly shift-invariant convolutional neural networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3773–3783.

[7] Stanley Chan et al. “Tutorial on diffusion models for imaging and vision”. In: *Foundations and Trends® in Computer Graphics and Vision* 16.4 (2024), pp. 322–471.

[8] Ian Char et al. “Bats: Best action trajectory stitching”. In: *arXiv preprint arXiv:2204.12026* (2022).

[9] Boyuan Chen et al. “Diffusion forcing: Next-token prediction meets full-sequence diffusion”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 24081–24125.

[10] Chang Chen et al. *Simple Hierarchical Planning with Diffusion*. 2024. arXiv: 2401.02644 [cs.LG]. URL: https://arxiv.org/abs/2401.02644.

[11] Haoxin Chen et al. “Videocrafter1: Open diffusion models for high-quality video generation”. In: *arXiv preprint arXiv:2310.19512* (2023).

[12] Lili Chen et al. “Decision transformer: Reinforcement learning via sequence modeling”. In: *Advances in neural information processing systems* 34 (2021), pp. 15084–15097.

[13] Cheng Chi et al. “Diffusion policy: Visuomotor policy learning via action diffusion”. In: *The International Journal of Robotics Research* (2023), p. 02783649241273668.

[14] Taco Cohen et al. “Equivariant convolutional networks”. PhD thesis. Taco Cohen, 2021.

[15] Taco Cohen and Max Welling. “Group equivariant convolutional networks”. In: *International conference on machine learning*. PMLR. 2016, pp. 2990–2999.

[16] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. “Causal confusion in imitation learning”. In: *Advances in neural information processing systems* 32 (2019).

[17] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[18] Zibin Dong et al. “CleanDiffuser: An Easy-to-use Modularized Library for Diffusion Models in Decision Making”. In: *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. 2024. URL: https://openreview.net/forum?id=7ey2ugXs36.

[19] Yilun Du and Leslie Pack Kaelbling. “Compositional Generative Modeling: A Single Model is Not All You Need”. In: *CoRR* (2024).

[20] Yilun Du, Shuang Li, and Igor Mordatch. “Compositional visual generation with energy based models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6637–6647.

[21] Scott Emmons et al. “RvS: What is Essential for Offline RL via Supervised Learning?” In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=S874XAIpkR-.

[22] Logan Engstrom et al. “Implementation matters in deep rl: A case study on ppo and trpo”. In: *International conference on learning representations*. 2019.

[23] Alessandro Favero et al. “How compositional generalization and creativity improve as diffusion models are trained”. In: *arXiv preprint arXiv:2502.12089* (2025).

[24] Justin Fu et al. “D4rl: Datasets for deep data-driven reinforcement learning”. In: *arXiv preprint arXiv:2004.07219* (2020).

[25] Abraham George and Amir Barati Farimani. “One act play: Single demonstration behavior cloning with action chunking transformers”. In: *arXiv preprint arXiv:2309.10175* (2023).

[26] Raj Ghugare et al. “Closing the Gap between TD Learning and Supervised Learning–A Generalisation Point of View”. In: *arXiv preprint arXiv:2401.11237* (2024).

[27] Chengyang He et al. *Demystifying Diffusion Policies: Action Memorization and Simple Lookup Table Alternatives*. 2025. arXiv: 2505.05787 [cs.RO]. URL: https://arxiv.org/abs/2505.05787.

[28] Charles A Hepburn and Giovanni Montana. "Model-based trajectory stitching for improved offline reinforcement learning". In: *arXiv preprint arXiv:2211.11603* (2022).

[29] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in neural information processing systems* 29 (2016).

[30] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[31] Jonathan Ho and Tim Salimans. "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021. URL: https://openreview.net/forum?id=qw8AKxfYbI.

[32] Physical Intelligence et al. $\pi_{0.5}$*: a Vision-Language-Action Model with Open-World Generalization*. 2025. arXiv: 2504.16054 [cs.LG]. URL: https://arxiv.org/abs/2504.16054.

[33] Alex Irpan. *Deep Reinforcement Learning Doesn't Work Yet*. www.alexirpan.com/2018/02/14/rl-hard.html. 2018.

[34] Michael Janner et al. "Planning with diffusion for flexible behavior synthesis". In: *arXiv preprint arXiv:2205.09991* (2022).

[35] Zahra Kadkhodaie et al. "Generalization in diffusion models arises from geometry-adaptive harmonic representations". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=ANvmVS2Yr0.

[36] Mason Kamb and Surya Ganguli. "An analytic theory of creativity in convolutional diffusion models". In: *arXiv preprint arXiv:2412.20292* (2024).

[37] Alex Krizhevsky, Vinod Nair, Geoffrey Hinton, et al. "The CIFAR-10 dataset". In: *online: http://www. cs. toronto. edu/kriz/cifar. html* 55.5 (2014), p. 2.

[38] Aviral Kumar et al. "When Should We Prefer Offine Reinforcement Learning Over Behavior Cloning?" In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=AP1MKT37rJ.

[39] Kyowoon Lee and Jaesik Choi. "State-Covering Trajectory Stitching for Diffusion Planners". In: *arXiv preprint arXiv:2506.00895* (2025).

[40] Yewon Lee et al. "Stamp: Differentiable task and motion planning via stein variational gradient descent". In: *arXiv preprint arXiv:2310.01775* (2023).

[41] Guanghe Li et al. "DiffStitch: Boosting Offline Reinforcement Learning with Diffusion-based Trajectory Stitching". In: *International Conference on Machine Learning*. PMLR. 2024, pp. 28597–28609.

[42] Wenhao Li et al. "Hierarchical diffusion for offline decision making". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 20035–20064.

[43] Jack Lindsey et al. "On the Biology of a Large Language Model". In: *Transformer Circuits Thread* (2025). URL: https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

[44] Nan Liu et al. "Learning to compose visual relations". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 23166–23178.

[45] Ilya Loshchilov, Frank Hutter, et al. "Fixing weight decay regularization in adam". In: *arXiv preprint arXiv:1711.05101* 5 (2017), p. 5.

[46] Haofei Lu et al. "What Makes a Good Diffusion Planner for Decision Making?" In: *The Thirteenth International Conference on Learning Representations*. 2025.

[47] Andreas Lugmayr et al. "Repaint: Inpainting using denoising diffusion probabilistic models". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11461–11471.

[48] Yunhao Luo et al. "Generative Trajectory Stitching through Diffusion Composition". In: *arXiv preprint arXiv:2503.05153* (2025).

[49] Nick McGreivy and Ammar Hakim. "Convolutional layers are equivariant to discrete shifts but not continuous translations". In: *arXiv preprint arXiv:2206.04979* (2022).

[50] Preetum Nakkiran et al. "Step-by-Step Diffusion: An Elementary Tutorial". In: *CoRR* abs/2406.08929 (2024). URL: https://doi.org/10.48550/arXiv.2406.08929.

[51] Tianwei Ni et al. “When do transformers shine in rl? decoupling memory from credit assignment”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 50429–50452.

[52] Matthew Niedoba et al. “Towards a Mechanistic Explanation of Diffusion Model Generalization”. In: *arXiv preprint arXiv:2411.19339* (2024).

[53] Maya Okawa et al. “Compositional abilities emerge multiplicatively: Exploring diffusion models on a synthetic task”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 50173–50195.

[54] Seohong Park et al. “OGBench: Benchmarking Offline Goal-Conditioned RL”. In: *arXiv preprint arXiv:2410.20092* (2024).

[55] Tim Pearce et al. “Imitating Human Behaviour with Diffusion Models”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=Pv1GPQzRrC8.

[56] William Peebles and Saining Xie. “Scalable diffusion models with transformers”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 4195–4205.

[57] Florian T Pokorny, Majd Hawasly, and Subramanian Ramamoorthy. “Topological trajectory classification with filtrations of simplicial complexes and persistent homology”. In: *The International Journal of Robotics Research* 35.1-3 (2016), pp. 204–223.

[58] Alec Radford et al. “Improving language understanding by generative pre-training. OpenAI”. In: *Preprint* (2018), pp. 1–12.

[59] Connor Shorten and Taghi M Khoshgoftaar. “A survey on image data augmentation for deep learning”. In: *Journal of big data* 6.1 (2019), pp. 1–48.

[60] Uriel Singer et al. “Make-a-video: Text-to-video generation without text-video data”. In: *arXiv preprint arXiv:2209.14792* (2022).

[61] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International conference on machine learning*. pmlr. 2015, pp. 2256–2265.

[62] Kiwhan Song et al. “History-Guided Video Diffusion”. In: *arXiv preprint arXiv:2502.06764* (2025).

[63] Toshihide Ubukata, Jialong Li, and Kenji Tei. “Diffusion model for planning: A systematic literature review”. In: *arXiv preprint arXiv:2408.10266* (2024).

[64] Hado Van Hasselt et al. “Deep reinforcement learning and the deadly triad”. In: *arXiv preprint arXiv:1812.02648* (2018).

[65] John J Vastola. “Generalization through variance: how noise shapes inductive biases in diffusion models”. In: *arXiv preprint arXiv:2504.12532* (2025).

[66] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[67] Dequan Wang et al. “Monocular plan view networks for autonomous driving”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 2876–2883.

[68] Huijie Zhang et al. “The emergence of reproducibility and consistency in diffusion models”. In: *Forty-first International Conference on Machine Learning*. 2024.

[69] Ji Zhang et al. “Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation”. In: *Journal of Field Robotics* 37.8 (2020), pp. 1300–1313.

[70] Richard Zhang. “Making convolutional networks shift-invariant again”. In: *International conference on machine learning*. PMLR. 2019, pp. 7324–7334.

[71] Siyuan Zhou et al. “Adaptive online replanning with diffusion models”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 44000–44016.

# A Appendix

**Contents**



### A.1 More Related Work

Prior to diffusion planners, other generative models were common in imitation learning [29]. Most modern approaches are either based around autoregressive decoder-only transformers [58], such as the Decision Transformer [12], ACT [25], or diffusion models [29]. A more comprehensive survey on Diffusion Planners was prepared by Ubukata, Li, and Tei [63].

We draw from a rich body of work in diffusion model generalization, which typically examine image-based models. Approaches include examining the effect of the diffusion process noise [65], architectural biases [36, 52], and representation learning [35] among others.

Another work that similarly analyzes architecture and inference strategies for diffusion planning is Lu et al. [46], which interestingly finds that the best diffusion backbone are transformers, which have *no* inductive biases towards locality and usually have positional attention added. However, their work is about general performance in offline RL and not about emphasizing compositionality, which as discussed may not be an important properly for strong performance on the D4RL benchmark due to a lack of required compositionality [26]. Analyzing how transformers differ from CNNs with respect to compositionality remains unexplored as future work.

When analyzing replanning, we apply a common form of it where replanning simply happens after some fixed number of execution steps. However, more sophisticated approaches to replanning exist

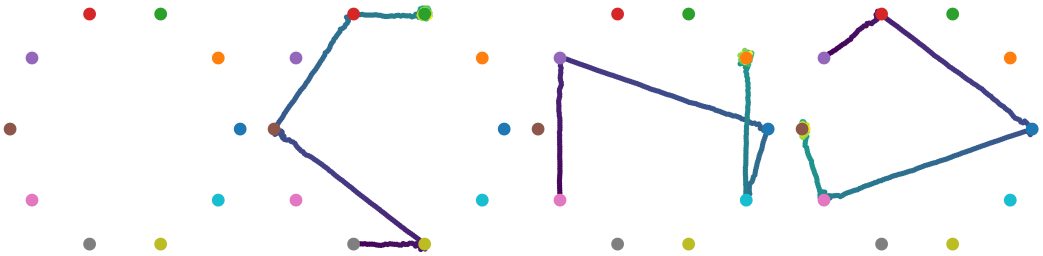


Figure 8: Visualizations of our Decagon environment. The left plot shows the empty environment, and the others show a trajectory in the training dataset.

which may have different effects on composition. For instance, Zhou et al. [71] adaptively re-plans when the estimated likelihood of the current partially executed trajectory falls too low.

### A.2 Additional Experiment Details

#### A.2.1 Unconditional Completion Rates

| | U-Net | U-Net+Aug | Eq-Net+PosEnc | Eq-Net | Replan (16) |
|---|---|---|---|---|---|
| Trials | 6400 | 6400 | 6400 | 128000 | 4000 |
| Completion | $1.0 \pm 0$ | $0.996 \pm 0.0021$ | $0.992 \pm 0.0032$ | $0.994 \pm 0.002$ | $1.0 \pm 0$ |

Table 1: Completion rates of algorithms in the unconditional environment. Confidence bounds represent a 3-sigma deviation, calculated assuming success is drawn from the binomial distribution.

Here we report completion rates in the unconditional environment. Table 1 shows the completion rates for each algorithm. As shown, every algorithm completes it close to $100\%$ of the time, with only the full-sequence models producing illegitimate trajectories a small portion (roughly $0.5\%$) of the time.

#### A.2.2 Data Scaling Experiment Details

For our data scaling experiments, we used a different environment, consisting of point navigation between points arranged in a decagon in euclidean space. Figure 8 shows a visualization. Each trajectory was tasked with travelling from point-to-point for a total of 5 visited points per trajectory. Data was collected with the same Gaussian noise as in the grid environment, and the trajectory length was also 500 states long. We trained each of our models in this context for 200000 gradient steps, with hyperparameters otherwise the same as in Table 2.

We ran each model 1000 times to approximate the distribution of the model's output. The metric we report in 7 is the percentage of generations that are not seen in the training set, where membership in the training set is determined by finding which key-points a trajectory visits in which order, and comparing them to that particular model's dataset.

Note that this metric is somewhat biased towards the models that can compose but are trained on less data, as the space of compositionally legitimate but unseen in training trajectories is larger than for the model that already has most of the support of legitimate trajectories in its training set. However, our results show that the threshold is fairly binary, as the very low data models will not or will only very rarely produce a novel composition, but past a threshold the models will consistently produce novel trajectories a large percentage of the time.

#### A.2.3 Algorithm Hyperparameters

Here we report the common hyperparameter we ran our experiments with, unless otherwise specified. Table 2 shows the full values.

Instead of jointly generating states and actions as in Janner et al. [34], we use a separate inverse dynamics model (as in Ajay et al. [1]). This removes the need to tune action versus state weighing in the diffusion objective, and generally performs better than joint prediction [46]. We execute

this inverse dynamics model with feedback, executing the action $a_t$ at each time step $t$ by giving the model the current state in the plan $p_t$ and the current state from the environment $s_t$, such that $a_t = \text{invdyn}(p_t, s_t)$.

#### A.2.4 Experiment Computing Resources

All experiments for this work were done on a laptop with a 13th Gen Intel Core i9-13950HX and NVIDIA GeForce RTX 4080 Laptop GPU. The final models used to produce the results shown in the paper were trained over roughly 2 weeks, and model inference to generate the final results took roughly 1 week.

### A.3 More Environment Details

#### A.3.1 Grid Environment Details

Our main environment is a small $6 \times 6$ grid where a point agent navigates from one part of the grid to another while avoiding a series obstacles. The observation is the x-y coordinates of the agent, and the action is a x-y step which is normalized to a small step size. In the middle of the maze are several obstacles, which will kill the agent if it stays inside of the block for several steps in a row. Figure 1 shows examples of our environment, including points navigating to and from different locations in the grid. Navigating from one corner of the grid to the other requires $\sim 400$ steps out of a total allowed episode length of $500$, challenging the diffusion planner's long-horizon planning. The obstacles also ensure the trajectory must remain compositionally legitimate over the whole horizon, as accumulating error can cause the agent to wander off of the grid to its death.

#### A.3.2 Grid Environment Possible Trajectories

**Lemma 1**: For a grid of size $N$ where $N$ is the number of blocks along the grid's height and width, the number of possible topologically distinct trajectories that are optimal in length is upper-bounded (inclusive) by $\frac{N!}{(N/2)!(N/2)!}$ .

**Proof**: First, observe that the number of optimal topologically distinct trajectories for any start-goal pair that is *not* from a corner to the opposite corner will be upper bounded by those that are. This is because they will have a smaller number of segments to traverse, meaning all of the analysis below applies with a smaller asymptotic quantity. This means finding the maximum number of distinct optimal trajectories in the corner to opposite corner case will upper bound the same quantity for any other start-goal pair.

Assume we are navigating from the top left to the bottom right corner for notational purposes. Observe that for any optimal path, the agent will move right $N/2$ times and down $N/2$ times. This means all possible trajectories can be represented by a binary string of length $\frac{N}{2} + \frac{N}{2} = N$ where there are an

| Hyperparameter | Value |
|---|---|
| EMA Rate | 0.9999 |
| Diffusion Steps | 100 |
| Optimizer | AdamW [45] |
| Base LR | 2e-4 |
| Weight Decay | 1e-5 |
| LR Schedule | Cosine |
| Predict Noise or Clean | Clean |
| Sampling Temperature | 0.5 |
| Batch Size | 128 |
| Gradient Steps | 500,000 |
| Training Horizon Length | 512 |
| Convolution Padding | Repeat |
| Solver | DDPM[30] |
| SDE Form | Discrete[30] |

Table 2: The common hyper-parameters used in our experiments, unless otherwise specified.

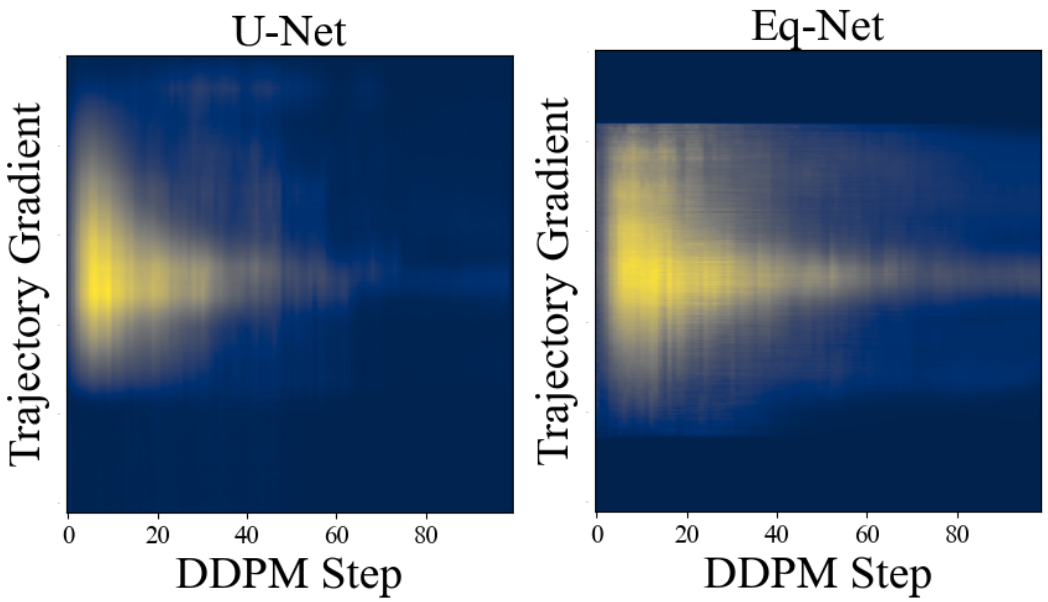



Figure 9: Denoising gradient maps of the centre pixel averaged over 50 generations for U-Net and Eq-Net. Yellow indicates a larger gradient magnitude, and blue smaller. U-Net has non-zero gradient information even at the edges of the sequence early in the denoising process (towards the left of the gradient map), indicating it attends to nearly the whole sequence in a single step. Comparatively, Eq-Net has a hard locality constraint set through its architecture, as shown by the lack of any gradient information past a fixed window around the centre pixel.

equal number of 1s and 0s. The size of this set is $(\frac{N}{N/2}) = \frac{N!}{(N/2)!(N/2)!}$, because from $N$ positions we choose $N/2$ to be filled with either 0s or 1s.

#### A.3.3 Data Collection

In the unconditional setting, each trajectory starts in the top-left corner and navigates to the bottom-right corner, randomly switching from moving either downwards or right at each intersection. In total, our unconditional dataset includes $42$ unique trajectories out of a possible $252$ trajectories.[9] This data quantity means our diffusion planner must generalize from a small number of samples to be able to generate samples in the (much larger) space of unseen but legitimate trajectories.

In the conditional setting, we replicate the above data collection process, but for all four corners, to ensure a diverse set of subsequences. We additionally add data starting from each edge intersection point going towards $4$ other randomly selected edge intersection points. This ensures that the dataset contains a wide range of subsequences, but that the majority of start-goal states are not seen during training. This dataset contains $233$ trajectories.

### A.4 Further Architecture Details and Experiments

#### A.4.1 U-Net is Non-Local and Not Shift Equivariant

**U-Net is Not Local**: Here we support the claim made in the main text that the U-Net architecture proposed by Janner et al. [34] is not shift equivariant and displays heavy non-locality during the denoising process.

To show this, we follow the procedure of prior works in analyzing diffusion model local field receptiveness by calculating the gradient of an intermediate sample point with respect to the input sequence. We took our final trained U-Net and Eq-Net models in the unconditional environment and generated $50$ samples. For each of the $100$ denoising steps, we calculated the gradient of the middle pixel in

[9] See Appendix A.3 for proof.

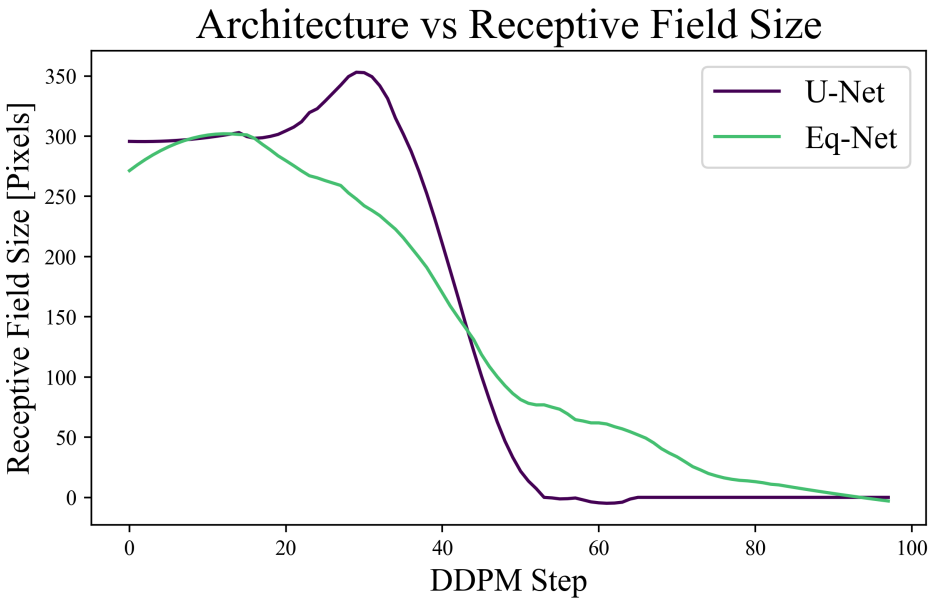



Figure 10: The approximate local receptive field of different model architectures. U-Net has a larger receptive field early during denoising than Eq-Net.

the sequence with respect to the whole input sequence (possible because of auto-differentiation), and averaged these gradients across the 50 generations for each time step. Formally, for a diffusion model $x_{t+1} = f(x_t, t)$ where $x_{t+1}$ is the next de-noised sample, $x_t$ is the previous sample, and $t$ is the denoising step, we calculate $|\frac{\partial x_{t+1}[250]}{\partial x_t}| = |[\frac{\partial x_{t+1}[250]}{\partial x_t[1]}, \frac{\partial x_{t+1}[250]}{\partial x_t[2]}, ..., \frac{\partial x_{t+1}[250]}{\partial x_t[250]}]^T|$ for each diffusion time-step from $1 \rightarrow T$. Because we generate a 1-D sequence, we can plot these gradient maps as an image, where the x-axis is the denoising step and the y-axis represents the sequence gradient.

Figure 9 shows these gradient maps. Visually, it is clear that the U-Net attends early in the denoising process to parts of the sequence far away from the centre pixel, even though the influence is much less strong than the pixels directly surrounding the centre. The size of the receptive field early in training spans far, containing non-zero information as far as the boundary states of the trajectory. On the other hand, Eq-Net has a hard boundary roughly 150 pixels to each side of the centre pixel past which the model cannot attend to in a single denoising step. Like in prior works [36, 52], our diffusion models' receptive fields progressively shrink during training, starting large and ending small.

We also plot a rough estimate of the local receptive field size at each denoising step. We do this the same way as Niedoba et al. [52]. We set a threshold value of the 75th percentile of the value of the absolute gradients in the image, find the window centred around the middle pixel that contains all values in that threshold for a given denoising step, and then plot the size of that window. Figure 10 shows the results of this, smoothed with a Savitzky-Golay filter. This plot shows that as expected, the local receptive field of the U-Net has a larger maximum value, approaching 350 pixels wide at its max, while the Eq-Net is architecturally capped at $\sim 300$.

This is a somewhat surprising result, given that Janner et al. [34] specifically highlights the local receptiveness from convolutions as supporting composition. Given that our results in both the conditional and unconditional setting found that the U-Net architecture struggles with composition, we directly analyzed the network during the diffusion process to gain some insight into whether the model is sufficiently local. We suspect part of the reason for this is because while stride 1 convolutions are locally receptive if applied to the whole sequence, pooling operations and long-stride convolutions both down-sample the image substantially, such that the internal trajectory length at the bottle-neck of the U-Net may be small enough that even a small convolutional kernel is capable of attending to information far across the sequence.

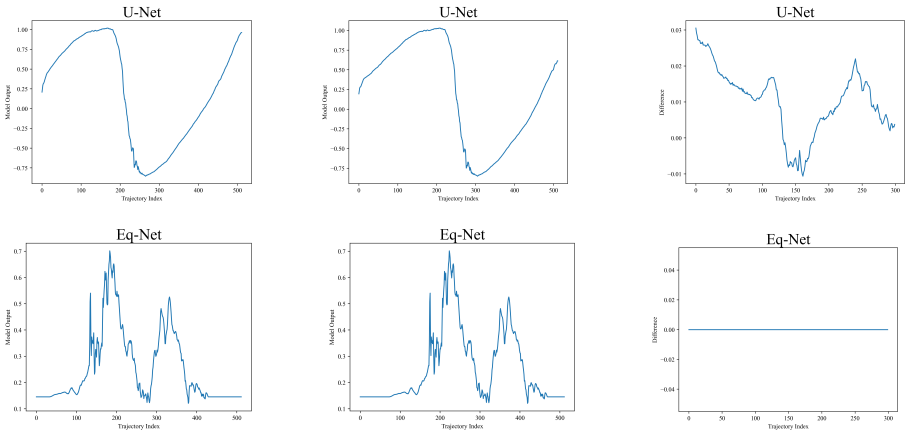



Figure 11: A small experiment showing that the U-Net architecture is not shift equivariant, while our Eq-Net is. The U-Net architecture has substantial differences in the output when shifted, shown as a different in the rightmost plot, while Eq-Net retains the same output but translated.

**U-Net is Not Shift Equivariant**

Strictly speaking, empirical evidence that U-Net is not shift equivariant or that Eq-Net is should not be required, as it directly follows from known properties of their architecture: the U-Net has down-sampling strides which break shift equivariance [70], and Eq-Net has only 1-length strides with no pooling layers which preserves convolutions' shift equivariance.

For the sake of visual demonstration, we show this by taking our trained U-Net and Eq-Net models in the unconditional environment and giving the model an input sequence of all zeros, besides a patch of Gaussian noise in the middle of the trajectory. We then shift this Gaussian patch by $40$ pixels and analyze the output of each model. Formally, we take two initially empty trajectories $\tau_1, \tau_2 = [0, ...0]^T$ and add a patch of Gaussian noise $50$ pixels wide to the middle of the first trajectory: $n \sim \mathcal{N}(0, 10)$,$\tau_1[225:275] = n$ and add the same Gaussian noise shifted by $40$ pixels from the middle to the second trajectory: $\tau_2[265:315] = n$. Finally, we run each through a given diffusion model at $t = 0$: $x_1 = f(\tau_1, 0), x_2 = f(\tau_2, 0)$. A model with shift equivariance should find that the pixels of the output de-noised trajectory corresponding to the same patch as the Gaussian noise should be equal, that is, $x_1[225:275] = x_2[265:315]$.

Figure 11 shows the output of each model at the original input position and the input position shifted to the right, as well as the difference between the outputs when the shift is corrected for. The U-Net architecture has substantial differences in the output when shifted, shown as a difference in the rightmost plot, while Eq-Net retains the same output but translated.

### A.4.2 Engineering Details for Eq-Net

The main purpose of Eq-Net is not to propose a competitive architecture, but to show a proof of concept that locality and shift equivariance can be baked into architecture to enable stitching. Given this, we did not aggressively tune the architecture parameters of Eq-Net, instead finding values for network depth, kernel size, and channel size that worked well in our benchmark. However, we found it has some limitations and advantages that we will discuss here.

Eq-Net is substantially more memory efficient than the base U-Net model, with the base U-Net having $15.78$ million parameters and Eq-Net only having $1.04$ million parameters.

Despite this, Eq-Net was around half as efficient at both training and inference time. We suspect the former is because the channel sizes grew to be much larger in the intermediate representation layers of the U-Net ($512$ at maximum) while Eq-Net has a fixed channel size of $64$ throughout. The latter is likely because Eq-Net is $25$ layers deep, which prevents efficient forwards passes because fewer operations can be parallelized.

Future work adapting these findings into a more practical architecture might investigate using adaptive polyphase sampling [6], anti-aliasing layers [70], or other modern tricks for incorporating

shift equivariance into CNNs. Additionally, the depth of the network, way the layers of differing kernel sizes are layered, and channel size can likely be greatly optimized to increase computational efficiency and model capacity.

### A.4.3 Kernel Size Tuning

The main parameter that requires tuning in Eq-Net is the *kernel expansion rate* (KER) which determines how fast the kernels grow deeper into the network. In Eq-Net, the kernels in each layer starts as small as possible (3) and grows 2 larger every KER layers, meaning for a fixed depth of 25, the maximum kernel size the network has is determined by $\sim 25/\text{KER}$. Thus, a small KER corresponds to a network with more larger kernels, and a larger KER with many small kernels.

Broadly, we found that tuning KER trades off between *overall plan consistency* (how likely the trajectories were to be locally consistent and stay outside the obstacles) and *compositionality*. To show this, we trained Eq-Net models in our unconditional setting with 3 KER values: 2, 5, and 12. Our results are shown in Table 3. We found that the intermediate value of 5 generated both a strong diversity of plans as well as reliably well-composed plans, while making the model too non-local made the model too local failed to consistently produce legitimate composed trajectories.

| | Large (KER = 2) | Medium (KER = 5) | Small (KER = 12) |
|---|---|---|---|
| Unique Trajectories | 0 | $59 \pm 12.1$ | $158 \pm 10.7$ |
| Completion Rate | $1.0 \pm 0.0$ | $0.994 \pm 0.002$ | $0.538 \pm 0.015$ |

Table 3: Results of changing the kernel expansion rate in the unconditional environment, taken over 800 generations. Error bars are 2sigma taken over 5 samples.

We also generated denoising gradient graphs of the small and large kernel Eq-Nets, like in our analysis of U-Net and our final Eq-Net architecture. Figure 13 shows the gradient graphs. As expected, Eq-Net with large kernels is attentive to information across the trajectory, and interestingly produces a gradient map extremely similar to the default U-Net, while Eq-Net with small kernels has an extremely limited receptive field.

Our experiments with the default U-Net architecture and our Eq-Net architectures in total confirm our earlier hypothesis that while CNNs are meant to have strong inductive biases to encourage composition, the particular set of decisions made by Janner et al. [34] means at small data amounts they can still overfit heavily and attend to non-local informatin,

### A.4.4 Kernel Diversity

One thing that we found was important when designing Eq-Net was to include both very small (such as k=3) along with the larger kernel layers. When we used only medium sized kernels, it would lead

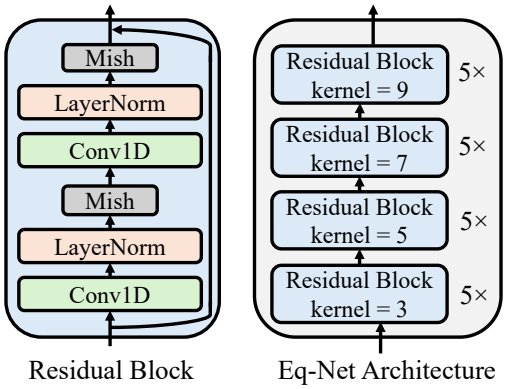



Figure 12: Our fully local, shift equivariant architecture. By stacking stride 1 convolutions with a small kernel length and removing pooling and down-striding, we ensure the CNN cannot attend to position within the trajectory or far-away states during denoising.

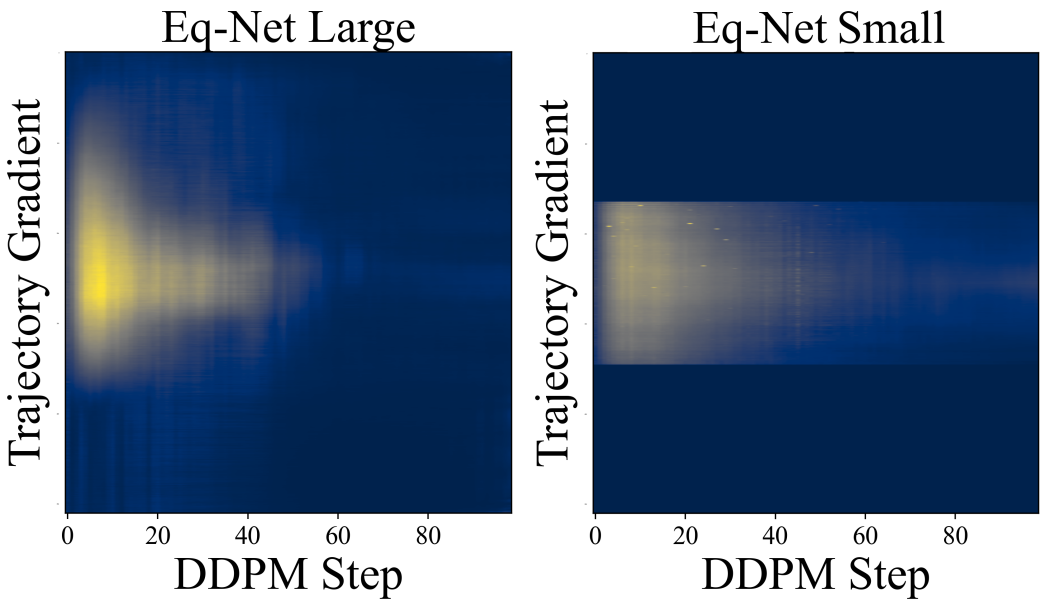



Figure 13: Denoising gradient maps of the centre pixel averaged over 50 generations for Eq-Net with large and small kernels. Eq-Net with large kernels is attentive to information across the trajectory, and produces a gradient map extremely similar to the default U-Net, while Eq-Net with small kernels has an extremely limited receptive field.

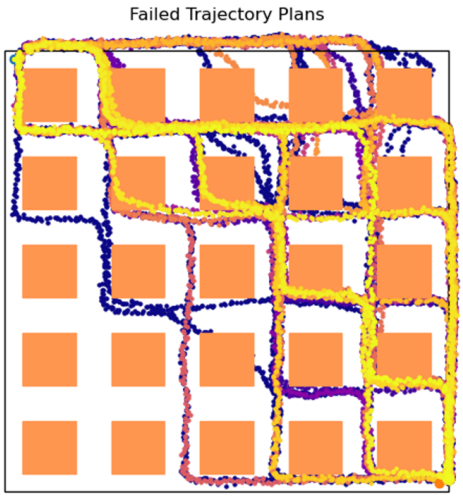



Figure 14: Generated plans when only using medium-sized kernels. Trajectories end up "ungrounded" to the grid, resulting in failed executions. Each colour represents a different generated trajectory.

to trajectories that were oddly "ungrounded" to the navigation plane, and would look correct in shape but would be shifted such that the resulting trajectory would nearly always hit obstacles. Figure 14 shows the visual results of this. We found that increasing the kernel size from small to large during training helped, somewhat mirroring the process in normal CNNs where fixed-size kernels stride over the whole image initially (capturing fine detail) and then smaller subsets after each downstride (capturing coarse detail). However, we did not explore whether the exact ordering of the kernel layers in Eq-Net matters or not.

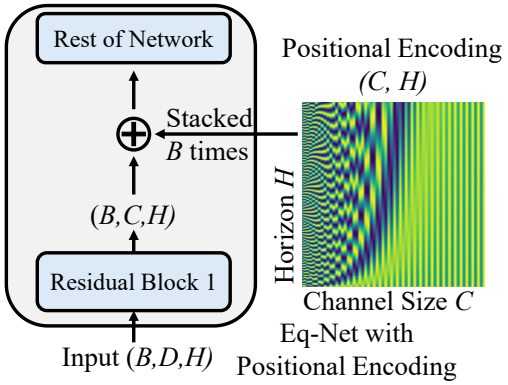



Figure 15: Our positional encoding approach. Sinusoidal embeddings which encode position channel-wise are generated, with a unique embedding for each position in the sequnce. These embeddings are then added after the first convolutional layer.

### A.4.5 Positional Encoding

When testing how our small receptive field model performs when it has access to positional information for each state, we use the same positional encodings as in Vaswani et al. [66], with slightly varied parameters. Figure 15 shows a visualization of how we add positional encoding to the Eq-Net architecture. These encodings turn the position of each state into waves forming a geometric progression from which the position of a particular state can be inferred. The encodings are added after the first residual block. Specifically, the positional encodings are defined as

$\text{PE}_(\text{pos}, 2i) = sin(\text{pos}/1000^{2i/H})$

$\text{PE}_(\text{pos}, 2i+1) = cos(\text{pos}/1000^{2i/H})$

We used a maximum length of $1000$ instead of $10000$ used by Vaswani et al. [66] because of the shorter length of our sequences.

It is worth noting that this strength of positional encoding (where the network is given enough information to precisely attend to the location of a state in the sequence) is likely quite a bit stronger than the positional attention from CNN aliasing. We included this as a representation of the worst-case scenario, to show the upper bound of models that are designed to attend to position, such as diffusion transformer architectures [56].

## A.5 More Experiments

### A.5.1 Overfitting Analysis

Here we analyze the effect of overfitting. A common strategy in all machine learning to increase generalization is to perform early stopping, where training is ended before loss on the training set plateaus. Here, we run all of the models shown in the unconditional experiments at gradient increments of $50000$ and run each model for $800$ samples, tracking both their completion rates and number of unique trajectories. Figure 16 shows that overfitting indeed reduces compositionality in the small-data setting, as models early in training achieve both high completion rates (indicating the generated plans are kinematically feasible) as well as more novel trajectories than later in training. However, the models that show stronger compositionality (such as Eq-Net) retain this property late into training (i.e, the losses in compositionality from overfitting plateaus). This hopefully indicates that the proper inductive biases from architecture choices can avoid the overfitting problem.

### A.5.2 Does Data Scaling Increase Locality?

In the Results section, we found that scaling data increased the compositionality of our base U-Net model, even though it does not have a strong inductive bias towards locality. To see if the compositionality induced by data scaling is attributable to data increasing locality, we analyzed the

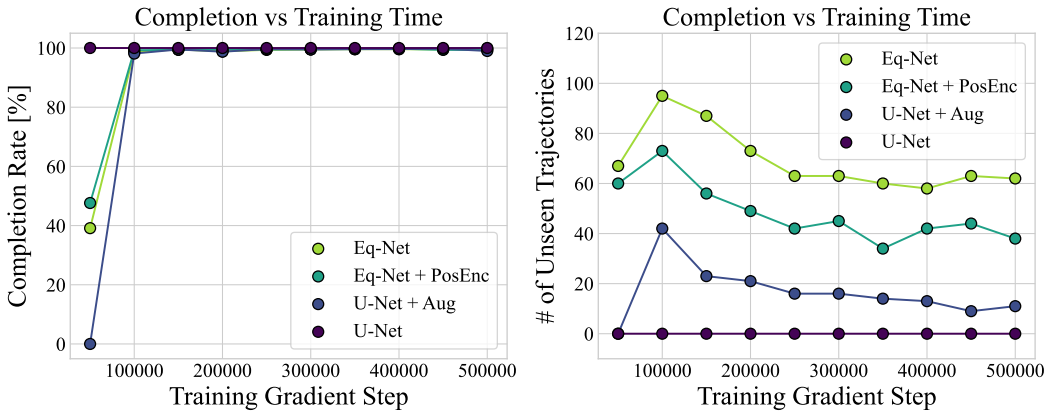



Figure 16: Completion rates and counted unique novel trajectories of each model in the unconditional setting, sampled 800 times, at various training checkpoints. Our results show that models lose some compositional ability when over-trained, but that the more compositional models retain compositionality in the training step limit. Note: the first data point is at 50000 gradient steps.

models as done previously in Appendix A.4.1 by recording the gradient of the middle pixel for 25 samples across each denoising step for each model. We then plot these directly, and also plot the estimated size of the local receptive field for each model. Following Niedoba et al. [52], to plot the receptive field we set a threshold value of the 75th percentile of the value of the absolute gradients in the image, found the window centred around the middle pixel that contains all values in that threshold for a given denoising step, and then plotted the size of that window.

Figure 17 shows the gradient maps. Note that we name each data scale, with the amount of data corresponding to each name appearing in parentheses (SMM = Small Medium Medium, SM = Small Medium, SSM = Small Small Medium). Visually, it can be seen that the receptive field of the largest model (Full) is smaller than the field of the smallest model (Small) early in the denoising process. The small model attends to information clearly at the bounds of the image around step 20 of the denoising process, while the full data model does not have substantial gradient information past around 150 pixels away from the centre on either end.

Figure 18 shows the plotted local receptive fields, smoothed with a Savitzky-Golay filter. The model trained on the highest data scale has a clearly smaller receptive field compared to the models trained on less data. This indicates that the reason why data scaling may help composition is because it trains the model to have a smaller receptive field, highlighting locality's importance for composition and the usefulness of our theory. More work should be done to understand why data scaling specifically induces such a property.

## A.6 More Discussion

### A.6.1 Broader Impacts

Our work does not have any broader societal impacts beyond the general impacts that sequential-decision making research may have on automation or enabling more intelligent agents. While our work is limited to planning, some of our finding might generalize to diffusion models more broadly, which have implications for copyright/artist's rights, deepfake generation, et cetera.

### A.6.2 How Should We Select a Method for Composition?

While we focus on the whole-sequence generation case, it is not clear what the strongest method is to generate heavily temporally correlated, sequential data. There is evidence that models trained only or mostly on auto-regressive prediction can still learn to do sophisticated forms of forwards and backwards planning [43]. Complex sequential generation models in video sometimes incorporate a form of auto-regressive prediction [62] but often generate the whole sequence in parallel with diffusion [60, 11].

Hierarchical diffusion appears to be a strong way to incorporate both the benefits of long-horizon planning and short-horizon consistency. Besides the empirical benefits shown in the hierarchical

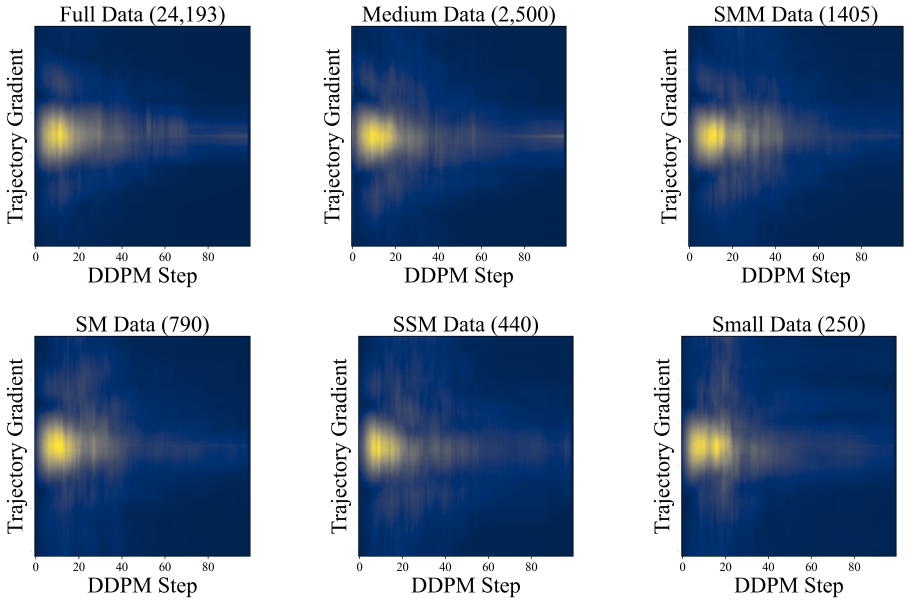



Figure 17: The denoising process gradient maps for the centre pixel for each model trained in our data scaling experiment. Each number in the subfigure titles corresponds to the exact amount of training data that model was given. Visual inspection shows that for the models trained on more data, the width of the receptive field early in the denoising process is smaller than for the models trained on less data. For instance, the Small Data model has significant gradient information that spans nearly the whole trajectory near DDPM step 20, whereas such steps with significant non-local information are not present in the Full Data model.

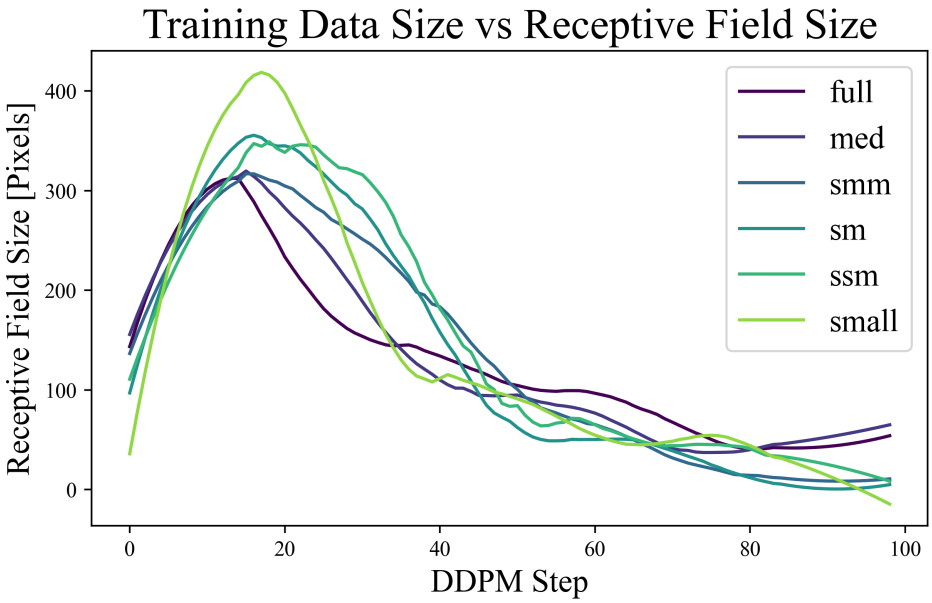



Figure 18: The approximate local receptive field of the models trained at each data scale, plotted together. Each model was taken from our training data scaling experiment in our euclidean navigation task, where each sequence is 500 time-steps long. Each pixel on the y-axis represents a single state in the trajectory. The model trained on the highest data scale (full) has a clearly smaller receptive field compared to the models trained on less data early in the denoising process.

diffusion planning papers discussed in 2, more evidence exists that hierarchical models can learn to minimize dynamics error in fewer training steps [19]. One downside of these methods is that how the task is hierarchically decomposed either needs to be set manually, requiring more tuning, or must rely on a more complex algorithm [42] which increases implementation cost and other tuning. Image models have generally shown that forgoing manually set hierarchy and instead letting models jointly learn both high level structure and low level detail can work well given proper data and architecture decisions.

One consideration we do not explicitly examine is that using locality to achieve composition necessarily hurts the model's ability to use important information from prior states. Long memory can be both a blessing and a curse, as it has found to increase a policy's ability to handle very distant observations [51] but can also replicate undesired correlations (i.e, correlations that are circumstantial and which the model should not replicate) between past observations and future actions [16, 67]. The lack of compositionality of non-locally attentive models can be thought of as a form of causal confusion, as they "confuse" specific transitions as being associated with state visitations in the distant pass that may not hold relevance for the future plan. However, recent work has found that taking a linear combination of the conditioned diffusion steps on a variety of past history lengths may preserve the benefits of both long-horizon planning and short-horizon composition, so it may be possible to combine the benefits of both [62].

### A.6.3 Explaining Composition in Other Diffusion Planning Works

Here, we apply our framework to a couple of other works in the diffusion planning literature.

**Hierarchical Diffusion Planning**: As discussed in Section 2, several works approach the composition problem by separately generating sub-sequences with a diffusion model, and then composing them. Exactly how these low-level subsequences are "orchestrated" into a single coherent plan varies, but each exhibits aspects of locality and shift equivariance.

Chen et al. [10] propose Hierarchical Diffuser, which uses two separate diffusion models: a high level one which generates a "plan sketch", and a low level one which generates a sequence connecting the states in the plan sketch to "fill-in" the high-level plan. Functionally similar was the prior HDI paper from Li et al. [42], which has a different form of generating sub-goals unrelated to diffusion. These approaches enable a form of shift equivariance by data augmentation: because the low-level subsequences generated by the low-level diffusion model are randomly cut out of the longer trajectories in the training data, they have their relative position within longer sequences removed. This also enables locality by disentangling the joint prediction of states within a particular subsequence to any other states outside of it, so each subsequence functions as a small local window without connectivity to previous or future states, preventing confounding from preventing stitching. Chen et al. [10] also confirms some of our findings that Diffuser with a larger kernel size often produces more coherent long-term plans (4.3) but that this trades-off with worse ODD compositional generalization (Appendix E).

Luo et al. [48] proposes a hierarchical method called CompDiffuser which, like the Hierarchical Diffuser paper, generates sub-sequences. Instead of having a separate high-level planner, they have each sub-sequence condition its generation on its immediate surrounding neighbour subsequence (3.2). This preserves shift equivariance in the same manner as Hierarchical Diffuser, but maintains locality through this process of having each sub-sequence only attend to its neighbours at each step, which stops causal confusion.

**Diffusion Forcing**: Some recent works extend diffusion into very long-horizon generation through using a combination of auto-regressive sampling and diffusion sampling called *diffusion forcing*. Diffusion forcing denoises over a whole sequence, but iteratively denoises near-future states more aggressively, leading to the sequence being progressively denoised from start to finish instead of the entire sequence being denoised consecutively. As discussed in the main text, short-memory and future prediction enforces locality by reducing the ability for long-horizon confounding effects to prevent composition.

Results from Chen et al. [9] shows that this approach can enable compositionality, but only when keeping short or no memory and only generating short plans, essentially using the diffusion forcing model to approximate short-horizon Model Predictive Control (E.2). When extending the generation length or memory, they find that it instead reproduces existing trajectories.

Follow up work from Song et al. [62] extends Diffusion Forcing by replacing their RNN-based backbone with a transformer, and introducing the idea of combining the scores of the same model conditioned with several history lengths to try to achieve both long-sequence compositionality and short-term reactivity. They tested this approach on a task where a robot must take a fruit in a slot and move it to another slot. This task requires both memory (of which fruit the model should move) and short term reactivity (a human perturbs the robot mid-trajectory). However, the training data does not have both memory of the original fruit location and examples of recovery behaviour from the human perturbance, requiring composition of training sub-sequences. They found that the model that keeps the full trajectory history overfits to the training datasets showing the original fruit, while only using short-horizon models would compose recovery behaviour but forget the original task. Ultimately they found combining the score produced trajectories that both remembered important fast information and were capable of composing sub-trajectories for reactive control. This supports our argument highlighting the importance of locality for composition, but highlights that locality alone can run into issues in long-horizon tasks, as discussed in Section 6.1. For details, see Sections 6.4: Task 3, C.8, and D.4.

**Diffusion Policies/VLAs**

Perhaps the most popular diffusion-based sequential decision making algorithms today are various forms of Diffusion Policies [13]. These are essentially similar to the prior Diffusion Behaviour Cloning framework [55] where the diffusion model acts as an explicit policy and generates an action, except that Diffusion Policy predicts a short horizon of actions that are executed. This framework of predicting a short horizon of actions with diffusion is also used by most modern foundation-model Vision Language Action models like Pi0.5 [32] or Gr00T [4].

This differs from the diffusion planning framework in that a long-horizon plan is not explicitly generated, so any long-horizon task guidance is obtained from conditioning the policy (like in goal-conditioned BC) rather than from explicit planning. Thus, in these models compositionality essentially comes from the same place that it does in TD learning: because no planning is done, the model composes through replanning. This replanning, like any explicit one-step or few-step policy, assumes the environment is Markovian and so does not include temporal information or preserve long history as conditioning, thus preserving shift equivariance and locality as we discussed in 3.2.

There is some limited early evidence that when trained from scratch on very small datasets, Diffusion Policies may perform learning more similar to memorization with nearest-neighbour interpolation than "generalization" in any meaningful sense, and thus obtain strong apparent trajectory composition just through being very reactive [27]. However, these results have only been found in very small data settings, so it is likely that they do not apply to larger VLAs or to multi-task Diffusion Policies trained on a heterogeneous mix of tasks.