Equivariant Denoisers Cannot Copy Graphs: Align Your Graph Diffusion Models

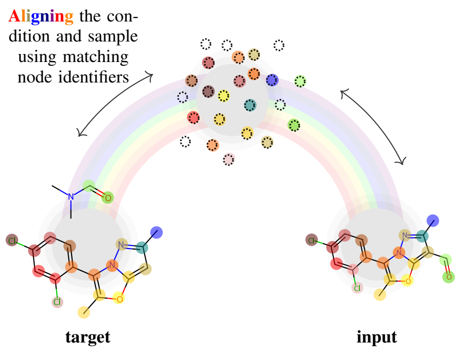
Anonymous authors
Paper under double-blind review

Abstract

Graph diffusion models, while dominant in graph generative modeling, remain relatively underexplored for graph-to-graph translation tasks like chemical reaction prediction. We show that standard permutation equivariant denoisers cause severe limitations on such tasks, a problem that we pinpoint to their inability to break symmetries present in the noisy inputs. We then propose to *align* the input and target graphs in order to break the input symmetries, while retaining permutation equivariance in the non-matching portions of the graph. We choose retrosynthesis as an application domain, and show how alignment takes the performance of a discrete diffusion model from a mere 5% to a SOTA-matching 54.7% top-1 accuracy.

1 Introduction

Graphs are a ubiquitous in a number of domains, ranging from knowledge representation to drug discovery (Ji et al., 2022; Vignac et al., 2023; Hoogeboom et al., 2022b). Graph-to-graph translation in particular is relevant to a number of applications, including molecular editing (Jin et al.), where the goal is to generate a molecular graph with a specific structure or properties, chemical reaction prediction (Shi et al., 2020b), requiring predicting a reactant molecular graph from a product molecular graph and vice-versa, and in general any task where the goal is to predict a future state of a graph given an initial configuration (Rossi et al., 2020). Using a generative model for graph-to-graph translation is a natural approach, allowing a probabilistic treatment of the task. Graph diffusion models are a particularly promising framework, given their excellent performance (Niu et al., 2020; Vignac et al., 2023) on graph data, and the rapid of diffusion models in generative modelling more widely.


Aligning the condition and sample using matching node identifiers
N
O
Cl
target
input

Aligned permutation equivariance

Figure 1: Nodes in the input and output graphs are given the same identifiers to enforce alignment, while the model is free to remain permutation equivariant w.r.t to the unmatched nodes.

Permutation equivariance is a fundamental inductive bias in most machine learning models operating on graphs, including graph diffusion models. This property ensures that the model's output remains consistent when input nodes are reordered or relabeled, thus eliminating the need for data augmentation through node permutations. While permutation equivariance is a powerful inductive bias, our paper demonstrates both theoretically and experimentally that it can be substantially sub-optimal for graph-to-graph conditional generation, especially in scenarios where there exists a competing inductive bias suggesting structural similarity between the input and output graphs. We identify the root cause of this issue: the symmetry inherent in samples drawn from the diffusion model prior (illustrated in the center of Fig. 1) and the neural network's inability to break these symmetries. Our findings align with previous and concurrent research highlighting the limitations imposed by equivariance and symmetries in neural networks (Lawrence et al., 2024; Xie & Smidt, 2024).

The key question then becomes how to give a model the ability to break symmetries while keeping the inductive bias of equivariance. We then propose a solution: *aligned* equivariance. The idea is to use identifiers in the input and target graphs to align their nodes when possible, relaxing equivariance

Sec. 5 The Theoretical Limitations of Equivariant Denoisers	
Copying graphs as a case-study on the limitations of equivariance.	Fig. 4
The optimal permutation equivariant denoiser.	Theorem 1
Sec. 6 Solution: Aligned Equivariant Denoisers	
Relaxing permutation equivariance through alignment.	Fig. 6a
Aligned denoisers induce aligned permutation invariant distributions.	Theorem 2
Sec. 7 Suitable Alignment Methods	
Skip-connection, positional encodings, input-alignment.	Fig. A9
Sec. 9 Experiments	
Proof of concept through simple graph copying tasks.	Fig. 5
Application: Retrosynthesis.	Table 2
Guided-generation for retrosynthesis.	Fig. 8a
Ibuprofen synthesis through inpainting.	Fig. 7

Figure 2: Overview of our contributions.

when the identifiers match. This helps break the symmetries of the input while retaining the benefits of permutation equivariance on non-matching subgraphs. We visualise this in Fig. 1. We summarize our contributions in Fig. 2.

2 Related Work

Graph-to-graph translation methods Graph-to-graph models have shown promise in tasks such as handwritten mathematical expression recognition Wu et al. (2021), molecular optimization Jin et al. (2019), and retrosynthesis Lin et al. (2023). More recently, Igashov et al. (2024) propose a Markov bridge model to map from the product to the reactant graph, which is the closest existing approach to diffusion-based graph-to-graph translation. The paper however does not discuss equivariance and symmetry-breaking as a key design component. In fact, the authors use a method similar to our input alignment method as discussed in Sec. 7, which leads us to believe that our theoretical work is also relevant in explaining the success of their model.

Equivariance and symmetry-breaking Equivariant neural networks have gained attention for their ability to incorporate known symmetries, often improving generalization . However, as noted by Smidt et al. (2021), these networks face a fundamental limitation when dealing with self-symmetric inputs, as they cannot break symmetries present in the data. This issue arises in various applications, including prediction tasks on symmetric domains and generative models that must reconstruct from highly symmetric latent spaces (Lawrence et al., 2024). Previous work has explored this problem in different contexts, such as graph representation learning (Srinivasan & Ribeiro, 2020), set generation (Zhang et al., 2022), and modeling physical systems (Kaba et al., 2023).

Permutation equivariance in graph diffusion Equivariance is widely used in graph diffusion models to parameterize the reverse process (Niu et al., 2020; Vignac et al., 2023; Hoogeboom et al., 2022b; Huang et al., 2022). One motivation is that, in diffusion models, permutation equivariant neural nets induce permutation invariant distributions (Niu et al., 2020), meaning that different permutations of the same graph have the same probabilities under the model. Recently, Yan et al. (2023) showed that relaxing permutation equivariance in graph diffusion using absolute positional encodings can improve performance empirically.

3 Preliminaries: Graph-to-Graph Translation

Consider a database of N_{obs} graphs $\mathcal{D} = \{(\mathbf{X}_n, \mathbf{Y}_n, \mathbf{P}_n^{\mathbf{Y}\to\mathbf{X}})\}_{n=1}^{N_{\text{obs}}}$, where $\mathbf{X}_n$ represent the target graph, $\mathbf{Y}_n$ the input graph, and $\mathbf{P}_n^{\mathbf{Y}\to\mathbf{X}}$ are matrices defining *node mappings* between the two

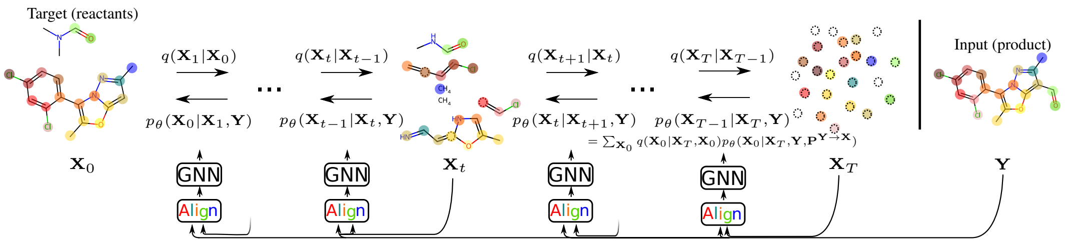

Target (reactants)
$q(\mathbf{X}_1|\mathbf{X}_0)$
$q(\mathbf{X}_t|\mathbf{X}_{t-1})$
$q(\mathbf{X}_{t+1}|\mathbf{X}_t)$
$q(\mathbf{X}_T|\mathbf{X}_{T-1})$
Input (product)
$p_\theta(\mathbf{X}_0|\mathbf{X}_1,\mathbf{Y})$
$p_\theta(\mathbf{X}_{t-1}|\mathbf{X}_t,\mathbf{Y})$
$p_\theta(\mathbf{X}_t|\mathbf{X}_{t+1},\mathbf{Y})$
$p_\theta(\mathbf{X}_{T-1}|\mathbf{X}_T,\mathbf{Y})$
$= \sum_{\mathbf{X}_0} q(\mathbf{X}_0|\mathbf{X}_T,\mathbf{X}_0)p_\theta(\mathbf{X}_0|\mathbf{X}_T,\mathbf{Y},\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$
$\mathbf{X}_0$
$\mathbf{X}_t$
$\mathbf{X}_T$
$\mathbf{Y}$
GNN
Align

Figure 3: An overview of a graph translation task using discrete diffusion models, illustrated through chemical reactions. We adopt *absorbing state diffusion* (Austin et al., 2021), such that samples from the stationary distribution are made of nodes and edges with the 'none' label. The condition and sample graphs are aligned using node mapping information (highlighted with the matching colours) which identifies pairs of matched nodes across the input and target graphs.

graphs. The *graph translation* task is: given that the data is sampled from an unknown distribution $p(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$, predict valid targets $\mathbf{X} \sim p(\mathbf{X} \mid \mathbf{Y})$ for a given input $\mathbf{Y}$.

We start by formally defining the graph objects $\mathbf{X}$ and $\mathbf{Y}$. To do so, we consider the space of one-hot vectors of dimension K as vert(Δ^{K-1}), where vert(.) denotes the vertices of the probability simplex Δ^{K-1} with $K-1$ degrees of freedom. We then define the target $\mathbf{X}$ as a tuple $(\mathbf{X}^{\mathcal{N}}, \mathbf{X}^{E})$ of a node feature matrix $\mathbf{X}^{\mathcal{N}} \in (\text{vert}(\Delta^{K_a-1}))^{N_X}$ and an edge feature matrix $\mathbf{X}^{E} \in (\text{vert}(\Delta^{K_b-1}))^{N_X \times N_X}$, s.t. K_a and K_b are node and edge feature dimensions respectively. The input graph $\mathbf{Y}$ is similarly defined as $(\mathbf{Y}^{\mathcal{N}}, \mathbf{Y}^{E})$ with the same node and edge features, but potentially a different number of nodes N_Y. The features consist of node labels (with K_a possible values) and edge labels (with K_b possible values), each including an empty value $\perp$ to represent missing nodes or edges.

Node mapping matrices are associations for node pairs between $\mathbf{X}$ and $\mathbf{Y}$. Formally, we define such matrices as: $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}} \in \{0,1\}^{N_X \times N_Y}$, with $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}_{i,j} = 1$ if the i^{th} node of the target corresponds with the j^{th} node of the input, and zero otherwise. The purpose is to give additional information about how nodes in the target are reconfigured from the input. Thus, $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{\mathcal{N}}$ equals $\mathbf{X}^{\mathcal{N}}$ with the non-mapped nodes zeroed out. Correspondingly, $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{E}(\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\top}$ equals $\mathbf{X}^{E}$ with edges to non-mapped nodes zeroed out. We expect an inductive bias such that if there is an edge between two mapped nodes in $\mathbf{X}$, there is a high probability of an edge between the corresponding nodes in $\mathbf{Y}$ as well, i.e., the graphs are structurally similar for the parts that are node-mapped.

4 Background

Discrete diffusion models for conditional graph generation We follow the framework of Vignac et al. (2023), which adapts discrete diffusion models (Austin et al., 2021) to graphs. We present next the main constituents of the framework.

We assume a Markov *forward* process

$$q(\mathbf{X}_{t+1} \mid \mathbf{X}_t) = \prod_{i=1}^{N_X} q(\mathbf{X}^{\mathcal{N},i}_{t+1} \mid \mathbf{X}^{\mathcal{N},i}_t) \prod_{i,j=1}^{N_X} q(\mathbf{X}^{E,ij}_{t+1} \mid \mathbf{X}^{E,ij}_t), \tag{1}$$

to diffuse the reactant to noise, and a reverse process

$$p_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}) = \prod_{i=1}^{N_X} p_\theta(\mathbf{X}^{\mathcal{N},i}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}) \prod_{i,j}^{N_X} p_\theta(\mathbf{X}^{E,ij}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}), \tag{2}$$

defining our generative model. Here, θ represents neural network parameters. Note that we always have time conditioning implicitly $p_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}, t)$, but we drop t for notational convenience. We will also condition on the node mapping $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$ in Sec. 7, but we will not include it in the notation until then. The full generative distribution is $p_\theta(\mathbf{X}_{0:T} \mid \mathbf{Y}) = p(\mathbf{X}_T) \prod_{t=1}^{T} p_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y})$, where $p(\mathbf{X}_T)$ is a predefined prior such that $p(\mathbf{X}_T) = q(\mathbf{X}_T \mid \mathbf{X}_0)$. Following Hoogeboom et al. (2021); Austin et al. (2021), we use the neural network specifically to predict ground truth labels from noised samples, meaning that the neural network outputs a distribution $\tilde{p}_\theta\big(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y}\big)$.

The reverse process is then parameterized by

$$p_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}) = \sum_{\mathbf{X}_0} q\big(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{X}_0\big)\tilde{p}_\theta\big(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y}\big). \tag{3}$$

Algorithm 1 Loss calculation

```
Input: condition $\mathbf{Y}$, target $\mathbf{X}_0$, and optional permutation matrix $\mathbf{P}^{\mathbf{X}\to\mathbf{Y}}$ for alignment
$t \sim \text{Uniform}(\{0,\dots,T\})$
$\mathbf{X}_t \sim q(\mathbf{X}_t \,|\, \mathbf{X}_0)$
$\tilde{\mathbf{X}}_0 = D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$
Return Cross-Entropy($\mathbf{X}_0, \tilde{\mathbf{X}}_0$)
```

Algorithm 2 Sampling

```
Input: condition $\mathbf{Y}$
Choose (for alignment): $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}} \in \mathbb{R}^{N_\mathbf{X}\times N_\mathbf{Y}}$
$\mathbf{X}_T \propto p(\mathbf{X}_T)$
for $t = T$ to 1 do
    $\tilde{\mathbf{X}}_0 = D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$
    $\mathbf{X}^i_{t-1} \sim \sum_k q(\mathbf{X}^i_{t-1} \,|\, \mathbf{X}^i_t, \mathbf{X}^i_0)\tilde{\mathbf{X}}^i_0$
Return $\mathbf{X}_0$
```

We write the connection to Eq. (2) explicitly in App. B.Throughout the paper, we denote the direct output of the neural network as $D_\theta(\mathbf{X}_t, \mathbf{Y}) = (D_\theta(\mathbf{X}_t, \mathbf{Y})^\mathcal{N}, D_\theta(\mathbf{X}_t, \mathbf{Y})^E)$, where $D_\theta(\mathbf{X}_t, \mathbf{Y})^\mathcal{N} \in (\Delta^{K_a-1})^{N_X}$ and $D_\theta(\mathbf{X}_t, \mathbf{Y})^E \in (\Delta^{K_b-1})^{(N_X\times N_X)}$, i.e., we have a probability vector for each node and edge. This implies $\tilde{p}_\theta\big(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y}\big)$ is a distribution factorized over nodes and edges.

The single-step transition for nodes (resp. for edges) is defined with a transition matrix $\mathbf{Q}^\mathcal{N}_t$ as

$$q(\mathbf{X}^{\mathcal{N},i}_t \,|\, \mathbf{X}^{\mathcal{N},i}_{t-1}) = \text{Cat}(\mathbf{X}^{\mathcal{N},i}_t; \mathbf{p} = \mathbf{X}^{\mathcal{N},i}_{t-1}\mathbf{Q}^\mathcal{N}_t). \tag{4}$$

In our experiments, we use the absorbing-state formulation from Austin et al. (2021), where nodes and edges gradually transfer to the *absorbing state*, defined as the empty state $\perp$. Formally, $\mathbf{Q}_t = (1-\beta_t)I + \beta_t \mathbb{1}e_\perp^\top$, where β_t defines the *diffusion schedule* and $e_\perp$ is one-hot on the absorbing state. Then, the marginal $q(\mathbf{X}_t \,|\, \mathbf{X}_0)$ and conditional posterior $q(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{X}_0)$ also have a closed form for the absorbing state transitions. The prior $p(\mathbf{X}_T)$ is correspondingly chosen to be a delta distribution at a graph with no edges and nodes set to the $\perp$ state. The noise schedule β_t is defined using the mutual information criterion proposed in Austin et al. (2021). While other transitions, like uniform and marginal (Vignac et al., 2023) are equally possible, the absorbing state model is somewhat simpler and has been empirically found to outperform other transitions (Austin et al., 2021; Lou et al., 2024), motivating our choice.

We use the cross-entropy loss, as discussed in Austin et al. (2021) and Vignac et al. (2023):

$$-\mathbb{E}_{q(\mathbf{X}_0,\mathbf{Y})q(t)q(\mathbf{X}_t \,|\, \mathbf{X}_0)}[\log \tilde{p}_\theta(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y})], \tag{5}$$

where $q(t)$ is a uniform distribution over $t \in \{1 \dots T\}$.

Fig. 3 provides an overview of the conditional graph diffusion framework we use. For completeness, a comprehensive definition of the model's components for nodes and edges is given in App. B. The training and sampling procedures with graph diffusion models are presented in Alg. 1 and Alg. 2, along with optional conditioning on $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$, as described in Sec. 7.

Permutation Equivariance and Invariance Permutation equivariance for the denoiser is defined as $D_\theta(\mathbf{P}\mathbf{X}) = \mathbf{P}D_\theta(\mathbf{X})$ where $\mathbf{P}$ is an arbitrary permutation matrix, and $\mathbf{P}\mathbf{X} = (\mathbf{P}\mathbf{X}^\mathcal{N}, \mathbf{P}\mathbf{X}^E\mathbf{P}^\top)$. This makes single-step reverse transitions equivariant as well, in the sense that $p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t) = p_\theta(\mathbf{P}\mathbf{X}_{t-1} \,|\, \mathbf{P}\mathbf{X}_t)$. For our conditional setting, permutation equivariance can be written as $D_\theta(\mathbf{P}\mathbf{X}, \mathbf{Y}) = \mathbf{P}D_\theta(\mathbf{X}, \mathbf{Y})$. The prior in graph diffusion models is also usually permutation invariant s.t. $p(\mathbf{P}\mathbf{X}_T) = p(\mathbf{X}_T)$ (Niu et al., 2020; Vignac et al., 2023; Hoogeboom et al., 2022b).

5 Permutation equivariant denoisers cannot learn the identity function

In this section, we consider a data set $\mathcal{D} = \{(\mathbf{X}_n, \mathbf{Y}_n, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}_n)\}^{N_\text{obs}}_{n=1}$, dubbed the 'identity data', where for all data points $\mathbf{X}_n = \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}_n\mathbf{Y}_n$. In other words, both the input and the target are equivalent, up to some permutation, as defined by the node-mapping matrix $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}_n$. While this may seem like an overly simplistic scenario, we can use it to reason about the expressiveness of our denoiser neural network. The motivation is that graph translation can be seen as close to the task of copying graphs (i.e., learning the identity function) due to the structural similarity between the input and target graphs. In this section, we show that standard graph diffusion models are highly ineffective at copying graphs.

First, we note that a denoiser implementing the identity function should always give the following output: $D_\theta(\mathbf{X}_t, \mathbf{Y}) = \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$, i.e., $D_\theta(\mathbf{X}_t, \mathbf{Y})$ should put all the probability mass on the correct node/edge label up to a permutation of $\mathbf{Y}$, thus becoming a vertex of the probability simplex as well. We note here that we pass the conditioning graph $\mathbf{Y}$ by concatenating it to the input $\mathbf{X}$ along the node dimension, resulting in a graph with $N_\mathbf{X} + N_\mathbf{Y}$ nodes and $(N_\mathbf{X} + N_\mathbf{Y}) \times (N_\mathbf{X} + N_\mathbf{Y})$ edges. No edges are added between the $\mathbf{X}$ and $\mathbf{Y}$ subgraphs in the input. However, we show that permutation equivariant denoisers D_θ are constrained to output 'mean' solutions instead, such that the outputs for all nodes and all edges are equal in the early stages of generation.

To illustrate this observation, we give an intuitive argument, and then proceed to a formal proof. Let us consider the case of a highly noisy input at $t = T$, and a permutation equivariant denoiser. Permutation equivariance means $D_\theta(\mathbf{R}\mathbf{X}_T, \mathbf{Y}) = \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$ for a random permutation matrix $\mathbf{R} \in \{0,1\}^{N_Y \to N_X}$. Since $\mathbf{X}_T$ does not contain any information about $\mathbf{Y}$, the model learns to ignore the $\mathbf{X}$ input, leading to $D_\theta(\mathbf{R}\mathbf{X}_T, \mathbf{Y}) = D_\theta(\mathbf{X}_T, \mathbf{Y}) = \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$. Thus, $\mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y} = \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$, which is a contradiction. The only way to make both properties hold, i.e., the equivariance $D_\theta(\mathbf{R}\mathbf{X}_T, \mathbf{Y}) = \mathbf{R}D_\theta(\mathbf{X}_T, \mathbf{Y})$ and the property of ignoring the input $D_\theta(\mathbf{R}\mathbf{X}_T, \mathbf{Y}) = D_\theta(\mathbf{X}_T, \mathbf{Y})$, is if $D_\theta(\mathbf{X}_T, \mathbf{Y})$ has the exact same output probability vector for each node and edge. The following theorem characterizes this rigorously.

Theorem 1. *(The optimal permutation equivariant denoiser) Let $D_\theta(\mathbf{X}_T, \mathbf{Y})$ be permutation equivariant s.t. $D_\theta(\mathbf{P}\mathbf{X}_T, \mathbf{Y}) = \mathbf{P}D_\theta(\mathbf{X}_T, \mathbf{Y})$, and let $q(\mathbf{X}_T)$ be permutation invariant. The optimal solution with respect to the cross-entropy loss with the identity data is, for all nodes i and j*

$$\begin{cases} D_\theta(\mathbf{X}_T, \mathbf{Y})^{\mathcal{N}}_{i,:} = \hat{\mathbf{y}}^{\mathcal{N}}, \hat{\mathbf{y}}^{\mathcal{N}}_k = \sum_i \mathbf{Y}_{i,k} / \sum_{i,k} \mathbf{Y}_{i,k}, \\ D_\theta(\mathbf{X}_T, \mathbf{Y})^{E}_{i,j,:} = \hat{\mathbf{y}}^{E}, \hat{\mathbf{y}}^{E}_k = \sum_{i,j} \mathbf{Y}_{i,j,k} / \sum_{i,j,k} \mathbf{Y}_{i,j,k}, \end{cases} \tag{6}$$

where $\hat{\mathbf{y}}^{\mathcal{N}}_k$ and $\hat{\mathbf{y}}^{E}_k$ are the marginal distributions of node and edge values in $\mathbf{Y}$.

The proof is given in App. A.2, with an extension to other time steps in App. A.7. the Theorem is also illustrated in Fig. 4. This makes it impossible for the model to solve the task in a single step with $T = 1$, clearly limiting the performance on this basic task. With enough steps, it is still possible, albeit slow, because smaller steps of the discrete generative process can break symmetries in $\mathbf{X}_t$, enabling the modeling of correlations between node and edge representations. In particular, in the case of absorbing diffusion and $T \to \infty$, the model becomes equivalent to an any-order autoregressive model (Hoogeboom et al., 2022a).

6 Solution: Aligned Permutation Equivariance

Given the failure of permutation equivariant denoisers to effectively copy graphs, we propose to relax the permutation equivariance assumption just enough to give the network the capacity to easily copy the condition $\mathbf{Y}$ to the denoiser output. The key observation we make is that it is enough to have permutation equivariance in the sense that *if we permute $\mathbf{X}$ and/or $\mathbf{Y}$, and accordingly permute the node mapping $P^{\mathbf{Y}\to\mathbf{X}}$ such that the matching between $\mathbf{Y}$ and $\mathbf{X}$ remains, the model output should be the same*. We call this *aligned permutation equivariance*. Formally, we use the node-mapping permutation matrix $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$ as an input to the denoiser and consider denoisers that satisfy the following constraint: $D_\theta(\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \mathbf{R}D_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$, where $\mathbf{R}$ and $\mathbf{Q}$ are permutation matrices of shapes $(N_\mathbf{X} \times N_\mathbf{X})$ and $(N_\mathbf{Y} \times N_\mathbf{Y})$, respectively.

With $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$ and $\mathbf{Y}$ as input, an unconstrained denoiser can output the ground-truth permutation $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$ for the identity data task. Restricting the function to the aligned equivariant class does not clash with this, as can be seen by writing out the optimal solution and the equivariance condition for a permuted input $\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top$:

$$\text{(Identity data)} \quad D_\theta(\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top\mathbf{Q}\mathbf{Y} = \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}, \tag{7}$$

$$\text{(Sec. 6)} \quad D_\theta(\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \mathbf{R}D_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) = \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}. \tag{8}$$

Note that the permutation invariance of $p_\theta(\mathbf{X}_0)$ does not apply as it does for fully permutation equivariant models. However, we show that a generalized form of distribution invariance holds.

Theorem 2. *(Aligned denoisers induce aligned permutation invariant distributions) If the denoiser function D_θ has the aligned equivariance property and the prior $p(\mathbf{X}_T)$ is permutation invariant, then the generative distribution $p_\theta(\mathbf{X}_0 \mid \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$ has the corresponding property for any permutation matrices $\mathbf{R}$ and $\mathbf{Q}$:*

$$p_\theta(\mathbf{R}\mathbf{X}_0 \mid \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = p_\theta(\mathbf{X}_0 \mid \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}). \tag{9}$$

A proof is given in App. A.4, while Figure 4 showcases the denoising output of an aligned equivariant model with a given $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$. Informally, the theorem states that the graph pairing has the same probability for all isomorphisms of the input and target graphs, as long as the node mapping is not reassigned to different nodes. Thus, during training, we can use the permutations present in the data, and be sure that the model generalizes to other permutations. During sampling, we only have access to $\mathbf{Y}$ without node mapping information. The theorem ensures we can assign node mappings arbitrarily and still obtain effectively the same distribution over the targets. To see this, we note that all node mapping matrices can be characterized by a base permutation matrix $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$ left multiplied by different permutations $\mathbf{R}$. Applying the theorem: $p_\theta(\mathbf{X}_0 \,|\, \mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) = p_\theta(\mathbf{R}^{-1}\mathbf{X}_0 \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$, where $\mathbf{R}^{-1}$ is just another permutation matrix, shows that the distribution equals $p_\theta(\mathbf{X}_0 \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$, up to some permutation. In fact, by sampling the node mapping matrix randomly, the distribution $p_\theta(\mathbf{X}_0 \,|\, \mathbf{Y}) = \sum_{\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}} p_\theta(\mathbf{X}_0 \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) p(\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$ becomes invariant to any permutation of $\mathbf{X}_0$ and $\mathbf{Y}$, although this is not strictly necessary in our context. Lastly, we note that our use of node mapping here is not a constraint: it is merely an additional input which allows the denoiser to use the structure of the conditional graph efficiently, and as such we benefit even from a partially correct node mapping.

7 Methods for Alignment in GNNs

We have established how constraining permutation equivariance to *aligned* permutation equivariance is a key component for the success of the denoising model. In this section, we discuss multiple methods to induce an alignment between the input and target graphs. We present the implementation details in this section, and prove for each method that they belong to the class of aligned equivariant models in App. A.5. We visualize the methods in Fig. A9.

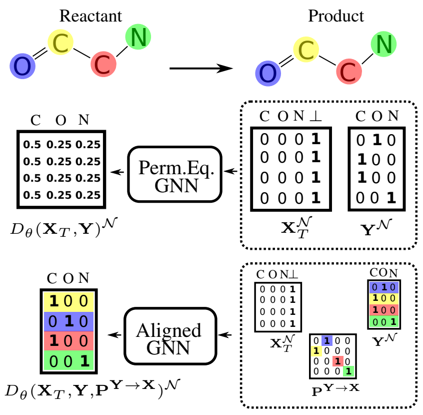

Reactant
Product
O
C
C
N
Perm.Eq. GNN
Aligned GNN
$D_\theta(\mathbf{X}_T,\mathbf{Y})^{\mathcal{N}}$
$\mathbf{X}_T^{\mathcal{N}}$
$\mathbf{Y}^{\mathcal{N}}$
$D_\theta(\mathbf{X}_T,\mathbf{Y},\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\mathcal{N}}$
$\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$

Figure 4: Comparing the optimal permutation equivariant denoiser to an aligned denoiser. As per Theorem 1, the permutation equivariant model (*top*) outputs the marginal distribution over node types whereas the aligned model (*bottom*) reconstructs the correct reactant nodes in a specific permutation $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$.

Node-mapped positional encodings Graph positional encodings are a standard tool to break permutation symmetry in GNNs, making them a natural candidate for inducing aligned permutation equivariance. We consider uniquely identifying pairs of nodes matched via the node-mapping matrix in both $\mathbf{X}_t$ and $\mathbf{Y}$ by adding a positional encoding vector to each unique node pair. In practice, we generate a set of distinct vectors $\boldsymbol{\varphi} \in \mathbf{R}^{N_Y \times d_\varphi}$ s.t. $\boldsymbol{\varphi} = g(\mathbf{Y})$ for each of the input graph nodes $\mathbf{Y}$, with g a function generating the encodings based on $\mathbf{Y}$. In this work, we take ϕ as a standard graph Laplacian positional encodings (Dwivedi et al., 2023), but in principle any positional encoding scheme is applicable. We then map the vectors ϕ to the corresponding inputs for the noisy graph nodes $\mathbf{X}_t$. Formally, we get $D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) = f_\theta([\mathbf{X}_t^{\mathcal{N}} \;\; \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\boldsymbol{\varphi}], \mathbf{X}_t^E, [\mathbf{Y}^{\mathcal{N}} \;\; \boldsymbol{\varphi}], \mathbf{Y}^E)$, where f_θ is the neural network that takes as inputs the augmented node features and regular edge features and $[\mathbf{X}_t^{\mathcal{N}} \;\; \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\boldsymbol{\varphi}] \in \mathbf{R}^{N_X \times (K_a + d_\varphi)}$ corresponds to concatenation along the feature dimension. The only change required for the neural network is to increase the initial linear layer size for the node inputs.

Direct skip connection Next, motivated by the identity function analysis in Sec. 5, we propose an alignment method that solves the identity task in a minimal way. In particular, we modify the network to include a direct connection from the condition to the target output: $D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) = \mathrm{softmax}(f_\theta^{\mathrm{logit}}(\mathbf{X}_t, \mathbf{Y}) + \lambda \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y})$ where $f_\theta^{\mathrm{logit}}(\mathbf{X}_t, \mathbf{Y})$ are the logits at the last layer of the neural network for the nodes and edges, λ is a learnable parameter and $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y} = (\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{\mathcal{N}}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^E(\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^\top)$. The sum is possible because $\mathbf{Y}$ is in one-hot format and the dimensionalities of the denoiser output and $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$ are the same. It is easy to see that when $\lambda \to \infty$, $D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) \to \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$ because the direct connection from $\mathbf{Y}$ dominates in the softmax.

Aligning $\mathbf{Y}$ in the input As a natural next step from aligning the graphs at the output via skip connections, we propose to align them in the input. This allows more expressivity from the network, since it can process the aligned graphs in the layers of the network before outputting the denoising solution. In practice, we align $\mathbf{Y}$ by concatenating $\mathbf{X}$ and $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}$ along the feature dimension before passing it to the neural network. This also presents an opportunity for computational savings: Since $\mathbf{Y}$ is already concatenated to $\mathbf{X}$ along the feature dimension, $\mathbf{Y}$ does not strictly need to be processed as a separate graph anymore. This is useful, e.g.,with the graph transformer model, since computation scales quadratically with the total amount of nodes, which is reduced from $(N_X + N_Y)$ to N_X. Thus, we can drop out the explicit $\mathbf{Y}$ graph entirely, and only process the $\mathbf{X}$ graph augmented with $\mathbf{Y}$: $D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) = f_\theta([\mathbf{X}_t^{\mathcal{N}} \quad \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{\mathcal{N}}], [\mathbf{X}_t^{E} \quad \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{E}(\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^\top])$, where $[\cdot]$ means concatenation along the feature dimension.

8 Adding Post-training Conditioning for Discrete Diffusion Models

To fully realize the potential of diffusion models for graph-to-graph translation, we derive a novel reconstruction guidance-like (Ho et al., 2022; Chung et al., 2023; Song et al., 2023a) method for discrete models. Note that this approach differs from Vignac et al. (2023) who use classifier-guidance in their conditional generation. We use the denoiser output to evaluate the likelihood $p(y \mid \mathbf{X}_0)$ of the condition y (e.g., probability of synthesisability) given the reactants $\mathbf{X}_0$, and backpropagate to get an adjustment of the backward update step:

$$\log \mathbf{P}_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}, y) \propto \nabla_{\mathbf{X}_t} \log(\mathbb{E}_{p_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})} p(y \mid \mathbf{X}_0))\mathbf{X}_{t-1} + \log \mathbf{P}_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}). \tag{10}$$

The derivation starts with similar steps as the classifier guidance method in (Vignac et al., 2023).More details in App. F. The method can be implemented with a few lines of code and works as long as $p(y \mid \mathbf{X}_0)$ is differentiable. For the retrosynthesis application, we experiment with a toy model of synthesisability that is based on the count of atoms in the reaction (Ertl & Schuffenhauer, 2009), a concept also known as atom economy (Trost, 1991).

9 Experiments

We assess the effect of aligned denoisers on the performance of regular discrete diffusion models. To this effect, we first demonstrate our model on a toy example: copying simple graphs. We then evaluate our method more rigorously on the real-world task of retrosynthesis, the task of defining precursors for a given compound. We also show how diffusion enables a number of downstream tasks within retrosynthesis, including inpainting and property-guided generation.

9.1 Copying graphs

We evaluate the model's ability to copy a graph as it is. To do so, we define a simple dataset made of 5×5 grids. We define an aligned and an equivariant denoiser, both with the same architecture: a 2-layer GNN with 16 nodes in the hidden dimension. We train both models on the same 100 samples with a batch size of 32 for 10 epochs. Figure 5 shows that our aligned denoiser recovers the original graph almost perfectly, while a fully equivariant denoiser outputs only a fraction of the same components.

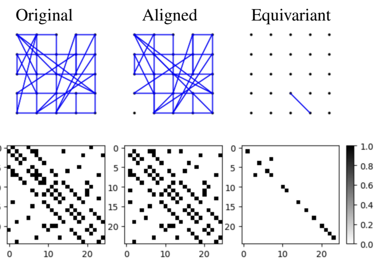


Figure 5: Showcasing performance on graph copying of grids.

9.2 Retrosynthesis

Retrosynthesis is a crucial step in the drug discovery pipeline, as it provides a concrete plan to create an identified compound. Single-step retrosynthesis in particular seeks to define the reactants in a single chemical reaction, which can then be chained to create a more comprehensive synthesis plan. The task lends itself naturally to graph translation models, since it seeks to generate a target graph (reactants) given an input graph (product). In this experiment, we compare the performance of the aligned denoiser to that of an equivariant denoiser using real-world chemical reactions.

Experimental setup We use the benchmark dataset USPTO-50k for our experiments. The dataset consists of 50000 chemical reactions, in SMILES format (Weininger, 1988), carefully curated by (Schneider et al., 2016) from an original 2 million reactions extracted through text mining by Lowe (2012). More information on the benchmark dataset USPTO and its various subsets can be found in App. C.2. We use the graph transformer architecture introduced by Dwivedi & Bresson (2021) and used by Vignac et al. (2023). The network architecture and hyperparameters are detailed in App. C.6. For each product, we generate 100 reactant sets, deduplicate them and rank the unique reactants from most-to-least likely, as judged by the model, using the likelihood lower bound and duplicate counts as a proxy. Based on the types of alignment discussed in Sec. 7, we experiment with *(1)* a permutation equivariant model (*Unaligned*), *(2)* a model augmented with atom-mapped positional encodings, calculated from the graph Laplacian eigendecomposition (*DiffAlign-PE*), *(3)* A model with Laplacian positional encodings as well as the skip connection (*DiffAlign-PE+skip*), *(4)* A model where $\mathbf{X}$ and $\mathbf{Y}$ are aligned at the input as well as in the output (*DiffAlign-input alignment*). We use $T = 100$. We trained the models for 400–600 epochs and chose the best checkpoint based on the Mean Reciprocal Rank (MRR, Liu et al. (2009)) score with $T = 10$ of the validation set. For more details, see App. C.6.

Evaluation We follow previous practices and use the accuracy of obtaining the ground-truth reactants as our main metric. This is measured by top-k accuracy, which counts the number of ground-truth matches in the deduplicated and ranked samples among the top-k generated reactions. We also report Mean Reciprocal Rank (MRR, Liu et al. (2009)) as used by Maziarz et al. (2023). We focus here on a comparison between aligned and equivariant denoisers. In order to place our models in the context of the available literature, we also compare to existing baselines in App. D.

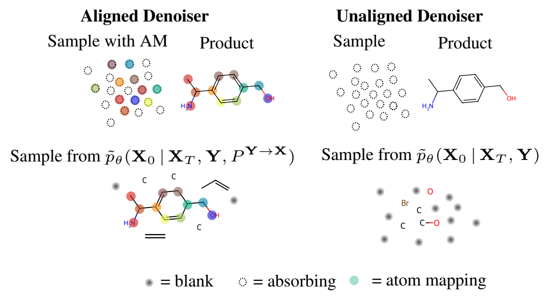
Results We visualize the aligned denoiser and the permutation equivariant denoiser in Fig. 6a. As predicted by Theorem 1, a sample from the permutation equivariant denoising distribution at high levels of noise $p_\theta(\mathbf{X}_0 \mid \mathbf{X}_T, \mathbf{Y})$ has no information about the structure of the product. In contrast, an aligned denoiser is able to copy the product structure, and the initial denoising output is of much higher quality. We report results for the different aligned models and the unaligned model for top-k

Table 1: Top-k accuracy and MRR on the USPTO-50k test data set. Aligned models outperform the unaligned one with the combined PE+skip model reaching the highest results. For a comparison to other retrosynthetic baselines see App. D.

Method	$\mathbf{k = 1} \uparrow$	$\mathbf{k = 3} \uparrow$	$\mathbf{k = 5} \uparrow$	$\mathbf{k = 10} \uparrow$	$\widehat{\textbf{MRR}} \uparrow$
Unaligned	4.1	6.5	7.8	9.8	0.056
DiffAlign-input	44.1	65.9	72.2	78.7	0.554
DiffAlign-PE	49.0	70.7	76.6	**81.8**	0.601
DiffAlign-PE+skip	**54.7**	**73.3**	**77.8**	81.1	**0.639**

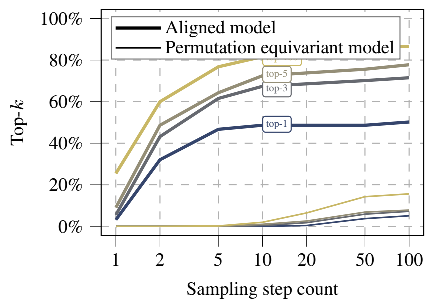
and the combined MRR score on the USPTO-50k test set in Table 1. The model without alignment performs worst, while combining the input alignment with positional encodings achieves the highest top-$k < 10$ and MRR scores across all models, making our diffusion-based model competitive with SOTA in retrosynthesis. Specifically, Table 2 shows our top-k accuracy and MRR score compared to other baselines. We also outperform template-based models in all top-k scores (see note on the evaluation of (Igashov et al., 2024) to understand why its results are not comparable), and outperform all non-pretrained models on top-1. While we use a relatively large value for T during training in our best models, we also highlight that the performance of the aligned model does not degrade significantly when reducing the count of sampling steps to a fraction of $T = 100$. See Fig. 6 for an ablation study with the number of sampling steps for our best model with positional encodings and skip connections. We also compare to a permutation equivariant model and show that the top-k scores go to near zero at 10 steps, while the aligned model sometimes recovers the ground truth even with a single denoising step.

In App. E, we provide further ablations for different transition matrices, a model using only the skip connections, and a model with matched Gaussian noise positional encodings. With the latter experiment, we show that the benefits brought by the positional encoding having the graph inductive bias due to the graph Laplacian eigenvector structure are not particularly significant, compared to the inductive bias of alignment brought by matching the positional encodings across sides.



Aligned Denoiser
Sample with AM
Product
Unaligned Denoiser
Sample
Product
Sample from $\tilde{p}_\theta(\mathbf{X}_0 \mid \mathbf{X}_T, \mathbf{Y}, P^{\mathbf{Y}\rightarrow\mathbf{X}})$
Sample from $\tilde{p}_\theta(\mathbf{X}_0 \mid \mathbf{X}_T, \mathbf{Y})$
● = blank
◌ = absorbing
● = atom mapping

(a) Aligned vs unaligned denoiser



(b) Quality of samples vs. sampling speed

Figure 6: Top-k scores for sampling step counts T for our PE-skip model and a model with a standard permutation equivariant denoiser, using the first 10% of validation set reactions.

Table 2: Top-k accuracy and MRR on the USPTO-50k test data set—extended comparison, including models with pretraining on larger data sets, and Retrobridge Igashov et al. (2024), where the evaluation is done with a relaxed metric that does not consider charges or stereochemistry.

	Method	$\mathbf{k = 1} \uparrow$	$\mathbf{k = 3} \uparrow$	$\mathbf{k = 5} \uparrow$	$\mathbf{k = 10} \uparrow$	$\widehat{\mathbf{MRR}} \uparrow$
Pre-trained	RSMILES (Zhong et al., 2022)	56.3	**79.2**	**86.2**	**91.0**	0.680
	PMSR (Jiang et al., 2023)	**62.0**	78.4	82.9	86.8	**0.704**
Temp.	Retrosym (Coley et al., 2017b)	37.3	54.7	63.3	74.1	0.480
	GLN (Dai et al., 2019)	52.5	74.7	81.2	87.9	0.641
	LocalRetro (Chen & Jung, 2021)	**52.6**	**76.0**	**84.4**	**90.6**	**0.650**
Synthon	GraphRetro (Somnath et al., 2021)	**53.7**	68.3	72.2	75.5	0.611
	RetroDiff (Wang et al., 2023)	52.6	**71.2**	**81.0**	83.3	**0.629**
	MEGAN (Sacha et al., 2021)	48.0	70.9	78.1	**85.4**	0.601
	G2G (Shi et al., 2020a)	48.9	67.6	72.5	75.5	0.582
Template-free	SCROP (Zheng et al., 2019)	43.7	60.0	65.2	68.7	0.521
	Tied Transformer (Kim et al., 2021)	47.1	67.1	73.1	76.3	0.572
	Aug. Transformer (Tetko et al., 2020)	48.3	-	73.4	77.4	0.569
	Retrobridge (*) (Igashov et al., 2024)	50.3	**74.0**	**80.3**	**85.1**	0.622
	GTA_aug (Seo et al., 2021)	51.1	67.6	74.8	81.6	0.605
	Graph2SMILES (Tu & Coley, 2022)	52.9	66.5	70.0	72.9	0.597
	Retroformer (Wan et al., 2022)	53.2	71.1	76.6	82.1	0.626
	DualTF_aug (Sun et al., 2021)	53.6	70.7	74.6	77.0	0.619
Ours	Unaligned	4.1	6.5	7.8	9.8	0.056
	DiffAlign-input	44.1	65.9	72.2	78.7	0.554
	DiffAlign-PE	49.0	70.7	76.6	81.8	0.601
	DiffAlign-PE+skip	**54.7**	73.3	77.8	81.1	**0.639**

9.3 Benefits of diffusion: guided generation and inpainting

We study the use of an external function for guided generation and inpainting, thus demonstrating the advantages of graph-to-graph diffusion in the field of retrosynthesis as a concrete application domain. In retrosynthesis, an interesting use-case for posthoc-conditioning is to increase the probability of the generated reactants being synthesisable, using some pre-trained synthesisability model. To showcase the idea, we use a toy synthesisability model based on the total count of atoms in the reactants (Ertl & Schuffenhauer, 2009). Figure 8a shows an example where we nudge the model towards precursors with lower atom count with details of the procedure given in Alg. 3. More detailed comments on this procedure are available in App. F. To illustrate the benefits of inpainting, we replicate finding a known synthesis pathway hypothetically, in particular, BHC's green synthesis of Ibuprofen (Cann & Connelly, 2000). Fig. 7 compares the output of our model to the ground truth synthesis. We write out the steps in detail in App. H.

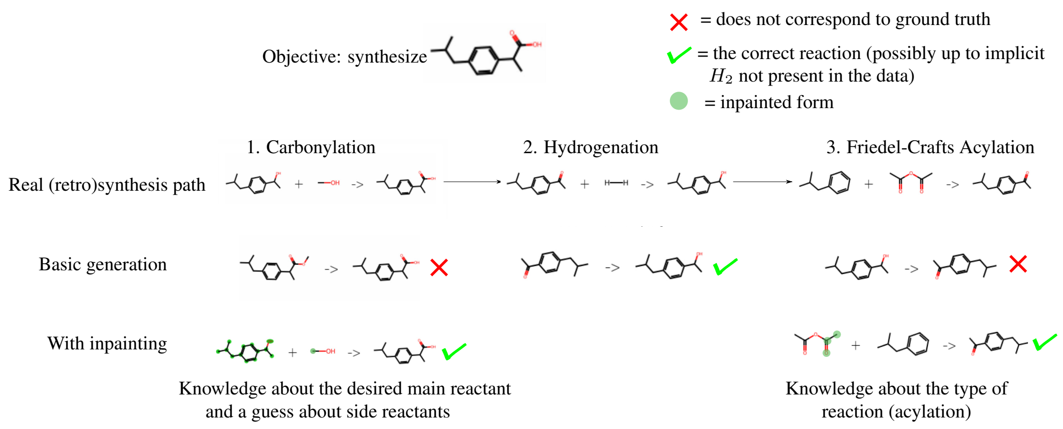

Objective: synthesize
✗ = does not correspond to ground truth
✓ = the correct reaction (possibly up to implicit H_2 not present in the data)
● = inpainted form
1. Carbonylation
2. Hydrogenation
3. Friedel-Crafts Acylation
Real (retro)synthesis path
Basic generation
With inpainting
Knowledge about the desired main reactant and a guess about side reactants
Knowledge about the type of reaction (acylation)

Figure 7: Replicating BHC's green synthesis of Ibuprofen using our model interactively, and comparing to the known synthesis path.

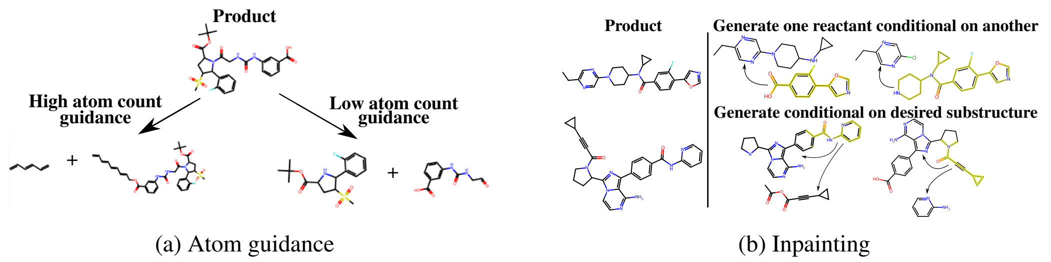


Figure 8: (a) Atom count guidance lets us specify if reactants should have many or few atoms, controlling the *atom economy*. (b) Examples of inpainting with our model. Parts highlighted in yellow are fixed by a practitioner to reflect desired characteristics, and the diffusion model completes the reaction.

10 Conclusion

In this work, we studied an important aspect of the design space of conditional graph diffusion models: the equivariance of the denoiser. We showed that a permutation equivariant model converges to a 'mean' distribution for all the components of the graph. We discuss how this result limits the expressivity of the denoising process, which ultimately impedes the performance of graph diffusion models. We propose aligned permutation equivariance to force the model to only consider permutations which maintain the intrinsic alignment between the conditioning and generated graphs. Our aligned denoisers achieve state-of-the-art results among template-free methods, reaching a top-1 accuracy beyond that of template-based methods. Furthermore, aligned permutation equivariance opens up the benefits of graph diffusion in retrosynthesis, including flexible post-training conditioning mechanisms, and the ability to adjust the amount of sampling steps during inference freely. These properties could be used, e.g., in interactive applications or to add more controls for multi-step retrosynthesis planners.

A limitation of the method is the requirement of some mapping information between the conditional and generated graphs, although not fully mapped graphs. The model can handle errors in mapping with additional denoising steps. Another limitation is the high computational demand characteristic of iterative denoising. Advances in accelerated diffusion sampling methods (Hoogeboom et al., 2022a; Karras et al., 2022; Lu et al., 2022; Song et al., 2023b; Sauer et al., 2023; Shih et al., 2023) are likely to improve this aspect.

REFERENCES

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021.

Benjamin Boys, Mark Girolami, Jakiw Pidstrigach, Sebastian Reich, Alan Mosca, and O Deniz Akyildiz. Tweedie moment projected diffusions for inverse problems. *arXiv preprint arXiv:2310.06721*, 2023.

M. C. Cann and M. E. Connelly. Synthesis of ibuprofen: A greener synthesis of ibuprofen which creates less waste and fewer byproducts. In *Real World Cases in Green Chemistry*. American Chemical Society, Washington, DC, 2000.

Shuan Chen and Yousung Jung. Deep retrosynthetic reaction prediction using local reactivity and global attention. *JACS Au*, 1(10):1612–1620, 2021.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

Connor W. Coley, Regina Barzilay, Tommi S. Jaakkola, William H. Green, and Klavs F. Jensen. Prediction of organic reaction outcomes using machine learning. *ACS Central Science*, 3(5): 434–443, 2017a.

Connor W. Coley, Luke Rogers, William H. Green, and Klavs F. Jensen. Computer-assisted retrosynthesis based on molecular similarity. *ACS Central Science*, 3(12):1237–1245, 2017b.

Hanjun Dai, Chengtao Li, Connor Coley, Bo Dai, and Le Song. Retrosynthesis prediction with conditional graph logic network. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8872–8882. Curran Associates, Inc., 2019.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Zehao Dou and Yang Song. Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *International Conference on Learning Representations*, 2024.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. In *AAAI Workshop on Deep Learning on Graphs*, 2021.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of Cheminformatics*, 1:1–11, 2009.

Marc Anton Finzi, Anudhyan Boral, Andrew Gordon Wilson, Fei Sha, and Leonardo Zepeda-Nunez. User-defined event sampling and uncertainty quantification in diffusion models for physical dynamical systems. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10136–10152. PMLR, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12454–12465. Curran Associates, Inc., 2021.

Emiel Hoogeboom, Alexey A. Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. In *International Conference on Learning Representations*, 2022a.

Emiel Hoogeboom, Víctor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3D. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 8867–8887. PMLR, 2022b.

H. Huang, L. Sun, B. Du, Y. Fu, and W. Lv. Graphgdp: Generative diffusion processes for permutation invariant graph generation. In *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 201–210. IEEE Computer Society, 2022.

Ilia Igashov, Arne Schneuing, Marwin Segler, Michael M Bronstein, and Bruno Correia. RetroBridge: Modeling retrosynthesis with Markov bridges. In *Conference on Learning Representations*, 2024.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2), 2022. doi: 10.1109/tnnls.2021.3070843.

Yinjie Jiang, WEI Ying, Fei Wu, Zhengxing Huang, Kun Kuang, and Zhihua Wang. Learning chemical rules of retrosynthesis with pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5113–5121, 2023.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*.

Wengong Jin, Connor Coley, Regina Barzilay, and Tommi Jaakkola. Predicting organic reaction outcomes with Weisfeiler-Lehman network. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2607–2616. Curran Associates, Inc., 2017.

Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecule optimization. In *International Conference on Learning Representations*, 2019.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577. Curran Associates, Inc., 2022.

Eunji Kim, Dongseon Lee, Youngchun Kwon, Min Sik Park, and Youn-Suk Choi. Valid, plausible, and diverse retrosynthesis using tied two-way transformers with latent variables. *Journal of Chemical Information and Modeling*, 61(1):123–133, 2021.

Hannah Lawrence, Vasco Portilheiro, Yan Zhang, and Sékou-Oumar Kaba. Improving equivariant networks with probabilistic symmetry breaking. In *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024.

Zaiyun Lin, Shiqiu Yin, Lei Shi, Wenbiao Zhou, and Yingsheng John Zhang. G2gt: Retrosynthesis prediction with graph-to-graph attention neural network and self-training. *Journal of Chemical Information and Modeling*, 63:1894–1905, 2023.

Songtao Liu, Zhengkai Tu, Minkai Xu, Zuobai Zhang, Lu Lin, Rex Ying, Jian Tang, Peilin Zhao, and Dinghao Wu. FusionRetro: Molecule representation fusion via in-context learning for retrosynthetic planning. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 22028–22041. PMLR, 2023.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion language modeling by estimating the ratios of the data distribution. 2024.

Daniel Mark Lowe. *Extraction of Chemical Structures and Reactions from the Literature*. PhD thesis, University of Cambridge, 2012.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan LI, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5775–5787. Curran Associates, Inc., 2022.

Krzysztof Maziarz, Austin Tripp, Guoqing Liu, Megan Stanley, Shufang Xie, Piotr Gaiński, Philipp Seidl, and Marwin Segler. Re-evaluating retrosynthesis algorithms with syntheseus. In *NeurIPS 2023 AI for Science Workshop*, 2023.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4474–4484. PMLR, 2020.

Xinyu Peng, Ziyang Zheng, Wenrui Dai, Nuoqian Xiao, Chenglin Li, Junni Zou, and Hongkai Xiong. Improving diffusion models for inverse problems using optimal posterior covariance. *arXiv preprint arXiv:2402.02149*, 2024.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

Mikołaj Sacha, Mikołaj Błaz, Piotr Byrski, Paweł Dabrowski-Tumanski, Mikołaj Chrominski, Rafał Loska, Paweł Włodarczyk-Pruszynski, and Stanisław Jastrzebski. Molecule edit graph attention network: modeling chemical reactions as sequences of graph edits. *Journal of Chemical Information and Modeling*, 61(7):3273–3284, 2021.

Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023.

Nadine Schneider, Nikolaus Stiefl, and Gregory A Landrum. What's what: The (nearly) definitive guide to reaction role assignment. *Journal of Chemical Information and Modeling*, 56(12):2336–2346, 2016.

Philippe Schwaller, Theophile Gaudin, David Lanyi, Costas Bekas, and Teodoro Laino. "Found in Translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models. *Chemical Science*, 9(28):6091–6098, 2018.

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS central science*, 5(9):1572–1583, 2019a.

Philippe Schwaller, Teodoro Laino, Théophile Gaudin, Peter Bolgar, Christopher A Hunter, Costas Bekas, and Alpha A Lee. Molecular transformer: A model for uncertainty-calibrated chemical reaction prediction. *ACS Central Science*, 5(9):1572–1583, 2019b.

Marwin HS Segler and Mark P Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry–A European Journal*, 23(25):5966–5971, 2017.

Seung-Woo Seo, You Young Song, June Yong Yang, Seohui Bae, Hankook Lee, Jinwoo Shin, Sung Ju Hwang, and Eunho Yang. GTA: Graph truncated attention for retrosynthesis. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1):531–539, 2021.

Chence Shi, Minkai Xu, Hongyu Guo, Ming Zhang, and Jian Tang. A graph to graphs framework for retrosynthesis prediction. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8818–8827. PMLR, 2020a.

Chence Shi, Minkai Xu, Hongyu Guo, Ming Zhang, and Jian Tang. A graph to graphs framework for retrosynthesis prediction. In *International conference on machine learning*, pp. 8818–8827. PMLR, 2020b.

Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. In *Advances in Neural Information Processing Systems*, pp. 4263–4276. Curran Associates, Inc., 2023.

Tess E. Smidt, Mario Geiger, and Benjamin Kurt Miller. Finding symmetry breaking order parameters with euclidean neural networks. *Phys. Rev. Res.*, 3, 2021. doi: 10.1103/PhysRevResearch.3.L012002.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015.

Vignesh Ram Somnath, Charlotte Bunne, Connor Coley, Andreas Krause, and Regina Barzilay. Learning graph models for retrosynthesis prediction. In *Advances in Neural Information Processing Systems*, volume 34, pp. 9405–9415. Curran Associates, Inc., 2021.

Jiaming Song, Arash Vahdat, Morteza Mardani, and Jan Kautz. Pseudoinverse-guided diffusion models for inverse problems. In *International Conference on Learning Representations*, 2023a.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 2023b.

Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*, 2020.

Ruoxi Sun, Hanjun Dai, Li Li, Steven Kearnes, and Bo Dai. Towards understanding retrosynthesis by energy-based models. In *Advances in Neural Information Processing Systems*, volume 34, pp. 10186–10194. Curran Associates, Inc., 2021.

Igor V Tetko, Pavel Karpov, Ruud Van Deursen, and Guillaume Godin. State-of-the-art augmented NLP transformer models for direct and single-step retrosynthesis. *Nature Communications*, 11 (1):5575, 2020.

Barry M Trost. The atom economy—a search for synthetic efficiency. *Science*, 254(5037):1471–1477, 1991.

Zhengkai Tu and Connor W Coley. Permutation invariant graph-to-sequence model for template-free retrosynthesis and reaction prediction. *Journal of Chemical Information and Modeling*, 62 (15):3503–3513, 2022.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. DiGress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*, 2023.

Yue Wan, Chang-Yu Hsieh, Ben Liao, and Shengyu Zhang. Retroformer: Pushing the limits of end-to-end retrosynthesis transformer. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 22475–22490. PMLR, 2022.

Yiming Wang, Yuxuan Song, Minkai Xu, Rui Wang, Hao Zhou, and Weiying Ma. RetroDiff: Retrosynthesis as multi-stage distribution interpolation. *arXiv preprint arXiv:2311.14077*, 2023.

David Weininger. Smiles, a chemical language and information system. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

Jin-Wen Wu, Fei Yin, Yan-Ming Zhang, Xu-Yao Zhang, and Cheng-Lin Liu. Graph-to-graph: Towards accurate and interpretable online handwritten mathematical expression recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(4), 2021. doi: 10.1609/aaai.v35i4.16399.

Luhuan Wu, Brian Trippe, Christian Naesseth, David Blei, and John P Cunningham. Practical and asymptotically exact conditional sampling in diffusion models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 31372–31403. Curran Associates, Inc., 2023.

Shufang Xie, Rui Yan, Junliang Guo, Yingce Xia, Lijun Wu, and Tao Qin. Retrosynthesis prediction with local template retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 5330–5338, 2023.

YuQing Xie and Tess Smidt. Equivariant symmetry breaking sets, 2024.

Chaochao Yan, Qianggang Ding, Peilin Zhao, Shuangjia Zheng, JINYU YANG, Yang Yu, and Junzhou Huang. RetroXpert: Decompose retrosynthesis prediction like a chemist. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11248–11258. Curran Associates, Inc., 2020.

Qi Yan, Zhengyang Liang, Yang Song, Renjie Liao, and Lele Wang. SwinGNN: Rethinking permutation invariance in diffusion models for graph generation. *arXiv preprint arXiv:2307.01646*, 2023.

Yan Zhang, David W. Zhang, Simon Lacoste-Julien, Gertjan J. Burghouts, and Cees G. M. Snoek. Multiset-equivariant set prediction with approximate implicit differentiation, 2022.

Shuangjia Zheng, Jiahua Rao, Zhongyue Zhang, Jun Xu, and Yuedong Yang. Predicting retrosynthetic reactions using self-corrected transformer neural networks. *Journal of Chemical Information and Modeling*, 60(1):47–55, 2019.

Zipeng Zhong, Jie Song, Zunlei Feng, Tiantao Liu, Lingxiang Jia, Shaolun Yao, Min Wu, Tingjun Hou, and Mingli Song. Root-aligned SMILES: a tight representation for chemical reaction prediction. *Chemical Science*, 13(31):9023–9034, 2022.

APPENDICES

This appendix is organized as follows. App. A presents our theoretical results on aligned permutation equivariance and accompanying proofs. App. B provides additional details on the setup for conditional graph diffusion, including the transition matrices, noise schedule, and data encoding as graphs. App. C includes additional details to replicate our experimental setup. App. D provides detailed comparsion between our model and other retrosynthetic baselines. App. F develops a method to apply arbitrary post-training conditioning with discrete diffusion models, and presents case studies showcasing the usefulness of post-training conditional inference in applications relevant to retrosynthesis. This includes generating samples with desired properties and refining the generation interactively through inpainting.

A ALIGNED PERMUTATION EQUIVARIANCE **17**
A.1 VISUALIZING OUR ALIGNMENT METHODS 17
A.2 PROOF THAT PERMUTATION EQUIVARIANT DENOISERS DO NOT RECOVER THE IDENTITY DATA 17
A.3 PROOF THAT ALIGNED PERMUTATION EQUIVARIANT DENOISERS RECOVER THE IDENTITY DATA 19
A.4 PROOF OF THE GENERALIZED DISTRIBUTIONAL INVARIANCE WITH ALIGNED EQUIVARIANCE 20
A.5 PROOFS THAT OUR DENOISERS ARE ALIGNED PERMUTATION EQUIVARIANT . 21
A.6 A SINGLE-LAYER GRAPH TRANSFORMER WITH ORTHOGONAL ATOM-MAPPED POSITIONAL ENCODINGS IS ABLE TO IMPLEMENT THE IDENTITY DATA SOLUTION FOR NODES 22
A.7 EXTENDING THEOREM 1 TO SMALLER TIME STEPS 24

B DETAILS ON CONDITIONAL GRAPH DIFFUSION **25**

C EXPERIMENTAL SETUP **28**
C.1 COPYING TASK ON THE GRID DATASET 28
C.2 DATA: USPTO DATA SETS 28
C.3 NOTES ON OUR SAMPLING AND RANKING PROCEDURES 29
C.4 DETAILS ON STEREOCHEMISTRY 29
C.5 DETAILS OF THE EVALUATION PROCEDURE 29
C.6 NEURAL NETWORK ARCHITECTURE, HYPERPARAMETERS, AND COMPUTE RESOURCES 30

D COMPARISION TO RETROSYNTHETIC BASELINES **30**

E ADDITIONAL ABLATIONS **32**

F ADDING POST-TRAINING CONDITIONING TO DISCRETE DIFFUSION MODELS **34**

G HANDLING NOISY NODE MAPPINGS FOR THE TOY DATA **35**

H DETAILS FOR THE IBUPROFEN SYNTHESIS EXPERIMENT **36**

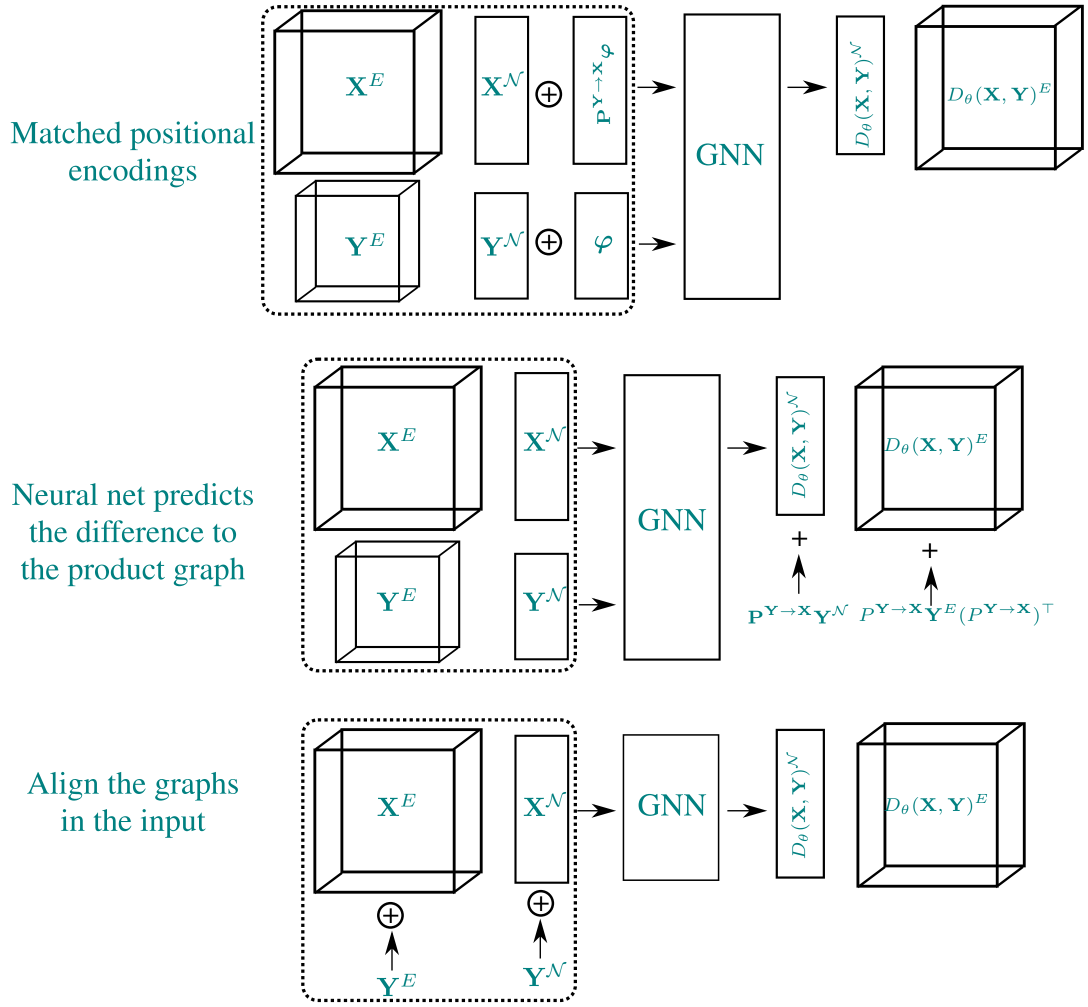

Matched positional encodings
$\mathbf{X}^E$
$\mathbf{X}^{\mathcal{N}}$ ⊕ $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\varphi$
$\mathbf{Y}^E$
$\mathbf{Y}^{\mathcal{N}}$ ⊕ φ
GNN
$D_\theta(\mathbf{X},\mathbf{Y})^{\mathcal{N}}$
$D_\theta(\mathbf{X},\mathbf{Y})^E$
Neural net predicts the difference to the product graph
$\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{\mathcal{N}}$
$P^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^E(P^{\mathbf{Y}\to\mathbf{X}})^\top$
Align the graphs in the input

Figure A9: Different ways to align the graphs in the architecture. All of them can be combined. The $\oplus$ sign means concatenation along the feature dimension, and $+$ is the standard addition. All of the methods can be combined together.

A Aligned Permutation Equivariance

A.1 Visualizing our Alignment Methods

We visualize the various alignment methods we use in this work in Fig. A9.

A.2 Proof that Permutation Equivariant Denoisers Do Not Recover the Identity Data

Definitions Let us consider a data set $\mathcal{D} = \{\mathbf{X}_n, \mathbf{Y}_n, \mathbf{P}_n^{\mathbf{Y}\to\mathbf{X}}\}_{n=1}^{N_{obs}}$, where for all data points, $\mathbf{X}_n = \mathbf{P}_n^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}_n$, that is, both sides of the reactions are equivalent, up to some permutation, as defined in the atom mapping matrix $\mathbf{P}_n^{\mathbf{Y}\to\mathbf{X}}$. It is always possible to preprocess the data such that the rows of $\mathbf{Y}_n$ are permuted with $\mathbf{Y}_n \leftarrow \mathbf{P}_n^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}_n$ so that the resulting atom mapping between $\mathbf{Y}_n$ and $\mathbf{X}_n$ is always identity. For simplicity, we assume such a preprocessed data set in this section.

Let us assume that the one-step denoiser probability, $p_\theta(\mathbf{X}_0 \mid \mathbf{X}_T, \mathbf{Y})$, is parameterized by the neural network $D_\theta(\mathbf{X}_T, \mathbf{Y}) \in \mathbb{R}^{N\times K}$ such that the probability factorises for the individual nodes and edges (so there is one output in the network for each node and each edge): $p_\theta(\mathbf{X}_0 \mid \mathbf{X}_T, \mathbf{Y}) = \prod_i \sum_k \mathbf{X}^N_{0,i,k} D_\theta(\mathbf{X}_T, \mathbf{Y})^{\mathcal{N}}_{i,k} \prod_{i,j} \sum_k \mathbf{X}^E_{0,i,j,k} D_\theta(\mathbf{X}_T, \mathbf{Y})^E_{i,j,k}$.

The ideal denoiser Clearly, the correct one-step denoiser $D_\theta(\mathbf{X}_T, \mathbf{Y}) = \mathbf{Y}$. This can be shown with the Bayes' rule by $q(\mathbf{X}_0 \,|\, \mathbf{X}_T, \mathbf{Y}) = \frac{q(\mathbf{X}_T \,|\, \mathbf{X}_0, \mathbf{Y}) q(\mathbf{X}_0 \,|\, \mathbf{Y})}{q(\mathbf{X}_T \,|\, \mathbf{Y})} = \frac{q(\mathbf{X}_T) q(\mathbf{X}_0 \,|\, \mathbf{Y})}{q(\mathbf{X}_T)} = q(\mathbf{X}_0 \,|\, \mathbf{Y})$. Because one $\mathbf{X}$ always matches with exactly one $\mathbf{Y}$ in the data, this is a delta distribution $q(\mathbf{X}_0 \,|\, \mathbf{Y}) = \prod_i \delta_{x_{0,i}, y_i}$, where we define $x_{0,i}$ and y_i as the value of the i:th node / edge. It is also easy to see that $D_\theta(\mathbf{X}_T, \mathbf{Y}) = \mathbf{Y}$ is the optimal solution for the cross-entropy loss:

$$- \sum_{(\mathbf{X}_0, \mathbf{Y})} q(\mathbf{X}_0) \log p_\theta(\mathbf{X}_0 \,|\, \mathbf{X}_T, \mathbf{Y}) \propto - \sum_{\mathbf{Y}} \log p_\theta(\mathbf{Y} \,|\, \mathbf{X}_T, \mathbf{Y})$$

$$= - \sum_{\mathbf{Y}} \log \left[\prod_i \sum_k \mathbf{Y}^N_{0,i,k} D_\theta(\mathbf{X}_T, \mathbf{Y})^{\mathcal{N}}_{i,k} \prod_{i,j} \sum_k \mathbf{Y}^E_{0,i,j,k} D_\theta(\mathbf{X}_T, \mathbf{Y})^E_{i,j,k} \right]. \quad (11)$$

All of the sums $\sum_k \mathbf{Y}^N_{0,i,k} D_\theta(\mathbf{X}_T, \mathbf{Y})^{\mathcal{N}}_{i,k}$ and $\sum_k \mathbf{Y}^E_{0,i,j,k} D_\theta(\mathbf{X}_T, \mathbf{Y})^E_{i,j,k}$ are maximized for each i, j and $\mathbf{Y}$ if $D_\theta(\mathbf{X}_T, \mathbf{Y}) = \mathbf{Y}$. In this case, the loss goes to zero.

The following theorem states that if the neural net is permutation equivariant, it will converge to a 'mean' solution, where the output for each node and each edge is the marginal distribution of nodes and edges in the conditioning product molecule, instead of the global optimum $D_\theta(\mathbf{X}_T, \mathbf{Y}) = \mathbf{Y}$.

Theorem 3. *The optimal permutation equivariant denoiser Let $D_\theta(\mathbf{X}_T, \mathbf{Y})$ be permutation equivariant s.t. $D_\theta(\mathbf{P}\mathbf{X}_T, \mathbf{Y}) = \mathbf{P} D_\theta(\mathbf{X}_T, \mathbf{Y})$, and let $q(\mathbf{X}_T)$ be permutation invariant. The optimal solution with respect to the cross-entropy loss with the identity data is, for all nodes i and j*

$$\begin{cases} D_\theta(\mathbf{X}_T, \mathbf{Y})^{\mathcal{N}}_{i,:} = \hat{\mathbf{y}}^{\mathcal{N}}, \hat{\mathbf{y}}^{\mathcal{N}}_k = \sum_i \mathbf{Y}_{i,k} / \sum_{i,k} \mathbf{Y}_{i,k}, \\ D_\theta(\mathbf{X}_T, \mathbf{Y})^E_{i,j,:} = \hat{\mathbf{y}}^E, \hat{\mathbf{y}}^E_k = \sum_{i,j} \mathbf{Y}_{i,j,k} / \sum_{i,j,k} \mathbf{Y}_{i,j,k}, \end{cases} \quad (12)$$

where $\hat{\mathbf{y}}^{\mathcal{N}}_k$ and $\hat{\mathbf{y}}^E_k$ are the marginal distributions of node and edge values in $\mathbf{Y}$.

Proof. **Nodes.** The cross-entropy denoising loss for the nodes can be written as

$$CE = - \sum_{(\mathbf{X}_0, \mathbf{Y})} \mathbb{E}_{q(\mathbf{X}_T \,|\, \mathbf{X}_0)} \sum_{i,k} \mathbf{X}^{\mathcal{N}}_{0,i,k} \log D_\theta(\mathbf{X}_T, \mathbf{Y})_{i,k} \quad (13)$$

$$= - \sum_{(\mathbf{X}_0, \mathbf{Y})} \mathbb{E}_{q(\mathbf{X}_T)} \sum_{i,k} \mathbf{X}^{\mathcal{N}}_{0,i,k} \log D_\theta(\mathbf{X}_T, \mathbf{Y})_{i,k} \quad (14)$$

$$= - \sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}_T)} \sum_{i,k} \mathbf{Y}^{\mathcal{N}}_{i,k} \log D_\theta(\mathbf{X}_T, \mathbf{Y})_{i,k}, \quad (15)$$

where the first equality is due to $q(\mathbf{X}_T \,|\, \mathbf{X}_0)$ containing no information about $\mathbf{X}_0$ at the end of the forward process, and the second equality is due to $\mathbf{X}_0 = \mathbf{Y}$ in the data. Since $q(\mathbf{X}_T)$ is permutation invariant, that is, all permuted versions $\mathbf{P}\mathbf{X}_T$ of $\mathbf{X}_T$ are equally probable, we can split the expectation into two parts $\mathbb{E}_{q(\mathbf{X}_T)}[\cdot] \propto \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{\mathbf{P}} [\cdot]$, where $\mathbf{X}'_T$ contain only graphs in distinct isomorphism classes, and $\sum_{\mathbf{P}}$ sums over all permutation matrices of size $N \times N$,

$$CE \propto - \sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{\mathbf{P}} \sum_{i,k} \mathbf{Y}^{\mathcal{N}}_{i,k} \log D_\theta(\mathbf{P}\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{i,k}. \quad (16)$$

Due to the permutation equivariance, $D_\theta(\mathbf{P}\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}} = \mathbf{P} D_\theta(\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}$, and $D_\theta(\mathbf{P}\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{i,k} = D_\theta(\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{\pi(i),k}$, where $\pi(i)$ denotes the index the index i is mapped to in the permutation $\mathbf{P}$. Thus,

$$CE \propto - \sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{\pi} \sum_{i,k} \mathbf{Y}^{\mathcal{N}}_{i,k} \log D_\theta(\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{\pi(i),k} \quad (17)$$

$$= - \sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{i,k} \sum_{\pi} \mathbf{Y}^{\mathcal{N}}_{\pi^{-1}(i),k} \log D_\theta(\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{i,k}, \quad (18)$$

where the equality is due to all permutations being in a symmetric position: What matters is the relative permutation between $\mathbf{Y}^{\mathcal{N}}$ and $D_\theta(\mathbf{X}'_T, \mathbf{Y})$. Now, $\sum_\pi \mathbf{Y}^{\mathcal{N}}_{\pi^{-1}(i),k} = \sum_\pi \mathbf{Y}^{\mathcal{N}}_{\pi(i),k} = C \sum_i \mathbf{Y}^{\mathcal{N}}_{i,k}$, because for each node index i, all the nodes in $\mathbf{Y}$ are included equally often due to symmetry. This is proportional to the marginal distribution $\hat{\mathbf{y}}^{\mathcal{N}}$ up to some constant, and thus we have:

$$CE \propto -\sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{i,k} \hat{\mathbf{y}}^{\mathcal{N}}_k \log D_\theta(\mathbf{X}'_T, \mathbf{Y})_{i,k}. \tag{19}$$

Clearly, the optimal value for each node output i is the empirical marginal distribution $D_\theta(\mathbf{X}'_T, \mathbf{Y})^N_{i,:} = (\hat{\mathbf{y}}^{\mathcal{N}})^\top$.

Edges With the exact same steps, we can get the equivalent of Eq. (16) for the edges:

$$CE \propto -\sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{P} \sum_{i,j,k} \mathbf{Y}^E_{i,j,k} \log D_\theta(P\mathbf{X}'_T, \mathbf{Y})^E_{i,j,k}. \tag{20}$$

The permutation equivariance property for the edges is now written as $D_\theta(\mathbf{P}\mathbf{X}'_T, \mathbf{Y})^E = \mathbf{P}D_\theta(\mathbf{X}'_T, \mathbf{Y})^E\mathbf{P}^\top$, and $D_\theta(\mathbf{P}\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{i,jk} = D_\theta(\mathbf{X}'_T, \mathbf{Y})^{\mathcal{N}}_{\pi(i),\pi(j),k}$. Thus,

$$CE \propto -\sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{\pi} \sum_{i,j,k} \mathbf{Y}^E_{i,j,k} \log D_\theta(\mathbf{X}'_T, \mathbf{Y})^E_{\pi(i),\pi(j),k} \tag{21}$$

$$= -\sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{i,j,k} \sum_{\pi} \mathbf{Y}^E_{\pi^{-1}(i),\pi^{-1}(j),k} \log D_\theta(\mathbf{X}'_T, \mathbf{Y})^E_{i,j,k}, \tag{22}$$

with the equality holding again due to symmetry. Now, for any pair of node indices i and j, the set of all permutations contains all pairs of node indices $(\pi(i), \pi(j))$ equally often due to symmetry. These pairs correspond to edges in $\mathbf{Y}^E$, and thus $\sum_\pi \mathbf{Y}^E_{\pi^{-1}(i),\pi^{-1}(j),k} = \sum_\pi \mathbf{Y}^E_{\pi(i),\pi(j),k} = D\sum_{i,j} \mathbf{Y}^E_{i,j,k}$, where D is a constant that counts how many times each edge pair appeared in the set of all permutations. This is again proportional to the marginal distribution over the edges $\hat{\mathbf{y}}^E$

$$CE \propto -\sum_{\mathbf{Y}} \mathbb{E}_{q(\mathbf{X}'_T)} \sum_{i,j,k} \hat{\mathbf{y}}^E_k \log D_\theta(\mathbf{X}'_T, \mathbf{Y})_{i,j,k}. \tag{23}$$

Again, the optimal value for each edge output (i, j) is $D_\theta(\mathbf{X}'_T, \mathbf{Y})^E_{i,j,:} = (\hat{\mathbf{y}}^E)^\top$.

□

A.3 Proof that Aligned Permutation Equivariant Denoisers Recover the Identity Data

Here, we detail the steps in the proof of Theorem 1, but with aligned permutation eqivariance intead of regular permutation equivariance, and highlight the principal differences in both calculations. We start with versions of Eq. (15) rewritten with atom mapping explicitly, and $P^{Y\to X}$ conditioning in the denoiser:

$$CE = -\sum_{(X_0, Y, P^{Y\to X})} \mathbb{E}_{q(X_T \mid X_0)} \sum_{i,k} X^N_{0,i,k} \log D_\theta(X_T, Y, P^{Y\to X})_{i,k} \tag{24}$$

$$= -\sum_{(Y, P^{Y\to X})} \mathbb{E}_{q(X_T)} \sum_{i,k} (P^{Y\to X} Y^N)_{i,k} \log D_\theta(X_T, Y, P^{Y\to X})_{i,k} \tag{25}$$

Versions of Eq. (16) to Eq. (18) with atom mapping conditioning and utilizing aligned permutation equivariance (instead of permutation equivariance):

$$CE \propto -\sum_{Y, P^{Y\to X}} \mathbb{E}_{q(X'_T)} \sum_{P} \sum_{i,k} (P^{Y\to X} Y^N)_{i,k} \log D_\theta(PX'_T, Y, P^{Y\to X})^N_{i,k} \tag{26}$$

$$= -\sum_{Y, P^{Y\to X}} \mathbb{E}_{q(X'_T)} \sum_{P} \sum_{i,k} (P^{Y\to X} Y^N)_{i,k} \log D_\theta(X'_T, Y, P^{-1}P^{Y\to X})^N_{\pi(i),k} \tag{27}$$

$$= -\sum_{Y, P^{Y\to X}} \mathbb{E}_{q(X'_T)} \sum_{i,k} \sum_{\pi} (P^{Y\to X} Y^N)_{\pi^{-1}(i),k} \log D_\theta(X'_T, Y, P^{-1}P^{Y\to X})^N_{i,k} \tag{28}$$

$$= -\sum_{Y, P^{Y\to X}} \mathbb{E}_{q(X'_T)} \sum_{i,k} \sum_{P} (P^{-1}P^{Y\to X} Y^N)_{i,k} \log D_\theta(X'_T, Y, P^{-1}P^{Y\to X})^N_{i,k} \tag{29}$$

We could assume again for notational simplicity (not necessary) that $P^{Y\to X} = I$ in the data. In any case, we have a cross-entropy loss where we try to predict a permutation of Y, $P^{-1}P^{Y\to X}Y$. Both $P^{-1}P^{Y\to X}$ and Y are given as explicit information to the neural network, so for each P, the optimal output is $P^{-1}P^{Y\to X}Y$, which is a permutation of the original Y.

The primary difference in these two proofs:

- With the version with regular permutation equivariance, only the data term Y is dependent on the permutation P in Eq. (18). This makes it possible to factor out the sum $\sum_\pi Y^N_{\pi^{-1}(i),k}$, leading to Eq. (19). This factorization is also why the optimal neural net output is not dependent on P.
- With aligned permutation equivariance, the denoiser term retains dependence on P^{-1}, and the maximal factorization is $\sum_P (P^{-1}Y^N)_{i,k} \log D_\theta(X'_T, Y, P^{-1})^N_{i,k}$. Thus, for each combination of P and Y, the neural net will have a different optimal output.

A.4 Proof of the Generalized Distributional Invariance with Aligned Equivariance

We start by proving a useful lemma, and then continue and continue to the proof of the main theorem.

Lemma 1. *(An aligned denoiser induces aligned distribution equivariance for a single reverse step)* *If the denoiser function D_θ has the aligned equivariance property $D_\theta(\mathbf{RX}, \mathbf{QY}, \mathbf{RP}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \mathbf{R}D_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$, then the conditional reverse distribution $p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$ has the property $p_\theta(\mathbf{RX}_{t-1} \,|\, \mathbf{RX}_t, \mathbf{QY}, \mathbf{RP}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$.*

Proof. First, let us denote the transition probabilities from t to $t-1$ with $F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$, where formally $F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\mathcal{N}}_{i,k} = \sum_{k'} q(\mathbf{X}^{\mathcal{N}}_{t-1,i,k} \,|\, \mathbf{X}^{\mathcal{N}}_{t,i,k}, \mathbf{X}^{\mathcal{N}}_{0,i,k}) D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\mathcal{N}}_{i,k'}$ and $F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{E}_{i,j,k} = \sum_{k'} q(\mathbf{X}^{E}_{t-1,i,j,k} \,|\, \mathbf{X}^{E}_{t,i,j,k}, \mathbf{X}^{E}_{0,i,j,k}) D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{E}_{i,j,k'}$. Clearly, since the values of F_θ depend only pointwise on the values of D_θ, F_θ is aligned permutation equivariant as well.

We continue by directly deriving the connection:

$$p_\theta(\mathbf{RX}_{t-1} \,|\, \mathbf{RX}_t, \mathbf{QY}, \mathbf{RP}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) \tag{30}$$

$$= \prod_i \sum_k (\mathbf{RX}_{t-1})^{\mathcal{N}}_{i,k} F_\theta(\mathbf{RX}_t, \mathbf{QY}, \mathbf{RP}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top)_{i,k} \times \prod_{i,j} \sum_k (\mathbf{RX}_{t-1})^{E}_{i,j,k} F_\theta(\mathbf{RX}_t, \mathbf{QY}, \mathbf{RP}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top)^{E}_{i,j,k} \tag{31}$$

$$= \prod_i \sum_k (\mathbf{X}_{t-1})^{\mathcal{N}}_{\pi(i),k} F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})_{\pi(i),k} \times \prod_{i,j} \sum_k (\mathbf{X}_{t-1})^{E}_{\pi(i),\pi(j),k} F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{E}_{\pi(i),\pi(j),k}, \tag{32}$$

where in the last line we used the aligned permutation equivariance definition, and the the effect of the permutation matrix $\mathbf{R}$ on index i was denoted as $\pi(i)$. Now, regardless of the permutation, the products contain all possible values i and pairs i, j exactly once. Thus, the expression remains equal if we replace $\pi(i)$ with just i:

$$p_\theta(\mathbf{RX}_{t-1} \,|\, \mathbf{RX}_t, \mathbf{QY}, \mathbf{RP}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) \tag{33}$$

$$= \prod_i \sum_k (\mathbf{X}_{t-1})^{\mathcal{N}}_{i,k} F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})_{i,k} \prod_{i,j} \sum_k (\mathbf{X}_{t-1})^{E}_{i,j,k} F_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{E}_{i,j,k} \tag{34}$$

$$= p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}), \tag{35}$$

which concludes the proof. □

Theorem 4. *Aligned denoisers induce aligned permutation invariant distributions If the denoiser function D_θ has the aligned equivariance property and the prior $p(\mathbf{X}_T)$ is permutation invariant,* *then the generative distribution $p_\theta(\mathbf{X}_0 \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$ has the corresponding property for any permutation matrices $\mathbf{R}$ and $\mathbf{Q}$:*

$$p_\theta(\mathbf{R}\mathbf{X}_0 \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = p_\theta(\mathbf{X}_0 \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) \tag{36}$$

Proof. Let us assume that the result holds for some noisy data level t: $p_\theta(\mathbf{R}\mathbf{X}_t \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = p_\theta(\mathbf{X}_t \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$. We will then show that the same will hold for $\mathbf{X}_{t-1}$, which we can use to inductively show that the property holds for $\mathbf{X}_0$. We begin as follows:

$$p_\theta(\mathbf{R}\mathbf{X}_{t-1} \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \sum_{\mathbf{X}_t} p_\theta(\mathbf{R}\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) p_\theta(\mathbf{X}_t \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) \tag{37}$$

$$= \sum_{\mathbf{X}_t} p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{R}^{-1}\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) p_\theta(\mathbf{R}^{-1}\mathbf{X}_t \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}). \tag{38}$$

where on the second line we used Lem. 1 and the assumption that the result holds for noise level t. The sum over $\mathbf{X}_t$ contains all possible graphs and all of their permutations. Thus, the exact value of $\mathbf{R}^{-1}$ does not affect the value of the final sum, as we simply go through the same permutations in a different order, and aggregate the permutations with the sum. Thus,

$$p_\theta(\mathbf{R}\mathbf{X}_{t-1} \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \sum_{\mathbf{X}_t} p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) p_\theta(\mathbf{X}_t \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) \tag{39}$$

$$= p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) \tag{40}$$

showing that if the result holds for level t, then it also holds for level $t-1$. We only need to show that it holds for level $\mathbf{X}_{T-1}$ to start the inductive chain:

$$p_\theta(\mathbf{R}\mathbf{X}_{T-1} \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \sum_{\mathbf{X}_T} p_\theta(\mathbf{R}\mathbf{X}_{T-1} \,|\, \mathbf{X}_T, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) p(\mathbf{X}_T) \tag{41}$$

$$= \sum_{\mathbf{X}_T} p_\theta(\mathbf{X}_{T-1} \,|\, \mathbf{R}^{-1}\mathbf{X}_T, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) p(\mathbf{R}^{-1}\mathbf{X}_T), \tag{42}$$

where on the second line we again used Lem. 1 and the permutation invariance of $p(\mathbf{X}_T)$. Again, the exact value of $\mathbf{R}^{-1}$ does not matter for the sum, since the sum goes through all possible permutations in any case. Thus we have

$$p_\theta(\mathbf{R}\mathbf{X}_{T-1} \,|\, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \sum_{\mathbf{X}_T} p_\theta(\mathbf{X}_{T-1} \,|\, \mathbf{X}_T, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) p(\mathbf{X}_T) \tag{43}$$

$$= p_\theta(\mathbf{X}_{T-1} \,|\, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}). \tag{44}$$

Thus, since the property holds for $\mathbf{X}_{T-1}$, it also holds for $\mathbf{X}_{T-2}, \ldots,$ until $\mathbf{X}_0$. This concludes the proof. □

A.5 Proofs that Our Denoisers Are Aligned Permutation Equivariant

In this section, we show for each of the three alignment methods that the corresponding denoisers do indeed fall within the aligned permutation equivariance function class. Fig. A9 summarizes the different alignment methods.

Atom-mapped positional encodings We start by writing out one side of the aligned permutation equivariance condition, $D_\theta(\mathbf{R}\mathbf{X}_t, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top)$ for this particular function class, and directly show that it equals $\mathbf{P}D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$.

$$D_\theta(\mathbf{R}\mathbf{X}_t, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = f_\theta^{\mathbf{X}}(\left[\mathbf{R}\mathbf{X}_t^N \quad \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top\mathbf{Q}\varphi\right], \mathbf{R}\mathbf{X}^E\mathbf{R}^\top, \left[\mathbf{Q}\mathbf{Y}^N \quad \mathbf{Q}\varphi\right], \mathbf{Q}\mathbf{Y}^E\mathbf{Q}^\top) \tag{45}$$

$$= f_\theta^{\mathbf{X}}(\left[\mathbf{R}\mathbf{X}_t^N \quad \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\varphi\right], \mathbf{R}\mathbf{X}^E\mathbf{R}^\top, \left[\mathbf{Q}\mathbf{Y}^N \quad \mathbf{Q}\varphi\right], \mathbf{Q}\mathbf{Y}^E\mathbf{Q}^\top), \tag{46}$$

where f_θ itself is a function that is permutation equivariant for the combined $\mathbf{X}$ and $\mathbf{Y}$ graph as input. This means that the neural net itself gives an output for the entire combined graph, but we only consider the $\mathbf{X}$ subgraph as the denoiser output, denoted here as $f_\theta^{\mathbf{X}}$. For clarity, we can combine the reactant and product node features and adjacency matrices in the notation, and use the permutation equivariance property of our GNN:

$$f_\theta(\begin{bmatrix}\mathbf{R}\mathbf{X}_t^{\mathcal{N}} & \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\varphi \\ \mathbf{Q}\mathbf{Y}^{\mathcal{N}} & \mathbf{Q}\varphi\end{bmatrix}, \begin{bmatrix}\mathbf{R}\mathbf{X}_t^{E}\mathbf{R}^\top & 0 \\ 0 & \mathbf{Q}\mathbf{Y}^{E}\mathbf{Q}^\top\end{bmatrix}) \tag{47}$$

$$= \begin{bmatrix}\mathbf{R} & 0 \\ 0 & \mathbf{Q}\end{bmatrix} f_\theta(\begin{bmatrix}\mathbf{X}_t^{\mathcal{N}} & \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\varphi \\ \mathbf{Y}^{\mathcal{N}} & \varphi\end{bmatrix}, \begin{bmatrix}\mathbf{X}_t^{E} & 0 \\ 0 & \mathbf{Y}^{E}\end{bmatrix}) \quad \text{(permutation equivariance of base NN)} \tag{48}$$

Taking only the $\mathbf{X}$ part of the output and reverting to D_θ notation, we directly arrive at the result that $D_\theta(\mathbf{R}\mathbf{X}_t, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = \mathbf{R}D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})$.

Directly adding $\mathbf{Y}$ to the output We again start by writing out an aligned equivariant input to the denoiser, with f_θ again denoting a network that is permutation equivariant with respect to the combined $\mathbf{X}$ and $\mathbf{Y}$ graph:

$$D_\theta(\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{X}\to\mathbf{Y}}\mathbf{Q}^\top) = \text{softmax}(f_\theta^{\mathbf{X}}(\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}) + \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top\mathbf{Q}\mathbf{Y}) \tag{49}$$

$$= \text{softmax}(\mathbf{R}f_\theta^{\mathbf{X}}(\mathbf{X}, \mathbf{Y}) + \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}) \quad \text{(permutation equivariance of base denoiser)} \tag{50}$$

$$= \mathbf{R}\,\text{softmax}(f_\theta^{\mathbf{X}}(\mathbf{X}, \mathbf{Y}) + \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}) \tag{51}$$

$$= \mathbf{R}D_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{X}\to\mathbf{Y}}) \tag{52}$$

where we were able to move the permutation outside the softmax since the softmax is applied on each node and edge separately.

Aligning $\mathbf{Y}$ and $\mathbf{X}$ at the input to the model Let's denote by $[\mathbf{X}\ \ \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}]$ concatenation along the feature dimension for both the nodes and edges of the graphs. Recall then that the definition of aligning the graphs in the input is $D_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) = f_\theta([\mathbf{X}\ \ \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}])$, where f_θ is a permutation equivariant denoiser. Writing out the aligned equivariance condition

$$D_\theta(\mathbf{R}\mathbf{X}, \mathbf{Q}\mathbf{Y}, \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top) = f_\theta([\mathbf{R}\mathbf{X}\ \ \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Q}^\top\mathbf{Q}\mathbf{Y}]) = f_\theta([\mathbf{R}\mathbf{X}\ \ \mathbf{R}\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}]) \tag{53}$$

$$= f_\theta(\mathbf{R}[\mathbf{X}\ \ \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}]) = \mathbf{R}f_\theta([\mathbf{X}\ \ \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}]) \tag{54}$$

$$= D_\theta(\mathbf{X}, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}}) \tag{55}$$

which shows that this method results in aligned equivariance as well.

A.6 A Single-layer Graph Transformer with Orthogonal Atom-mapped Positional Encodings is Able to Implement the Identity Data Solution for Nodes

Here, we show that a single-layer Graph Transformer neural net can model the identity data for the nodes given orthogonal atom-mapped positional encodings (e.g., the graph Laplacian eigenvector-based ones). In particular it is possible to find a θ such that $D_\theta(\mathbf{X}_t, \mathbf{Y}, \mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\mathcal{N}} = \mathbf{Y}^{\mathcal{N}}$. We hope that this section can serve as an intuitive motivation for why matched positional encodings help in copying over the structure from the product to the reactant side.

Recall that we have N atoms on both sides of the reaction. Let us map the atom mapping indices to basis vectors in an orthogonal basis $\varphi = [\varphi_1, \varphi_2, \ldots, \varphi_N]^\top$. In practice, the node input to the neural net on the reactant side is now $\mathbf{X}_t^* = [\mathbf{X}_t^{\mathcal{N}}, \varphi]$, and the node input on the product side is $\mathbf{Y}^* = [\mathbf{Y}^{\mathcal{N}}, \varphi]$.

Now, a single-layer Graph Transformer looks as follows (as defined in the Vignac et al. (2023) codebase):

$$\mathbf{N} = [(\mathbf{X}_t^{*\mathcal{N}})^\top, (\mathbf{Y}^{*\mathcal{N}})^\top]^\top \quad \text{(Concatenate rows)} \tag{56}$$

$$\mathbf{E} = \begin{bmatrix} \mathbf{X}_t^E & 0 \\ 0 & \mathbf{Y}^E \end{bmatrix} \quad \text{(Create joined graph)} \tag{57}$$

$$\mathbf{N}_1 = MLP_N(\mathbf{N}) \quad \text{(Applied with respect to last dimension)} \tag{58}$$

$$\mathbf{E}_1 = \frac{1}{2}(MLP_E(\mathbf{E}) + MLP_E(\mathbf{E})^\top) \quad \text{(Symmetrize the input)} \tag{59}$$

$$t_1 = MLP_t(t) \tag{60}$$

$$\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{W}_Q\mathbf{N}_1, \mathbf{W}_K\mathbf{N}_1, \mathbf{W}_V\mathbf{N}_1 \quad \text{(One attention head for simplicity)} \tag{61}$$

$$\mathbf{A}_1 = \mathbf{Q}\mathbf{K}^\top \tag{62}$$

$$\mathbf{A}_2 = \mathbf{A}_1 * (\mathbf{W}_{\mathbf{E}_2}\mathbf{E}_1 + 1) + \mathbf{W}_{\mathbf{E}_3}\mathbf{E}_1 \tag{63}$$

$$\mathbf{A}_3 = \text{softmax}(\mathbf{A}_3) \tag{64}$$

$$\mathbf{N}_2 = \mathbf{A}\mathbf{V}/\sqrt{d_f} \quad (d_f \text{ is the embedding dim of V}) \tag{65}$$

$$\mathbf{N}_3 = \mathbf{N}_2 * (\mathbf{W}_{t_2}t_1 + 1) + \mathbf{W}_{t_3}t_1 \tag{66}$$

$$\mathbf{E}_2 = \mathbf{W}(\mathbf{A}_2 * (\mathbf{W}_{t_4}t_1 + 1) + W_{t_5}t_1) \tag{67}$$

$$\begin{aligned} t_4 = {} & MLP_{t_4}(\mathbf{W}t_2 + \mathbf{W}[\min(\mathbf{N}_3), \max(\mathbf{N}_3), \text{mean}(\mathbf{N}_3), \text{std}(\mathbf{N}_3)] \\ & + \mathbf{W}[\min(\mathbf{N}_3), \max(\mathbf{N}_3), \text{mean}(\mathbf{N}_3), \text{std}(\mathbf{N}_3)] \end{aligned} \tag{68}$$

$$\mathbf{E}_3 = MLP_{\mathbf{E}_2}(\mathbf{E}_2) + \mathbf{E} \tag{69}$$

$$\mathbf{E}_{out} = \frac{1}{2}(\mathbf{E}_3 + \mathbf{E}_3^\top) \quad \text{(Symmetrize the output)} \tag{70}$$

$$\mathbf{N}_{out} = MLP_{N_3}(\mathbf{N}_3) + \mathbf{N} \tag{71}$$

$$t_{out} = t_4 + t \tag{72}$$

Now, for purposes of illustration, we can define most linear layers to be zero layers: $MLP_E, MLP_t, \mathbf{W}_{E_2}, \mathbf{W}_{E_3}, \mathbf{W}_{t_2}, \mathbf{W}_{t_3}, \mathbf{W}_{t_4}, \mathbf{W}_{t_5} = 0$. In addition to this, we define MLP_N to be an identity transform. $\mathbf{W}_Q$ and $\mathbf{W}_K$ are both chosen as picking out the $\mathbf{U}$ columns of $\mathbf{N}_1$, with additional overall scaling by some constant α. $\mathbf{W}_V$ is chosen to pick out the product node labels: $\mathbf{W}_V\mathbf{N}_1 = \begin{bmatrix} 0 \\ \mathbf{Y}^{\mathcal{N}} \end{bmatrix}$. Now, we can easily see how the Graph Transformer can obtain the optimal denoising solution for the nodes. Consider an input $\mathbf{N} = [(\mathbf{P}\mathbf{X}_t^*)^\top, (\mathbf{Y})^\top]^\top$, where the reactant side is permuted. The output of the network should be $\mathbf{P}\mathbf{Y}$. Focusing on the parts of the network that compute the node features:

$$\mathbf{A}_1 = \alpha^2 \begin{bmatrix} \mathbf{P}\varphi\varphi^\top\mathbf{P}^\top & \mathbf{P}\varphi\varphi^\top \\ \varphi\varphi^\top\mathbf{P}^\top & \varphi\varphi^\top \end{bmatrix} = \alpha^2 \begin{bmatrix} \mathbf{I} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{I} \end{bmatrix}, \tag{73}$$

$$\mathbf{A}_2 = \mathbf{A}_1, \tag{74}$$

$$\mathbf{A}_3 \approx \text{softmax}(\mathbf{A}_2) = \frac{1}{2}\begin{bmatrix} \mathbf{I} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{I} \end{bmatrix} \quad (\text{If } \alpha \gg 1), \tag{75}$$

$$\mathbf{N}_2 = \frac{\mathbf{A}_3\mathbf{V}}{\sqrt{d_F}} = \begin{bmatrix} \mathbf{I} & \mathbf{P} \\ \mathbf{P}^\top & \mathbf{I} \end{bmatrix} \begin{bmatrix} 0 \\ Y_N \end{bmatrix} /(2\sqrt{d_f}) = \begin{bmatrix} \mathbf{P}\mathbf{Y}_N \\ \mathbf{Y}_N \end{bmatrix} /(2\sqrt{d_f}). \tag{76}$$

Here, we used the fact that we chose the positional embeddings to be an orthogonal basis, and $\varphi\varphi^\top = \mathbf{I}$, as well as the fact that $\mathbf{P}\mathbf{P}^\top = \mathbf{I}$ for any permutation matrices. The term $\frac{1}{2}$ in the third equation came from the fact that each row of $\mathbf{A}_1$ contains two non-zero values that are also equal. The probability gets divided between the two of them in the softmax if the logits are scaled large enough, and the approximation becomes arbitrarily accurate.

From now on, since we are interested in the denoising output only for the reactant side, we drop out the reactant side $\mathbf{Y}^{\mathcal{N}}$ and only focus on the $\mathbf{P}\mathbf{Y}^{\mathcal{N}}$ part. We choose the final $MLP_{\mathbf{N}_2}$ to scale the output by some factor $\beta \gg 1$:

$$\mathbf{N}_{out} = \beta\mathbf{N}_2 + \mathbf{N} \tag{77}$$

$$n_\theta(\mathbf{P}\mathbf{X}_t^*, \mathbf{Y}^*)^{\mathcal{N}} = \text{softmax}(\beta\mathbf{P}\mathbf{Y}^{\mathcal{N}} + \mathbf{N}) \approx \mathbf{P}\mathbf{Y}^{\mathcal{N}}. \tag{78}$$

Here, the approximation can be made arbitrarily accurate by scaling β to a higher value, since the logits will become more and more peaked towards the values where $P\mathbf{Y}^{\mathcal{N}}$ equals one instead of zero. This showcases how the attention mechanism in the Graph Transformer pairs with atom mapping-based orthogonal positional encodings to achieve the identity function from products to reactants.

A.7 Extending Theorem 1 to Smaller Time Steps

We prove the following result characterizing the optimal denoiser output for a more restricted setup of absorbing-state diffusion and on an identity data with no edges (i.e., just a set of nodes):

$$D_\theta(X_t, Y)_{i,k} = \begin{cases} Y_{i,k}, & \text{if } X_{t,i} \text{ is not in the absorbing state} \\ \hat{y}_k^M, & \text{if } X_{t,i} \text{ is in the absorbing state} \end{cases} \quad \text{where } \hat{y}_k^M = \frac{\sum_{j\in M_t} Y_{j,k}}{\sum_{j\in M_t}\sum_k Y_{j,k}}$$

where M_t refers to the set of nodes in the mask state. We will include the proof as an additional comment below. In words, the optimal output for the masked nodes is the marginal distribution of the nodes that have not been de-masked yet, that is, the mean of all possible placements of the remaining nodes.

The analysis for graphs with edges becomes more involved, but we postulate that a similar analysis would show that the output would converge to a mean of all possible permutations of Y that are in agreement with the current set of subgraphs X_t.

To recap, we have the simplified scenario where we generate a set of nodes X conditioned on another set of nodes Y, without edges. The goal is to characterize the optimal permutation equivariant denoiser for this process at any noise level.

Definitions. Let $X_0 \in \{0,1\}^{N\times K}$ be the initial set of N nodes, each represented by a one-hot vector of dimension K. We condition on $Y \in \{0,1\}^{N\times K}$, where $X_0 = Y$ in our data. The forward process is an absorbing state diffusion, defined as follows for $t \in 0, 1, \ldots, T$:

$$q(X_{t,i} = M | X_{0,i}) = t/T \tag{79}$$
$$q(X_{t,i} = X_{0,i} | X_{0,i}) = 1 - t/T \tag{80}$$

where M represents the additional "mask" state. We denote the denoiser as $D_\theta(X_t, Y)$, which is assumed to be permutation equivariant with respect to X_t.

Notation For a given X_t, we define: $M(X_t) \subset 1, \ldots, N$: Set of indices of masked nodes $U(X_t) = 1, \ldots, N \setminus M(X_t)$: Set of indices of non-masked nodes $\hat{y}_k^{M(X_t)} = \frac{1}{|M(X_t)|}\sum_{i\in M(X_t)} Y_{i,k}$: Marginal distribution of node values in Y for the node indices that are masked in X_t

Result

Theorem 5. *The optimal permutation equivariant denoiser $D_\theta(X_t, Y)$ for the absorbing state diffusion process at any time t and for any particular X_t is given by: $D_\theta(X_t, Y)_{i,k} = Y_{i,k}$, if $i \in U(X_t)$ $D_\theta(X_t, Y)_{i,k} = \hat{y}_k^{M(X_t)}$, if $i \in M(X_t)$*

Proof. The cross-entropy loss for the denoiser at time t is:

$$CE_t = -\mathbb{E}_{q(X_t|X_0)} \sum_{i=1}^{N} \sum_k X_{0,i,k} \log D_\theta(X_t, Y)_{i,k} \tag{81}$$

$$= -\mathbb{E}_{q(X_t|X_0)} \left[\sum_{i\in U(X_t)} \sum_k X_{0,i,k} \log D_\theta(X_t, Y)_{i,k} + \sum_{i\in M(X_t)} \sum_k X_{0,i,k} \log D_\theta(X_t, Y)_{i,k} \right] \tag{82}$$

We consider the optimal denoiser output for non-masked and masked nodes separately:

Non-masked nodes ($i \in U(X_t)$) For these nodes, the optimal output is clearly $D_\theta(X_t, Y)_{i,k} = Y_{i,k}$, since $Y_{i,k} = X_{0,i,k}$.

Masked nodes ($i \in M(X_t)$) Let's focus on a particular instantiation of X_t. Due to the permutation equivariance of D_θ, for any permutation P that only permutes indices within $M(X_t)$, we have: $D_\theta(PX_t, Y) = PD_\theta(X_t, Y)$. Moreover, because all nodes in $M(X_t)$ are in the same mask state, all such permutations of X_t are identical: $D_\theta(PX_t, Y) = D_\theta(X_t, Y)$ Combining these facts, we conclude that for any $i, j \in M(X_t)$: $D_\theta(X_t, Y)_i = D_\theta(X_t, Y)_j$ Thus, the denoiser must output the same value for all masked nodes in a given X_t. Now, consider the part of the cross-entropy loss corresponding to the masked nodes for this particular X_t:

$$CE_t^M(X_t) = -\sum_{i \in M(X_t)} \sum_k X_{0,i,k} \log D_\theta(X_t, Y)_{i,k} \tag{83}$$

$$= -\sum_k \left(\sum_{i \in M(X_t)} X_{0,i,k} \right) \log D_\theta(X_t, Y)_{m,k} \tag{84}$$

where m is any index in $M(X_t)$. Observe that $\sum_{i \in M(X_t)} X_{0,i,k}$ is proportional to the marginal distribution of the masked nodes in X_0 (which is equal to Y). Using our defined notation, we can rewrite the cross-entropy as: $CE_t^M(X_t) \propto -|M(X_t)| \sum_k \hat{y}_k^{M(X_t)} \log D_\theta(X_t, Y)_{m,k}$ The optimal $D_\theta(X_t, Y)_{m,k}$ that minimizes this expression is clearly $\hat{y}_k^{M(X_t)}$. Combining the results for masked and non-masked nodes yields the stated optimal denoiser. $\square$

B Details on Conditional Graph Diffusion

Our transition matrices To define $\mathbf{Q}_t^{\mathcal{N}}$ and $\mathbf{Q}_t^E$, we adopt the absorbing-state formulation from Austin et al. (2021), where nodes and edges gradually transfer to the *absorbing state* $\perp$. Formally, we give the generic form of the transition matrix $\mathbf{Q}_t$ for node input $\mathbf{X}^{\mathcal{N}} \in \mathbb{R}^{N_X \times N_X}$ [1]

$$\mathbf{Q}_t = (1 - \beta_t)\mathbf{I} + \beta_t \mathbb{1} e_\perp^\top, \tag{85}$$

where β_t defines the *diffusion schedule* and $e_\perp$ is one-hot on the absorbing state $\perp$. For completeness, we list the other two common transitions relevant to our application. The first is the uniform transition as proposed by Hoogeboom et al. (2021)

$$\mathbf{Q}_t = (1 - \beta_t)\mathbf{I} + \beta_t \frac{\mathbb{1}\mathbb{1}^\top}{K} \tag{86}$$

where β_t, I are as before and K is the number of element (edge or node) types, i.e., the number of input features for both nodes and edges. Vignac et al. (2023) also proposed a marginal transition matrix

$$\mathbf{Q}_t^{\mathcal{N}} = (1 - \beta^t)\mathbf{I} + \beta^t \mathbb{1}\left(m^{\mathcal{N}}\right)^\top \quad \text{and} \quad \mathbf{Q}_t^E = (1 - \beta^t)\mathbf{I} + \beta^t \mathbb{1}\left(m^E\right)^\top \tag{87}$$

which they argued leads to faster convergence. In this case, $m^{\mathcal{N}} \in \mathbb{R}^{K_a}$ and $m^E \in \mathbb{R}^{K_b}$ are row vectors representing the marginal distributions for node and edge types respectively. We tested all three types of transition matrices in early experiments and noted the absorbing state model to be slightly better than the others. The marginal $q(\mathbf{X}_t \mid \mathbf{X}_0)$ and conditional posterior $q(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{X}_0)$ also have a closed form for all of these transition matrices.

Noise schedule We use the mutual information noise schedule proposed by Austin et al. (2021), which leads to

$$\frac{t}{T} = 1 - \frac{I(\mathbf{X}_t; \mathbf{X}_0)}{H(\mathbf{X}_0)} = \frac{H(\mathbf{X}_0, \mathbf{X}_t) - H(\mathbf{X}_t)}{H(\mathbf{X}_0)} = \frac{\sum_{\mathbf{X}_0, \mathbf{X}_t} p(\mathbf{X}_0) q(\mathbf{X}_t \mid \mathbf{X}_0) \log \frac{q(\mathbf{X}_t \mid \mathbf{X}_0)}{\sum_{\mathbf{X}_0'} p(\mathbf{X}_0') q(\mathbf{X}_t \mid \mathbf{X}_0')}}{\sum_{\mathbf{X}_0} p(\mathbf{X}_0) \log p(\mathbf{X}_0)} \tag{88}$$

For absorbing state diffusion, these equations lead to $\beta_t = \frac{1}{T-t+1}$. Similarly, the total transition probability to the absorbing state at time t has a simple form: $q(\mathbf{X}_t = \perp \mid \mathbf{X}_0) = \frac{t}{T}$.

[1]The only difference between $\mathbf{Q}_t^{\mathcal{N}}$ and $\mathbf{Q}_t^E$ for the absorbing-state and uniform transitions is the dimensions of I, e, $\mathbb{1}$ and the value of K. We therefore give a generic form for both and imply choosing the right dimensions for each case.

Forward process posterior For transition matrices that factorize over dimensions, we have

$$q(\mathbf{X}_{t-1,i,:} \,|\, \mathbf{X}_{t,i,:}, \mathbf{X}_{0,i,:}) \sim \frac{\mathbf{X}_t \mathbf{Q}_t^\top \cdot \mathbf{X}_{0,i,:} \bar{\mathbf{Q}}_{t-1}}{\mathbf{X}_{0,i,:} \bar{\mathbf{Q}}_t \mathbf{X}^\top} \tag{89}$$

where $\mathbf{X}_{i,:}$ is the one-hot encoding of i$^{\text{th}}$ node/edge of the graph, in row vector format.

Variational lower-bound loss Diffusion models are commonly trained by minimizing the negative variational lower-bound on the model's likelihood (Ho et al., 2020). Austin et al. (2021) discuss the difference between optimizing the ELBO and cross-entropy losses and show that the two losses are equivalent for the absorbing-state transition. We choose to use cross-entropy, similar to Vignac et al. (2023), due to faster convergence during training. We include the formula for the ELBO in Eq. (90) for completeness.

$$\begin{aligned}\mathcal{L}_{\text{vb}} = \mathbb{E}_{\mathbf{X}_0 \sim q(\mathbf{X}_0)} \Big[& \underbrace{\mathrm{KL}\big[q(\mathbf{X}_T \,|\, \mathbf{X}_0) \,\|\, p(\mathbf{X}_T)\big]}_{\mathcal{L}_T} \\ & + \underbrace{\sum_{t=2}^{T} \mathbb{E}_{\mathbf{X}_t \sim q(\mathbf{X}_t \,|\, \mathbf{X}_0)} \mathrm{KL}\big[q(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{X}_0) \,\|\, p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t)\big]}_{\mathcal{L}_{t-1}} \\ & - \underbrace{\mathbb{E}_{\mathbf{X}_1 \sim q(\mathbf{X}_1 \,|\, \mathbf{X}_0)} \log p_\theta(\mathbf{X}_0 \,|\, \mathbf{X}_1)}_{\mathcal{L}_0} \Big]\end{aligned} \tag{90}$$

We also note that we use this quantity as part of the scoring function mentioned in Sec. 9.

Data encoding and atom-mapping We illustrate our graph encoding using atom-mapping and permutation matrices in Fig. A10.

Connection between the denoising parameterization $\tilde{p}(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y})$ and $p(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y})$. Here, we elaborate on how does $\tilde{p}(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y})$ connect with $p(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y})$.

Recall that the reverse transition is factorized with respect to nodes and edges:

$$p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) = \textstyle\prod_{i=1}^{N_{\mathbf{X}}} p_\theta(\mathbf{X}_{t-1}^{\mathcal{N},i} \,|\, \mathbf{X}_t, \mathbf{Y}) \prod_{i,j}^{N_{\mathbf{X}}} p_\theta(\mathbf{X}_{t-1}^{E,i,j} \,|\, \mathbf{X}_t, \mathbf{Y}). \tag{91}$$

We parameterize the transition as follows:

$$p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) = \textstyle\sum_{\mathbf{X}_0} q\big(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{X}_0\big) \tilde{p}_\theta\big(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y}\big). \tag{92}$$

The connection between this parameterization and the probabilties $p_\theta(\mathbf{X}_{t-1}^{\mathcal{N},i} \,|\, \mathbf{X}_t, \mathbf{Y})$ is obtained by noting that both $q\big(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{X}_0\big)$ and $\tilde{p}_\theta\big(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y}\big)$ factorize over dimensions:

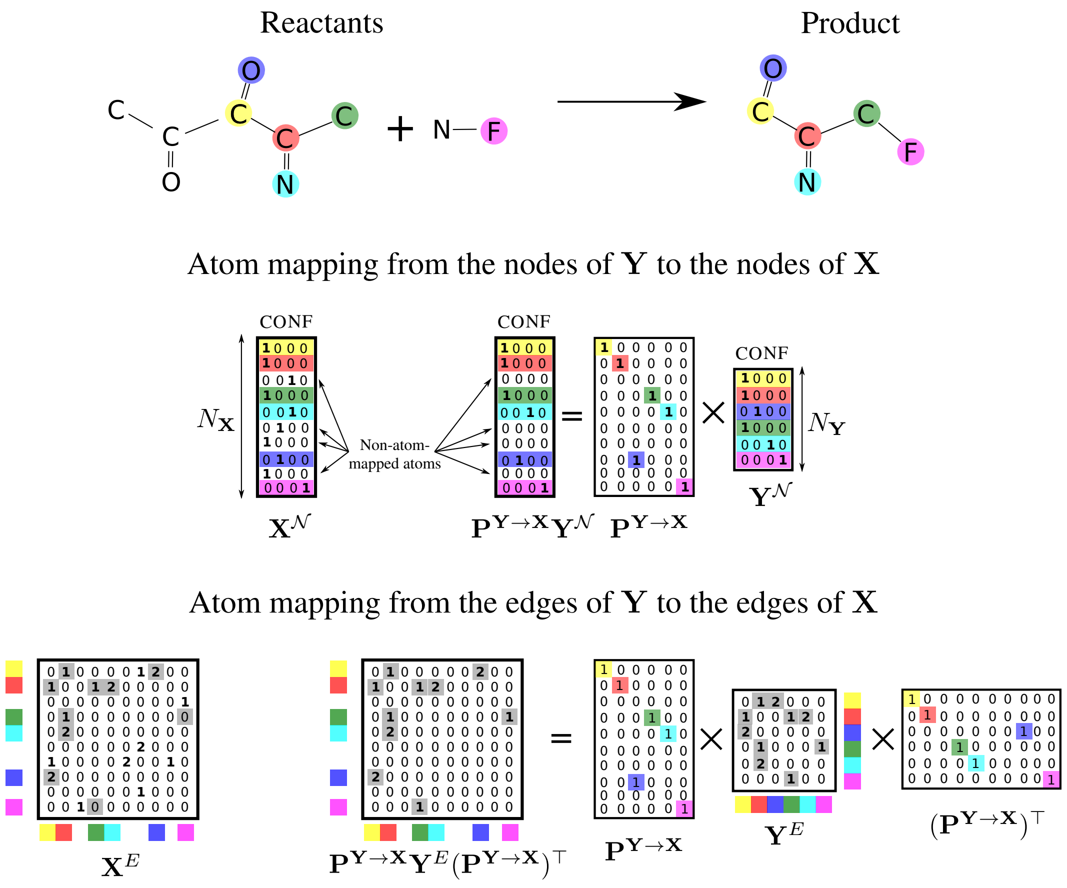
$$q\big(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{X}_0\big) = \prod_{i}^{N_{\mathbf{X}}} q\big(\mathbf{X}_{t-1}^{\mathcal{N},i} \,|\, \mathbf{X}_t^{\mathcal{N},i}, \mathbf{X}_0^{\mathcal{N},i}\big) \prod_{i,j}^{N_{\mathbf{X}}} q\big(\mathbf{X}_{t-1}^{E,i,j} \,|\, \mathbf{X}_t^{E,i,j}, \mathbf{X}_0^{E,i,j}\big) \tag{93}$$

$$\tilde{p}_\theta\big(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y}\big) = \prod_{i}^{N_{\mathbf{X}}} \tilde{p}_\theta\big(\mathbf{X}_0^{\mathcal{N},i} \,|\, \mathbf{X}_t, \mathbf{Y}\big) \prod_{i,j}^{N_{\mathbf{X}}} \tilde{p}_\theta\big(\mathbf{X}_0^{E,i,j} \,|\, \mathbf{X}_t, \mathbf{Y}\big) \tag{94}$$

Plugging these in to Eq. (92) and expanding the sum, we get

$$\begin{aligned}p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) = \sum_{\mathbf{X}_0^{\mathcal{N},0}} \sum_{\mathbf{X}_0^{\mathcal{N},1}} \cdots \sum_{\mathbf{X}_0^{E,N_{\mathbf{X}},N_{\mathbf{X}}}} & \Bigg(\prod_{i}^{N_{\mathbf{X}}} q\big(\mathbf{X}_{t-1}^{\mathcal{N},i} \,|\, \mathbf{X}_t^{\mathcal{N},i}, \mathbf{X}_0^{\mathcal{N},i}\big) \tilde{p}_\theta\big(\mathbf{X}_0^{\mathcal{N},i} \,|\, \mathbf{X}_t, \mathbf{Y}\big) \\ & \prod_{i,j}^{N_{\mathbf{X}}} q\big(\mathbf{X}_{t-1}^{E,i,j} \,|\, \mathbf{X}_t^{E,i,j}, \mathbf{X}_0^{E,i,j}\big) \tilde{p}_\theta\big(\mathbf{X}_0^{E,i,j} \,|\, \mathbf{X}_t, \mathbf{Y}\big) \Bigg)\end{aligned} \tag{95}$$

Figure A10: Illustrating graph encoding using atom-mapping.


Reactants
Product
Atom mapping from the nodes of $\mathbf{Y}$ to the nodes of $\mathbf{X}$
CONF
$N_{\mathbf{X}}$
Non-atom-mapped atoms
$\mathbf{X}^{\mathcal{N}}$
$\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{\mathcal{N}}$
$\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$
$N_{\mathbf{Y}}$
$\mathbf{Y}^{\mathcal{N}}$
Atom mapping from the edges of $\mathbf{Y}$ to the edges of $\mathbf{X}$
$\mathbf{X}^{E}$
$\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}\mathbf{Y}^{E}(\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\top}$
$\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$
$\mathbf{Y}^{E}$
$(\mathbf{P}^{\mathbf{Y}\to\mathbf{X}})^{\top}$

Gathering terms together, we get

$$= \underbrace{\left(\sum_{\mathbf{X}_0^{\mathcal{N},0}} q(\mathbf{X}_{t-1}^{\mathcal{N},0} \mid \mathbf{X}_t^{\mathcal{N},0}, \mathbf{X}_0^{\mathcal{N},0}) \tilde{p}_\theta(\mathbf{X}_0^{\mathcal{N},0} \mid \mathbf{X}_t, \mathbf{Y}) \right)}_{= p_\theta(\mathbf{X}_{t-1}^{\mathcal{N},0} \mid \mathbf{X}_t, \mathbf{Y})} \cdot \tag{96}$$

$$\underbrace{\left(\sum_{\mathbf{X}_0^{\mathcal{N},1}} q(\mathbf{X}_{t-1}^{\mathcal{N},1} \mid \mathbf{X}_t^{\mathcal{N},1}, \mathbf{X}_0^{\mathcal{N},1}) \tilde{p}_\theta(\mathbf{X}_0^{\mathcal{N},1} \mid \mathbf{X}_t, \mathbf{Y}) \right)}_{= p_\theta(\mathbf{X}_{t-1}^{\mathcal{N},1} \mid \mathbf{X}_t, \mathbf{Y})} \cdot \tag{97}$$

$$\dots \tag{98}$$

$$\underbrace{\left(\sum_{\mathbf{X}_0^{E,N_{\mathbf{X}},N_{\mathbf{X}}}} q(\mathbf{X}_{t-1}^{E,N_{\mathbf{X}},N_{\mathbf{X}}} \mid \mathbf{X}_t^{E,N_{\mathbf{X}},N_{\mathbf{X}}}, \mathbf{X}_0^{E,N_{\mathbf{X}},N_{\mathbf{X}}}) \tilde{p}_\theta(\mathbf{X}_0^{E,N_{\mathbf{X}},N_{\mathbf{X}}} \mid \mathbf{X}_t, \mathbf{Y}) \right)}_{= p_\theta(\mathbf{X}_{t-1}^{E,N_{\mathbf{X}},N_{\mathbf{X}}} \mid \mathbf{X}_t, \mathbf{Y})} \tag{99}$$

$$= \prod_{i=1}^{N_X} p_\theta(\mathbf{X}_{t-1}^{\mathcal{N},i} \mid \mathbf{X}_t, \mathbf{Y}) \prod_{i,j}^{N_X} p_\theta(\mathbf{X}_{t-1}^{E,ij} \mid \mathbf{X}_t, \mathbf{Y}). \tag{100}$$

C Experimental Setup

C.1 Copying Task on the Grid Dataset

Data generation We generate 5×5 fully connected grids then noise them by flipping a fixed portion of edges chosen at random. We set the portion of edges to be flipped to 5% of the total number of edges (i.e., for 5×5 grids, 31 edges get flipped).

Neural network We use the same neural network architecture for both the aligned and unaligned denoisers. We use a graph transformer Dwivedi & Bresson (2021) with 2 layers, 1 attention head, and dimension 16 for the hidden layer. The activation function used is ReLu, and dropout rate of 0.1.

Training hyperparameters We train both models for 10 epochs, using Adam optimizer with a learning rate of 0.01. We set the batch size to 32.

C.2 Data: USPTO Data Sets

All open-source data sets available for reaction modelling are derived in some form from the patent mining work of Lowe (2012). We distinguish 5 subsets used in previous work: 15k, 50k, MIT, Stereo, and full (original data set). Table A3 provides key information about the subsets.

Table A3: UPSTO-50K subsets used in retrosynthesis

Subset	Introduced by	# of reactions	Preprocessing & Data split (script)
Full	Lowe (2012)	1 808 938	Dai et al. (2019)
Stereo	Schwaller et al. (2018)	1 002 970	Schwaller et al. (2019b)
MIT	Jin et al. (2017)	479 035	-
50k	Schneider et al. (2016)	50 016	Dai et al. (2019)

15k is proposed by Coley et al. (2017a). The subset includes reactions covered by the 1.7k most common templates. All molecules appearing in the reaction are included to model the involvement of reagents and solvents despite not contributing with atoms to the product.

50k is preprocessed by Schneider et al. (2016). The goal of the analysis is to assign roles (reactant, reagent, solvent) to different participants in a reaction through atom mapping. This effort led to the creation of an atom-mapped and classified subset of around 50k reactions, which is used nowadays as a benchmark for retrosynthesis tasks. It is not clear how said subset was selected.

MIT is used by Jin et al. (2017). The preprocessing is described as 'removing duplicate and erroneous reactions' with no further explanation of what qualifies as an erroneous reaction. The output of this filtering is a data set of 49k reactions (from an original set of 1.8M reactions).

Stereo is proposed by Schwaller et al. (2018). The authors apply a more flexible filtering strategy compared to USPTO-MIT. Their data set only discards 800k reactions from the original data set because they are duplicates or they could not be canonicalized by RDKit. In addition, the data set only considers single-product reactions (92% of the full data set), as opposed to splitting multi-product reactions. The preprocessing steps include removing reagents (molecules with no atoms appearing in the product), removing hydrogen atoms from molecules, discarding atom-mapping information and canonicalizing molecules. In addition, since the original method applied to this subset is a language model, tokenization is performed on the atoms.

Full is preprocessed by Dai et al. (2019). The processing includes removing duplicate reactions, splitting reactions with multiple products into multiple reactions with one product, removing reactant molecules appearing unchanged on the product side, removing all reactions with bad atom-mapping (i.e., when the sorted mapping between products and reactants is not one-to-one), and removing bad products (missing mapping, or not parsed by Rdkit).

Our choice Similar to many other works on retrosynthesis, we use 50k as the main data set to evaluate our method.

C.3 Notes on Our Sampling and Ranking Procedures

Duplicate removal Removing duplicates from the set of generated precursors is a common methodology in retrosynthesis, albeit often not discussed explicitly in papers. The benefit of duplicate removal is to ensure that an incorrect molecule that is nevertheless judged as the best one according to the ranking scheme does not fill up all of the top-k positions after ranking. While this does not affect top-1 scores, not removing the duplicates would degrade the other top-k scores significantly.

Choice of scoring function Specifically, we use the following formula for approximating the likelihood of the sample under the model.

$$s(\mathbf{X}) = (1-\lambda)\frac{\mathrm{count}(\mathbf{X})}{\sum_{\mathbf{X}'}\mathrm{count}(\mathbf{X}')} + \lambda \cdot \frac{e^{\mathrm{elbo}(\mathbf{X})}}{\sum_{\mathbf{X}'} e^{\mathrm{elbo}(\mathbf{X}')}}, \tag{101}$$

where count(.) returns the number of occurrences of sample X in the set of generated samples (by default, 100), elbo(.) computes the variational lower bound of the specific sample under the model, and λ is a weighting hyperparameter. The sums are taken over the set of generated samples. Intuitively, the idea is to provide an estimate of the likelihood from two routes. The ELBO is an estimate (a lower bound) of $\log p_\theta(\mathbf{X})$, and exponentiating and normalizing gives an estimate of the probability distribution. Since the same reactants are often repeated in a set of 100 samples, the counts can be used as a more direct proxy, although they inherently require a relatively large amount of samples to limit the variance of this estimate.

We set the value of λ to be 0.9, although we find that the top-k scores are not at all sensitive to variation in the exact value, as long as it is below 1, and the count information is used. Thus, the counts seem more important than the ELBO, which may be due to the lower bound nature of the ELBO or stochasticity in estimating its value. More accurate likelihood estimation schemes for diffusion models, such as exact likelihood values using the probability flow ODE (Song et al., 2021), could be a valuable direction for future research in the context of retrosynthetic diffusion models.

C.4 Details on Stereochemistry

Our model does not explicitly consider changes in stereochemistry in the reaction, but instead, we use the atom mappings implicitly assigned to the samples by the model to transfer the chiral tags from the products to the reactants. The initial choice of $\mathbf{P}^{\mathbf{Y}\to\mathbf{X}}$ at the start of sampling can be considered to be the atom mapping of the generated reactants, given that the model has been trained on correct atom mappings.

For the chiral tags, we take the ground-truth SMILES for the product molecules from the dataset and assign the corresponding chiral tag to the corresponding atom mapping on the generated reactants. For cis/trans isomerism, we use the `Chem.rdchem.BondDir` bond field in rdkit molecules and transfer them to the reactant side based on the atom mapping of the pair of atoms at the start and end of the bond.

Note that when using rdkit, transferring chirality requires some special care: The chiral tags `Chem.ChiralType.CHI_TETRAHEDRAL_CCW` and `Chem.ChiralType.CHI_TETRAHEDRAL_CW` are defined in the context of the order in which the bonds are attached to the chiral atom in the molecule data structure. Thus, the chiral tag sometimes has to be flipped to retain the correct stereochemistry, based on whether the order of the bonds is different on the reactant molecule data structure and the product molecule data structure.

C.5 Details of the Evaluation Procedure

Top-k scores We evaluate the top-k scores by ranking the list of generated and deduplicated samples and calculating the percentage of products for which the ground-truth reactants are in the first k elements in the list.

Mean reciprocal rank We formally define the MRR as MRR $= \mathbb{E}_{p(r)}[r^{-1}]$. Verbally, it is the expected value of the inverse of the amount of reactant suggestions that the model makes before encountering the ground truth, and as such measures how early on is the correct reactant encountered in the ranked samples. It also incorporates the intuition that the difference between obtaining the correct reactants in, say, the 9th or the 10th position, is not as significant as the difference between the 1st and 2nd positions.

While we do not have direct access to the entire $p(r)$ just based on the top-k scores, we can estimate it with a uniform distribution assumption on r within the different top-k ranges. Formally, we define four sets $S_1, S_2, S_3, S_4 = \{1\}, \{2, 3\}, \{4, 5\}, \{6, 7, 8, 9, 10\}$ in which $p(r)$ is assumed to be uniform, and $s(r) \in \{1, 2, 3, 4\}$ denotes the group that rank r belongs to. Top-k is denoted as top_k, where $k \in \{1, 3, 5, 10\}$ in our case. Note that they are also equal to the cumulative distribution of $p(r)$ until k. We thus define $\hat{p}(r) = \frac{\text{top}_{\max(G_{s(r)})} - \text{top}_{\max(G_{s(r)-1})}}{|G_{s(r)}|}$ and $\hat{p}(1) = \text{top}_1$. For the case where the ground truth was not in the top-10, we assume it is not recovered and place the rest of the probability mass on $p(\infty)$. Our MRR estimate is then defined as

$$\widehat{\text{MRR}} = \sum_{r=1}^{10} \hat{p}(r) \frac{1}{r}. \tag{102}$$

In cases where we do not have top-k values for all {1,3,5,10} (such as the Augmented Transformer in Table 2 for top-3), we assume that $\hat{p}(r)$ is constant in the wider interval between the preceding and following top-ks (2–5 in the case of the top-3 missing).

C.6 Neural Network Architecture, Hyperparameters, and Compute Resources

We discuss here the neural network architecture and hyper-parameters we choose. Our denoiser is implemented as a Graph Transformer (Dwivedi & Bresson, 2021), based on the implementation of Vignac et al. (2023) with additional graph-level level features added to the input of the model. See Vignac et al. (2023) for an in-depth discussion of the neural network.

In all of our models, we use 9 Graph Transformer layers. When using Laplacian positional encodings, we get the 20 eigenvectors of the Graph Laplacian matrix with the largest eigenvalues and assign to each node a 20-dimensional feature vector.

We use a maximum of 15 'blank' nodes, in practice meaning that the models have the capacity to add 15 additional atoms on the reactant side. In another detail, following Vignac et al. (2023), we weigh the edge components in the cross-entropy loss by a factor of 5 compared to the node components.

We used a batch size of 16 for the models where the expanded graph containing X and Y as subgraphs is given as input. These models were trained for approximately 600 epochs with a single A100/V100/AMD MI250x GPU. For the model where alignment is done by concatenating $\mathbf{Y}$ along the feature dimension in the input, the attention map sizes were smaller and we could fit a larger batch of 32 with a single V100 GPU. This model was trained for 600 epochs. The training time for all of our models was approximately three days. In early experiments and developing the model, we trained or partially trained multiple models that did not make it to the main paper. Sampling 100 samples for one product with $T = 100$ from the model takes roughly 60 seconds with the current version of our code with an AMD MI250x GPU, and 100 samples with $T = 10$ takes correspondingly about 6 seconds. It is likely that the inference could be optimized, increasing the sample throughput.

The reported models were chosen based on evaluating different checkpoints with 10 diffusion steps on the validation set for different checkpoints and chose the best checkpoint based on the MRR score.

D Comparision to Retrosynthetic Baselines

Overview of retrosynthetic baselines There are three main types of retrosynthesis models (Liu et al., 2023). Template-based models depend on the availability and quality of hard-coded chemical rules (Segler & Waller, 2017; Xie et al., 2023). Synthon-based models are limited by their definition of a reaction center, which does not necessarily hold for complex reactions (Yan et al., 2020; Shi et al., 2020a; Wang et al., 2023). See Fig. A11 for an example of a reaction impossible for synthon-based models but which our model gets correctly. Template-free methods are the most scalable since they do not use any chemical assumptions in their design but perform suboptimally compared to template-based methods on benchmarks like UPSTO-50k (Wan et al., 2022; Seo et al.,

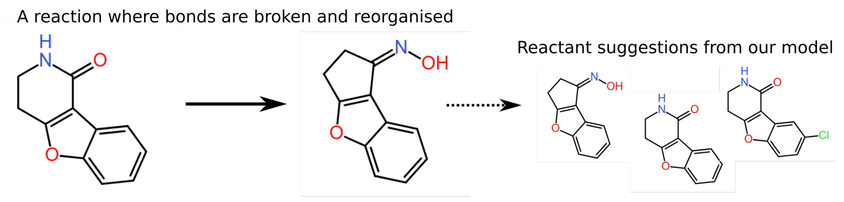

A reaction where bonds are broken and reorganised
Reactant suggestions from our model

Figure A11: Left: An example reaction from USPTO-50k where the reactant is not possible to predict with a synthon-based model. Right: example reactants generated by our model (includes the ground-truth one). The SMILES-string for the reaction is `ON=C1CCc2oc3ccccc3c21»O=C1NCCc2oc3ccccc3c21`.

Table A4: Comparison of top-k accuracy and MRR on the USPTO-50k test set without charges and without stereochemical information, following the evaluation setup of Igashov et al. (2024)

Model	$\mathbf{k = 1} \uparrow$	$\mathbf{k = 3} \uparrow$	$\mathbf{k = 5} \uparrow$	$\mathbf{k = 10} \uparrow$	$\widehat{\mathbf{MRR}} \uparrow$
RetroBridge (T=500)	50.8	74.1	**80.6**	**85.6**	0.622
DiffAlign-PE+Skip (T=100)	**56.3**	**75.2**	80.4	84.0	**0.658**

2021). Efforts to bridge the gap between the template-based and template-free paradigms include methods investigating pretraining (Zhong et al., 2022; Jiang et al., 2023). Note that our method is in between template-free and template-based methods, since it uses atom-mapping information but not an explicit (and thus limited) set of templates. We put our models in their own category in the results table.

Methods with a different evaluation procedure Despite Igashov et al. (2024)'s RetroBridge being a closely related to a diffusion model, we cannot include it in a straightforward comparison because it discards atom charges from the ground truth smiles during evaluation. Specifically, the model uses only atom types as node features and compares the generated samples to the smiles reconstructed from the ground truth data through the same encoding (i.e., without charges too) [2]. To compare our model to RetroBridge fairly, we trained our DiffAlign-PE+Skip on uncharged atom-labels, and evaluated it without considering atom charge nor stereochemistry. The results are shown in App. D. Many recently published methods for retrosynthesis have capped out their top-1 scores in the 49%-53% range (see our Table 2), indicating that the increase we achieve over RetroBridge's top-1 score (of over 5%) is quite significant. Furthermore, the results we achieve are with 5 times less denoising steps (100 instead of 500). For a closer look at the performance at different step counts, we invite the reader to compare Figure 5 in (Igashov et al., 2024) to our Figure Fig. 6. The comparison shows that Retrobridge's top-1 score decreases to almost 0 at a step count more than 10. In contrast, our top-1 is close to 50% even at 5 steps, and higher than Retrobridge at 100 steps. We attribute this improvement to our more careful analysis of alignment, and utilising multiple alignment techniques.
.

Methods with pretraining Zhong et al. (2022) and Jiang et al. (2023) pre-train their models on the USPTO-Full and Pistachio data sets, respectively, and as such the results are not directly comparable to models trained on the standard USPTO-50k benchmark. Pretraining with diffusion models is an interesting direction for future research, but we consider it outside the scope of our work. Furthermore, comparison between models with different pretraining datasets and pretraining strategies has the danger of complicating comparisons, given that relative increases in performance could be explained by the model, the pretraining strategy, or the pretraining dataset. As such, we believe that standardized benchmarks like USPTO-50k are necessary when researching modelling strategies.

Another commonly used metric used to evaluate retrosynthesis model is round-trip accuracy, in which a forward prediction model is used to evaluate whether our samples can indeed produce the input product. *Coverage* considers a sample correct if it matches exactly the ground truth or is deemed suitable by the forward prediction oracle. *Accuracy* counts the percentage of valid precur-

[2]This can be seen in the code shared by Igashov et al. (2024): https://github.com/igashov/RetroBridge

Table A5: Round-trip coverage and accuracy for different baselines. The methods are categorized as: template-based (TB), non-template-based (NTB), and models performing their evaluation without taking into account formal charges (NC) nor stereochemistry (NS). We achieve the highest coverage and top-1 accuracy among non-template-based methods. Our lower top-k>1 accuracy may be due to our model generating a higher number of unique predictions compared to competitors, visualized in Figure Fig. A12.

		Coverage (↑)			Accuracy (↑)		
	Model	$k = 1$	$k = 3$	$k = 5$	$k = 1$	$k = 3$	$k = 5$
TB	GLN (Dai et al., 2019)	**82.5**	92.0	94.0	**82.5**	**71.0**	66.2
NTB	MEGAN (Sacha et al., 2021)	78.1	88.6	91.3	78.1	67.3	61.7
	Graph2SMILES (Tu & Coley, 2022)	—	—	—	76.7	56.0	46.4
	Retroformer$_{\text{aug}}$ (Wan et al., 2022)	—	—	—	78.6	**71.8**	**67.1**
	DiffAlign-PE+skip (ours)	**81.6**	**90.0**	**91.8**	**81.6**	62.8	53.3
NS	LocalRetro (Chen & Jung, 2021)	82.1	92.3	94.7	82.1	71.0	66.7
NS-NC	RetroBridge (Igashov et al., 2024)	85.1	95.7	97.1	85.1	**73.6**	**67.8**
	DiffAlign-PE+skip-NSNC (ours)	**87.6**	**96.4**	**97.6**	**87.6**	69.3	59.1

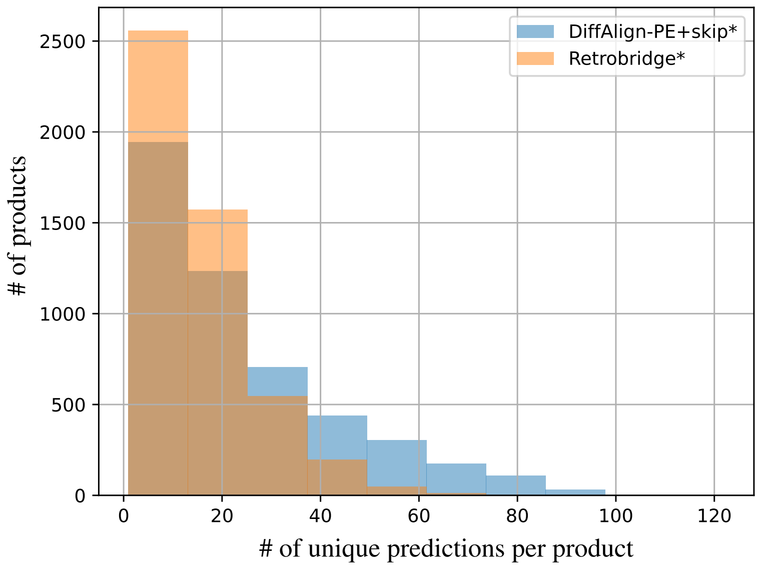


Figure A12: We compare the diversity (number of unique predictions per product) of our samples to RetroBridge (Igashov et al., 2024). It is important for a retrosynthetic model to generate diverse precursors, thus offering the practitioner with multiple synthesis strategies, especially given the limited accuracy of forward prediction models.

sors over all generated samples. We use the same prediction model as previous work, i.e., Molecular Transformer (Schwaller et al., 2019a). As can be seen from Table A5 our model outperforms all non-template-based baselines on all thresholds. We also highlight a known tradeoff between diversity and accuracy of generation in Fig. A12, which explains partly why our accuracy is lower than other baselines (we compare to samples from RetroBridge (Igashov et al., 2024) in particular as an illustration).

E Additional ablations

Disentangling the effect of the inductive biases due to graph positional encodings vs. the inductive bias of alignment. To disentangle the effect of the graph inductive bias brought by the graph

Model	Top-1	Top-3	Top-5	Top-10	Top-50	MRR
DiffAlign-PE-Gaussian	45.51	68.55	73.24	75.78	83.20	0.5649
DiffAlign-PE-Laplacian	46.09	68.75	72.85	75.98	83.98	0.5686

Table A6: Performance comparison of the model with the only alignment method being the matched Gaussian noise encoding versus a model with the only alignment method being the matched Laplacian eigenvector-based positional encoding. Both contain alignment information, but the first does not contain additional inductive biases about the structure of the conditioning graph $\mathbf{Y}$. Results are obtained with 512 samples form the USPTO-50k test set and 10% of the full training budget used in Table 2.

Model	Top-1	Top-3	Top-5	Top-10	Top-50	MRR
Uniform-PE+Skip	43.75	58.98	62.89	67.38	72.66	0.5156
Marginal-PE+Skip	44.73	60.94	63.28	66.60	69.92	0.5244
Absorbing-PE+Skip	**49.22**	**70.90**	**74.61**	**77.93**	**84.96**	**0.5952**

Table A7: Performance comparison of the PE+Skip models with different transition matrices, evaluated with 512 samples form the USPTO-50k test set and 10% of the full training budget.

Laplacian positional encodings and the alignment inductive bias brought by matching the positional encodings for the input and output graph, we trained two models:

1. A model where the only alignment method is the matched PE where the PE is generated from the graph Laplacian eigenvectors of the conditioning graph Y (our default PE method).

2. A model where the only alignment method is matched PEs where the PE is generated by sampling random Gaussian noise vectors on the conditioning graph nodes Y and placing these same vectors on the X side.

Both contain alignment information, but the latter does not contain the additional inductive bias brought by the graph positional encoding. We show the results at roughly 10% of the full training budget for the results in Table 2. The results are listed in Table A6. Clearly, the results are not significantly different, indicating that the inductive bias brought by the graph positional encodings is not as significant as the inductive bias of alignment.

Different transitions. (Vignac et al., 2023) did not originally consider the absorbing state transitions we use. Instead, they used the uniform and marginal transitions. With the uniform transitions, each step provides a non-zero chance of transitioning from one state to any of the others. This converges to an uniform distribution over the node and edge states at $\mathbf{X}_T$. With the marginal transitions, these transitions are biased such that the distribution of nodes and edges in $p(\mathbf{X}_T)$ is the marginal distribution of the node and edge types in the original data. Since there are much more no-edges than edges in the data, this encodes a sparsity to the graphs in $p(\mathbf{X}_T)$.

To provide a fair comparison, we compare our absorbing state model and these models on roughly 10% of the full training budget (80 epochs), and list the results for 512 samples from the test set in Table A7. The absorbing-state model performs the best, whereas the marginal and uniform models perform slightly worse. The marginal transitions somewhat outperform the uniform transitions.

Comparison to only using the skip connections. Here, we provide a comparison to a model that only uses the skip connections, and no other alignment method. The results are again shown for roughly 10% of the training budget (epoch (80) for the PE+Skip model and for the Skip-only model in Table A8. It is evident that only having the skip connection does not work well, although it is better than an entirely unaligned model (see Table 2). This is likely due to the skip-only model having limited expressivity: The neural net does not get the alignment information in the input at all, and thus is unable to use it apart from the skip connection, which enables literally copying the Y graph from input to output.

Model	Top-1	Top-3	Top-5	Top-10	Top-50	MRR
DiffAlign-Skip	9.77	17.38	24.61	29.30	32.23	0.1517
DiffAlign-PE+Skip (Laplacian)	**49.22**	**70.90**	**74.61**	**77.93**	**84.96**	**0.5952**

Table A8: Performance comparison for a model with the only alignment method being skip connections and a model where we also use the Laplacian positional encodings, evaluated with 512 samples form the USPTO-50k test set and 10% of the full training budget

F Adding Post-Training Conditioning to Discrete Diffusion Models

In this section, we show a method to add additional controls and conditions on the used discrete diffusion model post-training. Note that this approach differs from (Vignac et al., 2023)'s in that our method is an adaptation of reconstruction guidance (Ho et al., 2022; Chung et al., 2023; Song et al., 2023a), while they adapt classifier-guidance (Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021; Song et al., 2021) for their conditional model. While the notation is from the point of view of our graph-to-graph translation, the method here applies in general to any discrete diffusion model. We start out by following (Vignac et al., 2023) and (Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021), and then make the novel connection to reconstruction guidance. We write the Bayes' rule for an additional condition y (e.g., a specified level of drug-likeness or synthesizability, or an inpainting mask)

$$p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, y) \propto p(y \,|\, \mathbf{X}_{t-1}, \mathbf{X}_t, \mathbf{Y}) p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) \tag{103}$$

$$= p(y \,|\, \mathbf{X}_{t-1}, \mathbf{Y}) p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) \tag{104}$$

where the second equation was due to the Markovian structure of the generative process ($\mathbf{X}_{t-1}$ d-separates y and $\mathbf{X}_t$). Now, we can take the log and interpret the probabilities as tensors $\mathbf{P}_\theta(y \,|\, \mathbf{X}_{t-1}, \mathbf{Y})$ and $\mathbf{P}_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y})$ defined in the same space as the one-hot valued tensors $\mathbf{X}_{t-1}$ and $\mathbf{X}_t$. We get:

$$\log \mathbf{P}_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, y) \propto \log \mathbf{P}(y \,|\, \mathbf{X}_{t-1}, \mathbf{Y}) + \log \mathbf{P}_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) \tag{105}$$

Similarly to (Vignac et al., 2023), we can now Taylor expand $\log \mathbf{P}_\theta(y \,|\, \mathbf{X}_{t-1}, \mathbf{Y})$ around $\mathbf{X}_t$ with

$$\log \mathbf{P}(y \,|\, \mathbf{X}_{t-1}, \mathbf{Y}) \approx \log \mathbf{P}(y \,|\, \mathbf{X}_t, \mathbf{Y}) + \nabla_{\mathbf{X}'_t} \log \mathbf{P}(y \,|\, \mathbf{X}'_t, \mathbf{Y})|_{\mathbf{X}'_t = \mathbf{X}_t} (\mathbf{X}_{t-1} - \mathbf{X}_t) \tag{106}$$

Given that we are interested in the distribution w.r.t. $\mathbf{X}_{t-1}$, the $\mathbf{X}_t$ terms are constant when we plug them in to Eq. (105), resulting in

$$\log \mathbf{P}_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, y) \propto \nabla_{\mathbf{X}'_t} \log \mathbf{P}(y \,|\, \mathbf{X}'_t, \mathbf{Y})|_{\mathbf{X}'_t = \mathbf{X}_t} \mathbf{X}_{t-1} + \log \mathbf{P}_\theta(\mathbf{X}_{t-1} | \mathbf{X}_t, \mathbf{Y}) \tag{107}$$

Simplifying the notation to assume taking the gradient at $\mathbf{X}_t$, we can also write

$$\log \mathbf{P}_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, y) \propto \nabla_{\mathbf{X}_t} \log \mathbf{P}_\theta(y \,|\, \mathbf{X}_t, \mathbf{Y}) \mathbf{X}_{t-1} + \log \mathbf{P}_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}) \tag{108}$$

In practice, the equation means that given $\log p_\theta(y \,|\, \mathbf{X}_t, \mathbf{Y})$, we get $p_\theta(\mathbf{X}_{t-1} \,|\, \mathbf{X}_t, \mathbf{Y}, y)$ by adding the input gradient of $\log p_\theta(y \,|\, \mathbf{X}_t, \mathbf{Y})$ to the logits given by the regular reverse transition and renormalizing.

It would be possible to approximate $\log p_\theta(y \,|\, \mathbf{X}_t, \mathbf{Y})$ by training an additional classifier, leading to classifier guidance (Sohl-Dickstein et al., 2015; Dhariwal & Nichol, 2021; Song et al., 2021) and the exact method presented in (Vignac et al., 2023).We go further by adapting these formulas to reconstruction guidance (Ho et al., 2022; Chung et al., 2023; Song et al., 2023a). These methods and more advanced versions (Dou & Song, 2024; Finzi et al., 2023; Boys et al., 2023; Wu et al., 2023; Peng et al., 2024) provide different levels of approximations of the true conditional distribution. Here, we show an approach particularly similar to (Chung et al., 2023), by approximating the true denoising distribution $p(\mathbf{X}_0 | \mathbf{X}_t, \mathbf{Y})$ directly with the denoiser output

$$p(y \,|\, \mathbf{X}_t, \mathbf{Y}) = \sum_{\mathbf{X}_0} p(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y}) p(y \,|\, \mathbf{X}_0) \tag{109}$$

$$\approx \sum_{\mathbf{X}_0} \tilde{p}_\theta(\mathbf{X}_0 \,|\, \mathbf{X}_t, \mathbf{Y}) p(y \,|\, \mathbf{X}_0). \tag{110}$$

Algorithm 3 Sampling with atom-count guidance

```
Input: Product $\mathbf{Y}$
Choose: $P^{\mathbf{Y}\to\mathbf{X}} \in \mathbb{R}^{N_\mathbf{X} \times N_\mathbf{Y}}$
$\mathbf{X}_T \propto p(\mathbf{X}_T)$
for $t = T$ to $1$ do
    $\tilde{\mathbf{X}}_0 = D_\theta(\mathbf{X}_t, \mathbf{Y}, P^{\mathbf{Y}\to\mathbf{X}})$    ▷ Denoising output
    $\tilde{\mathbf{X}}^i_{t-1} = \sum_k q(\mathbf{X}^i_{t-1} \mid \mathbf{X}^i_t, \mathbf{X}^i_0)\tilde{\mathbf{X}}^i_0$    ▷ Regular reverse transition probabilities
    $\mathbf{X}^i_{t-1} \sim \text{softmax}(\log \tilde{\mathbf{X}}_{t-1} + \gamma \nabla_{\mathbf{X}_t} \log \sigma(\frac{\sum_{i\in S} \tilde{\mathbf{X}}_{0,i,d} - a}{b}))$    ▷ Renormalize
Return $\mathbf{X}_0$
```

Here $p(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})$ is the true, intractable distribution given by going through the entire sampling process and $\tilde{p}(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})$ is the factorized distribution that is given by the denoiser output, and 'jumps to' X_0 directly. This results in the following update step:

$$\log \mathbf{P}_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}, y) \propto \nabla_{\mathbf{X}_t} \log \left(\mathbb{E}_{\tilde{p}_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})} p(y \mid \mathbf{X}_0)\right) \mathbf{X}_{t-1} + \log \mathbf{P}_\theta(\mathbf{X}_{t-1} \mid \mathbf{X}_t, \mathbf{Y}). \tag{111}$$

Summing over all possible graphs $\mathbf{X}_0$ is prohibitive, however. Instead, we could sample $\mathbf{X}_0$ from $\tilde{p}_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})$ with the Gumbel-Softmax trick (Jang et al., 2016) and evaluate $\log p(y \mid \mathbf{X}_0)$. As long as $\log p(y \mid \mathbf{X}_0)$ is differentiable, we can then just use automatic differentiation to get our estimate of $\nabla_{\mathbf{X}_t} \log \mathbf{P}_\theta(y \mid \mathbf{X}_t, \mathbf{Y})$. Another, more simplified approach that avoids sampling from $\tilde{p}_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})$ is to relax the definition of the likelihood function to directly condition on the continuous-valued probability vector $\tilde{\mathbf{P}}_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})$ instead of the discrete-valued $\mathbf{X}_0$. For simplicity, we adopted this approach, but the full method with Gumbel-Softmax is not significantly more difficult to implement. The algorithm for sampling is shown in Alg. 3.

Toy Synthesisability Model: Controlling Atom Economy

The model of synthesizability that we use is

$$\tilde{\mathbf{X}}_0 = p_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y}), \tag{112}$$

$$p(y = \text{synthesizable} \mid \tilde{\mathbf{X}}_0) = \sigma(\frac{\sum_{i\in S} \tilde{\mathbf{X}}_{0,i,d} - a}{b})^\gamma, \tag{113}$$

where S is the set of non-atom-mapped nodes, a, b and γ are constants that define the synthesizability model and d refers to the dummy node index. The intuition is that the more nodes are classified as dummy nodes (non-atoms), the fewer atoms we have in total, leaving the atom economy higher. Note that $\sum_{i\in S} \tilde{\mathbf{X}}_{0,i,d}$ is the expected amount of dummy nodes from $p_\theta(\mathbf{X}_0 \mid \mathbf{X}_t, \mathbf{Y})$. We set a to half the amount of dummy nodes and b to one-quarter of the amount of dummy nodes. It turns out that this leaves γ as a useful parameter to tune the sharpness of the conditioning. The gradient estimate is then given by

$$\nabla_{\mathbf{X}_t} \log \mathbf{P}_\theta(y \mid \mathbf{X}_t, \mathbf{Y}) = \gamma \nabla_{\mathbf{X}_t} \log \sigma(\frac{\sum_{i\in S} \tilde{\mathbf{X}}_{0,i,d} - a}{b}), \tag{114}$$

which can be directly calculated with automatic differentiation.

G Handling noisy node mappings for the toy data

Sec. 9.1 introduced the graph copying as a toy data set. Here, we extend it to analyse the effect of imperfect node mapping during training. In particular, we consider a scheme for adding 'noise' to the node mapping by swapping them with each other on either the conditioning, or equivalently, the other side. We do it as follows:

1. Sample the number of pair swaps: $f \sim \text{Poisson}(\lambda)$
2. For each of the f swaps, randomly select two nodes i, j in the graph A and swap their mappings

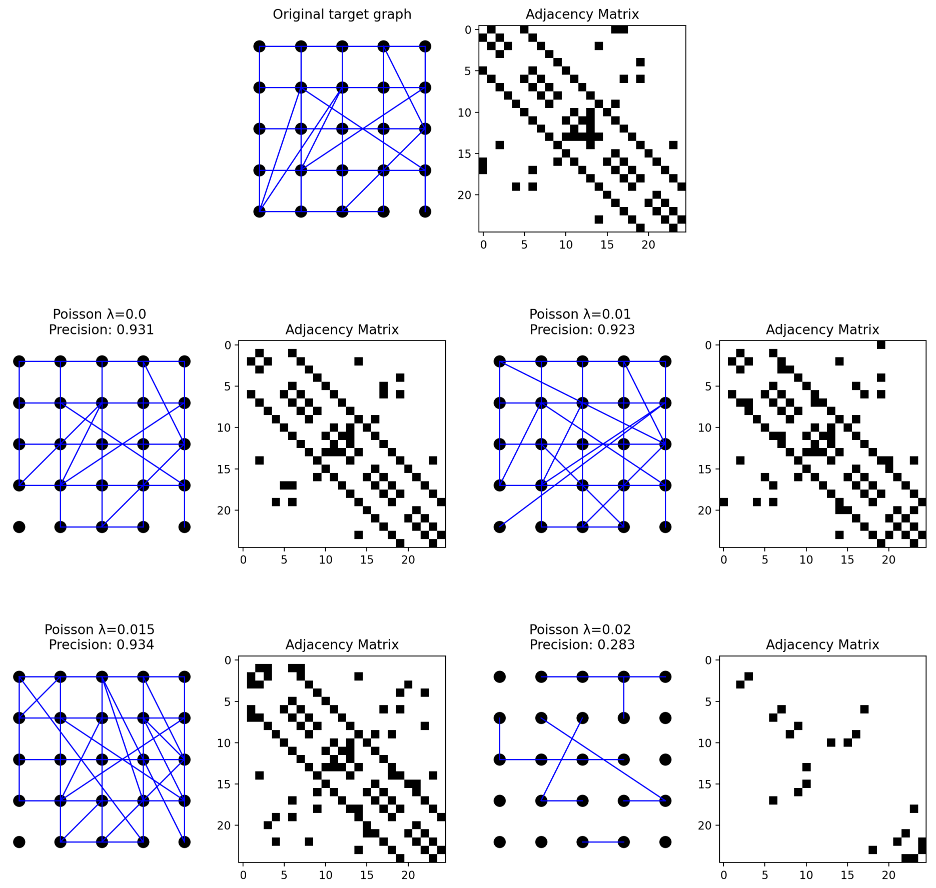


Figure A13: Results from training an aligned model for a 1000 steps in the graph copying task presented in Sec. 9.1, with different levels of noise in the atom mappings. λ parameterizes a Poisson distribution that defines the level of mistakes in the node mapping. On lower noise levels, the results do not significantly change. However, there seems to be a limit in which the neural net training dynamics changes to produce clearly suboptimal results.

The Poisson parameter λ controls the expected number of swaps, allowing us to tune the level of noise in the node mapping. We trained small models for a 1000 steps of training, and show the results with different noise levels in Fig. A13. Small errors in the node mapping are evidently not a significant issue, but larger λ does start affecting performance, at least in this early training phase. We also include a measure of similarity to the target graph: The precision is the fraction of edges that were correctly inferred, compared to the ground-truth target.

H Details for the ibuprofen synthesis experiment

Below we explain the synthesis steps visualized in Fig. 7 in detail:

1. The retrosynthesis begins with carbonylation, adding a 'CO' structure. Initial basic generation yields unpromising results, so the practitioner suggests a partial reactant structure, leading to a more viable path.
2. The model then proposes hydrogenation, a logical next step. While H2 molecules aren't explicitly represented in the data, they're inferred from the context of the reaction.
3. For the third step, the practitioner identifies an opportunity for acylation (involving the `C(=O)C` group), potentially leading to readily available reactants. Given this `C(=O)C`

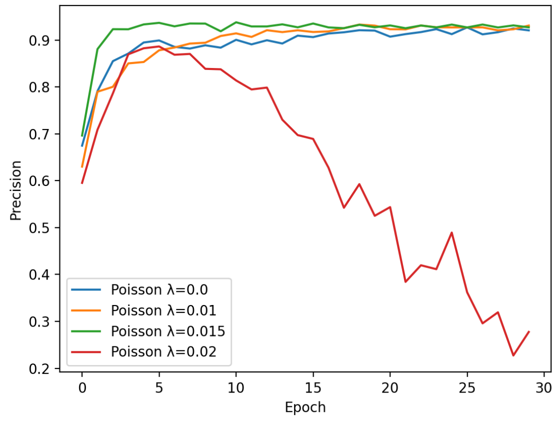


Figure A14: The precision of the adjacency matrix estimates from an aligned model across training, with different levels of node mapping noise λ. Going above $\lambda > 0.02$ nudges the training dynamics to perform poorly on this task.

structure, the model successfully completes the reaction. These steps align with the synthesis plan proposed by BHC.

4. The retrosynthesis path starts with carbonylation, a well-known synthetic reaction which adds a 'CO' structure to a compound. The practitioner tries basic generation but then notices that the suggested reactant is not promising. They then suggest a partial structure of the reactants which leads to a more sensible path.
5. Next, our model proposes hydrogenation, which is a sensible suggestion in this case. The data does not handle explicitly individual `H2` molecules, but they are inferred from the context.
6. In the third step, the practitioner notices that an acylation reaction (a reaction with the `C(=O)C` group) might lead to reactants that are readily available. The model is able to complete the rest of the reaction after knowing that C(=O)C is present. These steps match the synthesis plan proposed by BHC.