Equivariant 3D-Conditional Diffusion Models for Molecular Linker Design

Anonymous Author(s)
Affiliation
Address
email

Abstract

Fragment-based drug discovery has been an effective paradigm in early-stage drug development. An open challenge in this area is designing linkers between disconnected molecular fragments of interest to obtain chemically-relevant candidate drug molecules. In this work, we propose DiffLinker, an E(3)-equivariant 3D-conditional diffusion model for molecular linker design. Given a set of disconnected fragments, our model places missing atoms in between and designs a molecule incorporating all the initial fragments. Unlike previous approaches that are only able to connect pairs of molecular fragments, our method can link an arbitrary number of fragments. Additionally, the model automatically determines the number of atoms in the linker and its attachment points to the input fragments. We demonstrate that DiffLinker outperforms other methods on the standard datasets generating more diverse and synthetically-accessible molecules. Besides, we experimentally test our method in real-world applications, showing that it can successfully generate valid linkers conditioned on target protein pockets.

1 Introduction

The space of pharmacologically-relevant molecules is estimated to exceed 10^{60} structures [53], and searching in that space poses significant challenges for drug design. A successful approach to reduce the size of this space is to start from *fragments*, smaller molecular compounds that usually have no more than 20 heavy (non-hydrogen) atoms. This strategy is known as fragment-based drug design (FBDD) [12]. Given a protein pocket (a part of the target protein that employs suitable properties for binding a ligand), computationally determining fragments that interact with the pocket is a cheaper and more efficient alternative to experimental high-throughput screening methods [12]. Once the relevant fragments have been identified and docked to the target protein, it remains to combine them into a single, connected molecule. Among various strategies such as fragment linking, merging, and growing [31], the former has been preferred as it allows to boost rapidly the binding energy of the target and the compound [27, 16]. This work addresses the fragment linking problem.

Early computational methods for molecular linker design were based on database search and physical simulations [44], both of which are computationally intensive. Therefore, there is increasing interest for machine learning methods that can go beyond the available data and design diverse linkers more efficiently. Existing approaches are either based on syntactic pattern recognition [57] or on autoregressive models [24, 25, 21]. While the former method operates solely on SMILES [54], the latter take into account 3D positions and orientations of the input fragments. However, these methods are not equivariant with respect to the permutation of atoms and can only combine pairs of fragments.

Submitted to NeurIPS 2021 AI for Science Workshop.

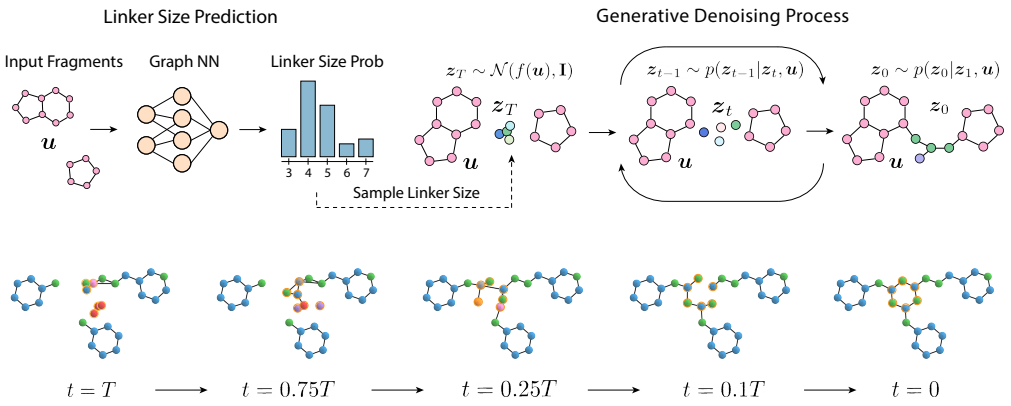


Figure 1: (Top) Overview of the molecular linker generation process. First, probabilities of linker sizes are computed for the input fragments. Next, linker atoms are sampled and denoised using our fragment-conditioned equivariant diffusion model. (Bottom) Example of the linker generation process. Linker atoms are highlighted in orange.

Linker design depends on the target protein pocket, and using this information correctly can improve the affinity of the resulting overall compound. To date, however, there is no computational method for molecular linker design that takes the pocket into account.

In this work, we introduce DiffLinker, a conditional diffusion model that generates molecular linkers for a set of input fragments represented as a 3D atomic point cloud. First, our model generates the size of the prospective linker and then samples initial linker atom types and positions from the normal distribution. Next, the linker atom types and coordinates are iteratively updated using a neural network that is conditioned on the input fragments. Eventually, the denoised linker atoms and the input fragment atoms form a single connected molecule, as shown in Figure 1.

DiffLinker enjoys several desirable properties: it is equivariant to translations, rotations, reflections and permutations, it is not limited by the number of input fragments, it does not require information on the attachment atoms and generates linkers with no predefined size. Besides, the conditioning mechanism of DiffLinker allows to pass additional information about the surrounding protein pocket atoms, which makes the model applicable in structure-based drug design applications [7].

We empirically show that DiffLinker is more effective than previous methods in generating chemically-relevant linkers between pairs of fragments. Our method achieves the state-of-the-art results in synthetic accessibility and drug-likeness which makes it preferable for using in drug design pipelines. Besides, DiffLinker remarkably outperforms other methods in the diversity of the generated linkers. We further propose a more challenging benchmark and show that our method is able to successfully link more than two fragments, which cannot be done by the other methods. We also demonstrate that DiffLinker can be conditioned on the target protein pocket: our model respects geometric constraints imposed by the surrounding protein atoms and generates molecules that have minimum clashes with the corresponding pockets. To the best of our knowledge, DiffLinker is the first method that is not limited by the number of input fragments and accounts the information about pockets. The overall goal of this work is to provide practitioners with an effective tool for molecular linker generation in realistic drug design scenarios.

2 Related Work

Molecular linker design has been widely used in the fragment-based drug discovery community [44]. Various de novo design methods refer to the fragment linking problem [52, 36, 41, 38, 48, 50, 22]. Early fragment linking methods were based on search in the predefined libraries of linkers [6, 34],

genetic algorithm, tabu search [14] and force field optimization [9]. Having been successfully used in multiple application cases [28, 46, 43], these methods are however computationally expensive and substantially limited by the available data.

Hence, there has recently been interest in developing learning-based methods for molecular linker design. Yang et al. [57] proposed SyntaLinker, a SMILES-based deep conditional transformer neural network that solves a sentence completion problem [58]. This method inherits the drawbacks of SMILES, which are the absence of 3D structure and the lack of consistency (atoms that are close in the molecule can be far away in the SMILES string). Imrie et al. [24] overcome these limitations by introducing an autoregressive model DeLinker and its extension DEVELOP [25] that uses additional pharmacophore information. Although these methods operate on 3D molecular conformations, they use a very limited geometric information and require input on the attachment atoms of the fragments. Recently, Huang et al. [21] have proposed another autoregressive method 3DLinker that does not require one to specify attachment points and leverages the geometric information to a much greater extent. It makes this approach more relevant for connecting docked fragments. As both DeLinker and 3DLinker are autoregressive models, they are not permutation equivariant which limits their sample efficiency and ability to scale to large molecules [11, 40]. Besides, these methods are capable of connecting only pairs of fragments and cannot be easily extended to larger sets of fragments.

Outside of the linker design problem, several recent works proposed denoising diffusion models for molecular data in 3D. Conformer generation methods GeoDiff [56] and ConfGF [45] condition the model on the adjacency matrix of the molecular graph. Since they have access to the connectivity information, they can compute torsion angles between atoms and optimize them [29]. Equivariant Diffusion Model (EDM) [19] generates 3D molecules from scratch and is able to be conditioned on the predefined scalar properties. Another diffusion model has been recently proposed for designing protein scaffolds given protein motifs [51]. Having the conditional sampling procedure, this model is however trained in an unconditional setup. Finally, Luo et al. [35] proposed a model for antibody design which combines discrete diffusion for the molecular graphs and continuous diffusion on the 3D coordinates. The conditioning mechanism proposed in this work is the closest to ours, but their model is targeted at generating chains of amino acids rather than atomic point clouds.

3 Preliminaries

Diffusion Models Diffusion models [47] are a class of generative methods that consist of two components. The first component, a *diffusion process*, progressively distorts a data point $\boldsymbol{x}$ mapping it to a noise. The probability of any intermediate state $\boldsymbol{z}_t$ for $t = 0, \dots, T$ is defined as $q(\boldsymbol{z}_t|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}_t; \alpha_t \boldsymbol{x}, \sigma_t^2 \boldsymbol{I})$, where $\alpha_t \in \mathbb{R}^+$ controls how much signal is retained and $\sigma_t \in \mathbb{R}^+$ controls how much noise is added. Transition probabilities are given by $q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(\boldsymbol{z}_t; \overline{\alpha}_t \boldsymbol{z}_{t-1}, \overline{\sigma}_t^2 \boldsymbol{I})$, where $\overline{\alpha}_t = \alpha_t / \alpha_{t-1}$ and $\overline{\sigma}_t^2 = \sigma_t^2 - \overline{\alpha}_t^2 \sigma_{t-1}^2$. The reversed *true denoising* process that reconstructs the data point from noise is defined by $q(\boldsymbol{z}_{t-1}|\boldsymbol{x}, \boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{z}_{t-1}; \boldsymbol{\mu}_t(\boldsymbol{x}, \boldsymbol{z}_t), \varsigma_t^2 \boldsymbol{I})$ with distribution parameters $\boldsymbol{\mu}_t(\boldsymbol{x}, \boldsymbol{z}_t)$ and ς_t can be derived analytically and are provided in Appendix A.1.

The second component of a diffusion model is a *generative denoising process* that learns to invert the diffusion trajectory having the data point $\boldsymbol{x}$ unknown. The generative transition distribution is defined as $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = q(\boldsymbol{z}_{t-1}|\hat{\boldsymbol{x}}, \boldsymbol{z}_t)$, where $\hat{\boldsymbol{x}} = (1/\alpha_t)\boldsymbol{z}_t - (\sigma_t/\alpha_t)\hat{\boldsymbol{\epsilon}}_t$ is an approximation of the data point $\boldsymbol{x}$, and $\hat{\boldsymbol{\epsilon}}_t$ is a prediction of the Gaussian noise computed by a neural network φ [17].

To generate a new data point, one first samples the Gaussian noise: $\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I})$. Then, for $t = T, \dots, 1$, one should iteratively sample $\boldsymbol{z}_{t-1} \sim p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ and finally sample $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}_0)$.

For training the neural network φ, we use the objective $\mathcal{L}(t) = ||\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_t||^2$ that can be optimized by mini-batch gradient descent using an estimator $\mathbb{E}_{t \sim \mathcal{U}(0,\dots,T)}[T \cdot \mathcal{L}(t)]$ [17, 30].

Molecule Representation We consider now diffusion models that can operate on molecules represented as 3D atomic point clouds. A data point $\boldsymbol{x}$, which is an attributed point cloud consisting of M atoms, is represented by atom coordinates $\boldsymbol{r} = (\boldsymbol{r}_1, \dots, \boldsymbol{r}_M) \in \mathbb{R}^{M \times 3}$ and the corresponding

feature vectors $\boldsymbol{h} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_M) \in \mathbb{R}^{M \times \text{nf}}$ which are one-hot encoded atom types. We will therefore denote point cloud $\boldsymbol{x}$ as a tuple $\boldsymbol{x} = [\boldsymbol{r}, \boldsymbol{h}]$.

Categorical Features Along with continuous atom coordinates, a molecular diffusion model has to operate on atom types that are discrete variables. While categorical diffusion models do exist [18, 2], we follow a simpler strategy [19] that is based on lifting the atom types to a continuous space: we consider a one-hot encoding of the discrete variables, and add Gaussian noise on top of it. In the end of the denoising process, once $\boldsymbol{z}_0$ is sampled, the continuous values corresponding to the atom types should be converted back to discrete values. We consider the argmax over the different categories and include it in the final transition from $\boldsymbol{z}_0$ to $\boldsymbol{x}$. For more details on the structure of the final transition distribution $p(\boldsymbol{x}|\boldsymbol{z}_0)$ and likelihood computation in this setting, we refer the reader to [19].

Equivariance Processing 3D molecules requires operations that respect data symmetries. In this work, we consider the Euclidean group E(3) that comprises translations, rotations and reflections of $\mathbb{R}^3$ and the orthogonal group O(3) that includes rotations and reflections of $\mathbb{R}^3$. A function $f : \mathbb{R}^3 \to \mathbb{R}^3$ is E(3)-equivariant if $f(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}) = \boldsymbol{R}f(\boldsymbol{x}) + \boldsymbol{t}$ for any orthogonal matrix $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$, $\det \boldsymbol{R} = \pm 1$, for any translation vector $\boldsymbol{t} \in \mathbb{R}^3$ and for any $\boldsymbol{x} \in \mathbb{R}^3$. Note that for simplicity we use notation $\boldsymbol{R}\boldsymbol{x} = (\boldsymbol{R}\boldsymbol{x}_1, \ldots, \boldsymbol{R}\boldsymbol{x}_M)^\top$. A conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$ is E(3)-equivariant if $p(\boldsymbol{R}\boldsymbol{x} + \boldsymbol{t}|\boldsymbol{R}\boldsymbol{y} + \boldsymbol{t}) = p(\boldsymbol{x}|\boldsymbol{y})$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$. A function f and a distribution p are O(3)-equivariant if $f(\boldsymbol{R}\boldsymbol{x}) = \boldsymbol{R}f(\boldsymbol{x})$ and $p(\boldsymbol{R}\boldsymbol{x}|\boldsymbol{R}\boldsymbol{y}) = p(\boldsymbol{x}|\boldsymbol{y})$ respectively. We call the function f translation invariant if $f(\boldsymbol{x} + \boldsymbol{t}) = f(\boldsymbol{x})$.

4 DiffLinker: Equivariant 3D-conditional Diffusion Model for Molecular Linker Design

4.1 Equivariant 3D-Conditional Diffusion Model

Unlike other diffusion models for molecule generation [19, 56], our method is conditioned on 3D data. More specifically, we assume that each point cloud $\boldsymbol{x}$ has a corresponding *context* $\boldsymbol{u}$, which is another point cloud consisting of all input fragments and (optionally) protein pocket atoms that remain unchanged throughout the diffusion and denoising processes, as shown in Figure 1. Hence, we consider the denoising generative process p to operate on point cloud $\boldsymbol{x}$ while being conditioned on the fixed corresponding context:

$$p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u}) = q(\boldsymbol{z}_{t-1}|\hat{\boldsymbol{x}}, \boldsymbol{z}_t), \quad \text{where} \quad \hat{\boldsymbol{x}} = (1/\alpha_t)\boldsymbol{z}_t - (\sigma_t/\alpha_t)\varphi(\boldsymbol{z}_t, \boldsymbol{u}, t). \tag{1}$$

Initial noise $\boldsymbol{z}_T$ is sampled from $p(\boldsymbol{z}_T|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{z}_T; f(\boldsymbol{u}), \boldsymbol{I})$, where function f transforms the point cloud $\boldsymbol{u}$ into a single point that is considered as a mean of the Gaussian noise. The choice of the function f highly depends on the problem being solved and the available priors. In our experiments, we consider two cases. First, following Imrie et al. [24], we make use of the information about atoms that should be connected by the linker. We call these atoms *anchors* and define $f(\boldsymbol{u})$ as the anchors' center of mass. However, in real-world scenarios it is unlikely to know which atoms should be the anchors. In this case we define $f(\boldsymbol{u})$ as the center of mass of the whole context $\boldsymbol{u}$.

As shown in Appendix A.2, the generative distribution $p(\boldsymbol{z}_0|\boldsymbol{u})$ is O(3)-equivariant if both f and φ are O(3)-equivariant. In order to make the generative process also independent of translations, we construct the network φ to be additionally translation invariant. Then, instead of sampling the initial noise from $\mathcal{N}(f(\boldsymbol{u}), \boldsymbol{I})$ we center the data at $f(\boldsymbol{u})$ and sample from $\mathcal{N}(0, \boldsymbol{I})$.

4.2 Equivariant Graph Neural Network

The learnable function φ that models the dynamics of the diffusion model is implemented as a modified E(3)-equivariant graph neural network (EGNN) [42]. Its input is the noisy version of the linker $\boldsymbol{z}_t$ at time t and the context $\boldsymbol{u}$. These two parts are modeled as a single fully-connected graph where nodes are represented by coordinates $\boldsymbol{r}$ and feature vectors $\boldsymbol{h}$ that include atom types, time t, fragment flags and (optionally) anchor flags. The predicted noise $\hat{\boldsymbol{\epsilon}}$ includes coordinate and feature

components: $\hat{\boldsymbol{\epsilon}} = [\hat{\boldsymbol{\epsilon}}^r, \hat{\boldsymbol{\epsilon}}^h]$. In order to make function φ invariant to translations, we subtract the initial coordinates from the coordinate component of the predicted noise following Hoogeboom et al. [19]:

$$\hat{\boldsymbol{\epsilon}} = [\hat{\boldsymbol{\epsilon}}^r, \hat{\boldsymbol{\epsilon}}^h] = \varphi(\boldsymbol{z}_t, \boldsymbol{u}, t) = \text{EGNN}(\boldsymbol{z}_t, \boldsymbol{u}, t) - [\boldsymbol{z}_t^r, 0]. \tag{2}$$

EGNN consists of the composition of Equivariant Graph Convolutional Layers $\boldsymbol{r}^{l+1}, \boldsymbol{h}^{l+1} = \text{EGCL}[\boldsymbol{r}^l, \boldsymbol{h}^l]$ which are defined as follows:

$$\boldsymbol{m}_{ij} = \phi_e(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l, d_{ij}^2), \quad \boldsymbol{h}_i^{l+1} = \phi_h(\boldsymbol{h}_i^l, \sum_{j \neq i} \boldsymbol{m}_{ij}), \quad \boldsymbol{r}_i^{l+1} = \boldsymbol{r}_i^l + \phi_{vel}(\boldsymbol{r}^l, \boldsymbol{h}^l, i), \tag{3}$$

where $d_{ij} = \|\boldsymbol{r}_i^l - \boldsymbol{r}_j^l\|$ and ϕ_e, ϕ_h are learnable functions parametrized by fully connected neural networks (see Appendix A.5 for details).

The latter update for the node coordinates is computed by the learnable function ϕ_{vel}. Note that our graph includes both a noisy linker $\boldsymbol{z}_t$ and a fixed context $\boldsymbol{u}$, and φ is intended to predict the noise that should be subtracted from the coordinates and features of $\boldsymbol{z}_t$. Therefore, it is natural to keep the context coordinates unchanged when computing dynamics and to apply non-zero displacements only to the linker part at each EGCL step. Hence, we model the node displacements as follows,

$$\phi_{vel}(\boldsymbol{r}^l, \boldsymbol{h}^l, i) = \begin{cases} \sum_{j \neq i} \frac{\boldsymbol{r}_i^l - \boldsymbol{r}_j^l}{d_{ij}+1} \phi_r(\boldsymbol{h}_i^l, \boldsymbol{h}_j^l, d_{ij}^2) & \text{if node } i \text{ belongs to the point cloud } \boldsymbol{z}_t, \\ 0 & \text{if node } i \text{ belongs to the context } \boldsymbol{u}, \end{cases} \tag{4}$$

where ϕ_r is a learnable function parametrized by a fully connected neural network.

After the sequence of EGCLs is applied, we have an updated graph with new node coordinates $\hat{\boldsymbol{r}} = [\boldsymbol{u}^r, \hat{\boldsymbol{z}}_t^r]$ and new node features $\hat{\boldsymbol{h}} = [\hat{\boldsymbol{u}}^h, \hat{\boldsymbol{z}}_t^h]$. Since we are interested only in the linker-related part, we discard the coordinates and features of context nodes and consider the tuple $[\hat{\boldsymbol{z}}_t^r, \hat{\boldsymbol{z}}_t^h]$ to be the EGNN output.

4.3 Linker Size Prediction

To predict the size of the missing linker between a set of fragments, we represent fragments as a fully-connected graph with one-hot encoded atom types as node features and distances between nodes as edge features. From this, a separately trained GNN (see Appendix A.6 for details) produces probabilities for the linker size. Our assumption is that relative fragment positions and orientations along with atom types contain all the information essential for predicting most likely size of the prospective linker. When generating a linker, we first sample its size with the predicted probabilities from the categorical distribution over the list of linker sizes seen in the training data, as shown in Figure 1.

4.4 Protein Pocket Conditioning

In real-world fragment-based drug design applications, it often occurs that fragments are selected and docked into a target protein pocket [23]. To propose a drug candidate molecule, the fragments have to be linked. When generating the linker, one should take the surrounding pocket into account and construct a linker that has no clashes with protein pocket atoms (in other words, the configuration of the linker and pocket atoms should be physically-realistic) and keeps the binding strength high. To add pocket conditioning to DiffLinker, we represent a protein pocket as an atomic point cloud and consider it as a part of the context $\boldsymbol{u}$ (cf. Section 4.1). We also extend node features with an additional binary flag marking atoms that belong to the protein pocket. Finally, as the new context point cloud contains much more atoms, we modify the joint representation of the data point $\boldsymbol{z}_t$ and the context $\boldsymbol{u}$ that are passed to the neural network φ. Instead of considering fully-connected graphs, we assign edges between nodes based on a 4 Å distance cutoff as it makes the resulting graphs less dense and counterbalances the increase in the number of nodes.

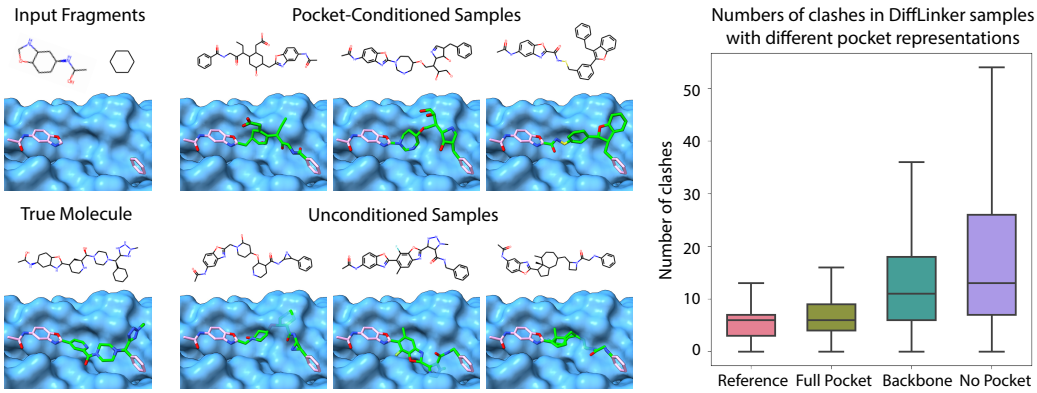


Figure 2: (Left) Examples of linkers sampled by DiffLinker conditioned on pocket atoms (top row) and unconditioned (bottom row). Linkers sampled by the unconditioned model have multiple clashes with the protein pocket. (Right) Distribution of numbers of clashes in reference test molecules and samples from three DiffLinker models differently conditioned (or unconditioned) on pocket atoms.

5 Experiments

5.1 Datasets

ZINC and CASF We follow Imrie et al. [24] and consider a subset of 250,000 molecules randomly selected by Gómez-Bombarelli et al. [15] from ZINC database [26]. First, we generate 3D conformers using RDKit [32] and define a reference 3D structure for each molecule by selecting the lowest energy conformation. Then, these molecules are fragmented by enumerating all double cuts of acyclic single bonds that are not within functional groups. The resulting splits are filtered by the number of atoms in the linker and fragments, synthetic accessibility [13], ring aromaticity, and pan-assay interference compounds (PAINS) [4] criteria. One molecule can therefore result in various combinations of two fragments with a linker in between. The resulting dataset is randomly split into train (438,610 examples), validation (400 examples), and test (400 examples) sets. Another evaluation benchmark used by Imrie et al. [24] is taken from the CASF-2016 dataset [49]. In contrast to ZINC, where molecule conformers were generated computationally, CASF includes experimentally verified 3D conformations. Following the same preprocessing procedures as for the ZINC dataset, Imrie et al. [24] obtained an additional test set of 309 examples, which we use in our work.

GEOM ZINC and CASF datasets used in previous works only contain pairs of fragments. However, real-world applications often require connecting more than two fragments with one or more linkers [23]. To address this case, we construct a new dataset based on GEOM molecules [3] which we decompose into three or more fragments with one or two linkers connecting them. To achieve such splits, we use RDKit implementations of two fragmentation techniques — an MMPA-based algorithm [10] and BRICS [8] — and combine results removing duplicates. Overall, we obtain 41,907 molecules and 285,142 fragmentations that are randomly split in train (282,602 examples), validation (1,250 examples) and test (1,290 examples) sets.

Pockets Dataset In order to assess the ability of DiffLinker to generate valid linkers given additional information about protein pockets, we use the protein-ligand dataset curated by Anonymous [1] from Binding MOAD [20]. To define pockets, we consider amino acids that have at least one atom closer than 6 Å to any atom of the ligand. All atoms belonging to these residues constitute the pocket. We split molecules into fragments using RDKit's implementation of MMPA-based algorithm [10]. We randomly split the resulting data into train (185,678 examples), validation (490 examples) and test (566 examples) sets taking into account Enzyme Commission (EC) numbers of the proteins.

Table 1: Performance metrics on ZINC, CASF and GEOM test sets. The first three metrics assess the chemical relevance of the generated molecules. The last three metrics evaluate the standard generative properties of the methods.

	Method	QED ↑	SA ↓	# Rings ↑	Valid, %	Unique, %	Novel, %
ZINC	DeLinker + ConfVAE + MMFF	0.64	3.11	0.21	**98.3**	44.2	**47.1**
	3DLinker (given anchors)	0.65	3.11	0.23	**99.3**	29.0	41.2
	3DLinker	0.65	3.14	0.24	71.5	29.2	41.9
	DiffLinker	**0.68**	**3.01**	0.25	93.8	24.0	30.3
	DiffLinker (given anchors)	**0.68**	**3.03**	0.26	97.6	22.7	32.4
	DiffLinker (sampled size)	0.65	3.19	**0.32**	90.6	**51.4**	42.9
	DiffLinker (given anchors, sampled size)	0.65	3.24	**0.36**	94.8	**50.9**	**47.7**
CASF	DeLinker + ConfVAE + MMFF	0.35	4.05	0.35	**95.7**	51.6	**55.6**
	DiffLinker	**0.41**	**4.00**	0.34	85.3	40.5	41.8
	DiffLinker (given anchors)	0.40	**4.03**	**0.38**	**90.2**	37.3	48.4
	DiffLinker (sampled size)	**0.40**	4.06	0.30	63.7	**60.0**	49.3
	DiffLinker (given anchors, sampled size)	0.40	4.10	**0.38**	68.4	**57.1**	**56.9**
GEOM	3DLinker	0.36	3.56	0.00	16.3	**73.7**	—
	DiffLinker	**0.48**	**2.99**	0.75	**94.0**	34.7	68.6
	DiffLinker (given anchors)	**0.49**	**3.01**	**0.79**	**94.0**	35.0	68.4
	DiffLinker (sampled size)	0.45	3.27	0.75	87.2	62.4	**76.0**
	DiffLinker (given anchors, sampled size)	0.46	3.33	**0.83**	88.8	**63.6**	**76.0**

5.2 Evaluation

Metrics First, we report several chemical properties of the generated molecules that are especially important in drug design applications: average quantitative estimation of drug-likeness (QED) [5], average synthetic accessibility (SA) [13] and average number of rings in the linker. Next, following Imrie et al. [24], we measure validity, uniqueness and novelty of the samples. We then determine if the generated linkers are consistent with the 2D filters used to produce the ZINC training set. These filters are explained in detail in Appendix A.10. In addition, we record the percentage of the original molecules that were recovered by the generation process. To compare the 3D shapes of the sampled and ground-truth molecules, we estimate the Root Mean Squared Deviation (RMSD) between the generated and real linker coordinates in the cases where true molecules are recovered. We also compute the SC_{RDKit} metric that evaluates the geometric and chemical similarity between the ground-truth and generated molecules [39, 33].

Baselines We compare our method with DeLinker [24] and 3DLinker [21] on the ZINC test set and with DeLinker on the CASF dataset. Besides, we adjust 3DLinker to connect more than two fragments (see Appendix A.7 for details) and evaluate its performane on the GEOM dataset. To obtain 3D conformations for the molecules generated by DeLinker on ZINC and CASF, we apply a pre-trained ConfVAE [55] followed by a force field relaxation procedure. For all methods including ours, we generate linkers with the ground-truth size unless explicitly noted otherwise. To obtain SMILES representations of atomic point clouds generated by our models, we utilize OpenBabel [37] to compute covalent bonds between atoms. We also use OpenBabel to rebuild covalent bonds for the molecules in test sets in order to correctly compute the recovery rate, RMSD and SC_{RDKit} scores for our models. In ZINC and CASF experiments, we sample 250 linkers for each input pair of fragments. For the GEOM dataset and in experiments with pocket conditioning, we sample 100 linkers for each input set of fragments.

5.3 Results

ZINC and CASF While our models have much greater flexibility and applicability in more applications as we show below, they also outperform other methods on standard benchmarks ZINC and CASF in terms of chemical relevance of the generated molecules. As shown in Table 1, molecules sampled by DiffLinker are more synthetically accessible and demonstrate higher drug-likeness, which is especially important for drug design applications. Besides, our models generate linkers containing

more rings. Moreover, our molecules usually share higher chemical and geometric similarity with the reference molecules as demonstrated by the SC_{RDKit} scores in Table 4. In terms of validity, our models perform on par with the other methods. Note that both autoregressive approaches employ valency rules at each generation step explicitly, while our model is shown to be able to learn these rules from the data. Remarkably, the validity of the reference molecules from CASF with covalent bonds computed by OpenBabel is 92.2% while our model generated molecules with 90.2% validity. Notably, sampling the size of the linker significantly improves novelty and uniqueness of the generated linkers without significant degradation of the most important metrics. Examples of linkers generated by DiffLinker for different input fragments are provided in Figure 5.

Multiple Fragments The major advantage of DiffLinker compared to recently proposed autoregressive models DeLinker and 3DLinker is one-shot generation of the whole linker between arbitrary amounts of fragments. This overcomes the limitation of DeLinker and 3DLinker, which can only link two fragments at a time. Although these autoregressive models can be adjusted to connect pairs of fragments iteratively while growing the molecule, the full context cannot be taken into account in this case. Therefore, suboptimal solutions are more likely to be produced. To illustrate this difference, we adapted 3DLinker to iteratively connect pairs of fragments in molecules where more than two fragments should be connected. As shown in Table 1, 3DLinker fails to construct valid molecules in almost 84% of cases and cannot recover any reference molecule while, despite the higher complexity of linkers in this dataset, our models achieve 94% validity and recover more than 50% of the reference molecules. Besides, molecules generated by 3DLinker have no rings in the linkers, have substantially lower QED and are much harder to synthesize. Examples of linkers generated by DiffLinker for different input fragments are provided in Figure 4. An example of the DiffLinker sampling process for the molecule from the GEOM dataset is shown in Figure 1.

Protein Pocket Conditioning To illustrate the ability of DiffLinker to take surrounding pockets into account, we trained three models on the Pockets Dataset: these are respectively conditioned on the full-atomic pocket representation, conditioned on the pocket backbone atoms and unconditioned. Besides the standard metrics reported in Tables 5 and 6, we also compute the number of clashes between generated molecules and surrounding pockets. We say that there is a clash between two atoms if the distance between them is lower than the sum of their van der Waals radii. As shown in Figure 2, the model conditioned on the full-atomic pocket representation generates molecules with almost the same amount of clashes (in average 7 clashes per molecule) as in the reference complexes from the test set (in average 6 clashes per molecule). There is a clear trend on the number of clashes depending on the amount of information about pockets DiffLinker is conditioned on: the model conditioned on pocket backbone atoms generates molecules with 14 clashes in average, and the unconditioned model produces molecules with 21 clashes in average.

6 Conclusion

In this work, we introduced DiffLinker, a new E(3)-equivariant 3D-conditional diffusion model for molecular linker design. DiffLinker designs realistic molecules from a set of disconnected fragments by generating a linker, i.e., an atomic point cloud that interconnects the input fragments. *While previous methods were only capable to connect pairs of fragments, DiffLinker naturally scales to an arbitrary number of fragments.* Our method does not require to specify the attachment points of the fragments and predicts the distribution of linker size from the fragments. We show that the proposed method outperforms other models on standard benchmarks and produces more chemically-relevant molecules. *Besides, we demonstrate that our model can be conditioned on protein pockets and generate linkers with a minimum number of clashes.* We believe that our method will accelerate the development of prospective drug candidates and has the potential to become widely used in the fragment-based drug design community.

References

[1] Anonymous. ICLR 2023 Submission. 2022.

[2] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[3] Simon Axelrod and Rafael Gómez-Bombarelli. Geom, energy-annotated molecular conformations for property prediction and molecular generation. *Scientific Data*, 9(1):185, 2022. doi: 10.1038/s41597-022-01288-4. URL `https://doi.org/10.1038/s41597-022-01288-4`.

[4] Jonathan B Baell and Georgina A Holloway. New substructure filters for removal of pan assay interference compounds (pains) from screening libraries and for their exclusion in bioassays. *Journal of medicinal chemistry*, 53(7):2719–2740, 2010.

[5] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[6] Hans-Joachim Böhm. The computer program ludi: a new method for the de novo design of enzyme inhibitors. *Journal of computer-aided molecular design*, 6(1):61–78, 1992.

[7] Miles Congreve, Christopher W Murray, and Tom L Blundell. Keynote review: Structural biology and drug discovery. *Drug discovery today*, 10(13):895–907, 2005.

[8] Jörg Degen, Christof Wegscheid-Gerlach, Andrea Zaliani, and Matthias Rarey. On the art of compiling and using'drug-like'chemical fragment spaces. *ChemMedChem: Chemistry Enabling Drug Discovery*, 3(10):1503–1507, 2008.

[9] Fabian Dey and Amedeo Caflisch. Fragment-based de novo ligand design by multiobjective evolutionary optimization. *Journal of chemical information and modeling*, 48(3):679–690, 2008.

[10] Alexander G Dossetter, Edward J Griffen, and Andrew G Leach. Matched molecular pair analysis in drug discovery. *Drug Discovery Today*, 18(15-16):724–731, 2013.

[11] Bryn Elesedy and Sheheryar Zaidi. Provably strict generalisation benefit for equivariant models. In *International Conference on Machine Learning*, pp. 2959–2969. PMLR, 2021.

[12] Daniel A Erlanson, Stephen W Fesik, Roderick E Hubbard, Wolfgang Jahnke, and Harren Jhoti. Twenty years on: the impact of fragments on drug discovery. *Nature reviews Drug discovery*, 15(9):605–619, 2016.

[13] 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):1–11, 2009.

[14] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[15] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[16] PJ Hajduk, G Sheppard, DG Nettesheim, ET Olejniczak, SB Shuker, RP Meadows, DH Steinman, GM Carrera, PA Marcotte, J Severin, et al. Discovery of potent nonpeptide inhibitors of stromelysin using sar by nmr. *Journal of the American Chemical Society*, 119(25):5818–5827, 1997.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[18] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

[19] Emiel Hoogeboom, Victor Garcia Satorras, Clément Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pp. 8867–8887. PMLR, 2022.

[20] Liegi Hu, Mark L Benson, Richard D Smith, Michael G Lerner, and Heather A Carlson. Binding moad (mother of all databases). *Proteins: Structure, Function, and Bioinformatics*, 60 (3):333–340, 2005.

[21] Yinan Huang, Xingang Peng, Jianzhu Ma, and Muhan Zhang. 3dlinker: An e (3) equivariant variational autoencoder for molecular linker design. *arXiv preprint arXiv:2205.07309*, 2022.

[22] Osamu Ichihara, John Barker, Richard J Law, and Mark Whittaker. Compound design by fragment-linking. *Molecular Informatics*, 30(4):298–306, 2011.

[23] Ilia Igashov, Arian R Jamasb, Ahmed Sadek, Freyr Sverrisson, Arne Schneuing, Pietro Lio, Tom L Blundell, Michael Bronstein, and Bruno Correia. Decoding surface fingerprints for protein-ligand interactions. *bioRxiv*, 2022.

[24] Fergus Imrie, Anthony R Bradley, Mihaela van der Schaar, and Charlotte M Deane. Deep generative models for 3d linker design. *Journal of chemical information and modeling*, 60(4): 1983–1995, 2020.

[25] Fergus Imrie, Thomas E Hadfield, Anthony R Bradley, and Charlotte M Deane. Deep generative design with 3d pharmacophoric constraints. *Chemical science*, 12(43):14577–14589, 2021.

[26] John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

[27] William P Jencks. On the attribution and additivity of binding energies. *Proceedings of the National Academy of Sciences*, 78(7):4046–4050, 1981.

[28] Haitao Ji, Wannian Zhang, Min Zhang, Makiko Kudo, Yuri Aoyama, Yuzo Yoshida, Chunquan Sheng, Yunlong Song, Song Yang, Youjun Zhou, et al. Structure-based de novo design, synthesis, and biological evaluation of non-azole inhibitors specific for lanosterol 14α-demethylase of fungi. *Journal of medicinal chemistry*, 46(4):474–485, 2003.

[29] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *arXiv preprint arXiv:2206.01729*, 2022.

[30] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

[31] Bas Lamoree and Roderick E Hubbard. Current perspectives in fragment-based lead discovery (fbld). *Essays in biochemistry*, 61(5):453–464, 2017.

[32] Greg Landrum. Rdkit documentation. *Release*, 1(1-79):4, 2013.

[33] Gregory A Landrum, Julie E Penzotti, and Santosh Putta. Feature-map vectors: a new class of informative descriptors for computational drug discovery. *Journal of computer-aided molecular design*, 20(12):751–762, 2006.

[34] Georges Lauri and Paul A Bartlett. Caveat: a program to facilitate the design of organic molecules. *Journal of computer-aided molecular design*, 8(1):51–66, 1994.

[35] Shitong Luo, Yufeng Su, Xingang Peng, Sheng Wang, Jian Peng, and Jianzhu Ma. Antigen-specific antibody design and optimization with diffusion-based generative models. *bioRxiv*, 2022.

[36] Andrew Miranker and Martin Karplus. An automated method for dynamic ligand design. *Proteins: Structure, Function, and Bioinformatics*, 23(4):472–490, 1995.

[37] Noel M O'Boyle, Michael Banck, Craig A James, Chris Morley, Tim Vandermeersch, and Geoffrey R Hutchison. Open babel: An open chemical toolbox. *Journal of cheminformatics*, 3 (1):1–14, 2011.

[38] David A Pearlman and Mark A Murcko. Concerts: dynamic connection of fragments as an approach to de novo ligand design. *Journal of medicinal chemistry*, 39(8):1651–1663, 1996.

[39] Santosh Putta, Gregory A Landrum, and Julie E Penzotti. Conformation mining: an algorithm for finding biologically relevant conformations. *Journal of medicinal chemistry*, 48(9):3313–3318, 2005.

[40] Matthias Rath and Alexandru Paul Condurache. Improving the sample-complexity of deep classification networks with invariant integration. *arXiv preprint arXiv:2202.03967*, 2022.

[41] Diana C Roe and Irwin D Kuntz. Builder v. 2: improving the chemistry of a de novo design strategy. *Journal of Computer-Aided Molecular Design*, 9(3):269–282, 1995.

[42] Victor Garcia Satorras, Emiel Hoogeboom, Fabian B Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows. *arXiv preprint arXiv:2105.09016*, 2021.

[43] Chunquan Sheng and Wannian Zhang. New lead structures in antifungal drug discovery. *Current medicinal chemistry*, 18(5):733–766, 2011.

[44] Chunquan Sheng and Wannian Zhang. Fragment informatics and computational fragment-based drug design: an overview and update. *Medicinal Research Reviews*, 33(3):554–598, 2013.

[45] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. Learning gradient fields for molecular conformation generation. In *International Conference on Machine Learning*, pp. 9558–9568. PMLR, 2021.

[46] Richard B Silverman. Design of selective neuronal nitric oxide synthase inhibitors for the prevention and treatment of neurodegenerative diseases. *Accounts of chemical research*, 42(3): 439–451, 2009.

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

[48] Martin Stahl, Nikolay P Todorov, Tim James, Harald Mauser, Hans-Joachim Boehm, and Philip M Dean. A validation study on the practical use of automated de novo design. *Journal of computer-aided molecular design*, 16(7):459–478, 2002.

[49] Minyi Su, Qifan Yang, Yu Du, Guoqin Feng, Zhihai Liu, Yan Li, and Renxiao Wang. Comparative assessment of scoring functions: The CASF-2016 update. *Journal of Chemical Information and Modeling*, 59(2):895–913, November 2018.

[50] David C Thompson, R Aldrin Denny, Ramaswamy Nilakantan, Christine Humblet, Diane Joseph-McCarthy, and Eric Feyfant. Confirm: connecting fragments found in receptor molecules. *Journal of Computer-Aided Molecular Design*, 22(10):761–772, 2008.

[51] Brian L Trippe, Jason Yim, Doug Tischer, Tamara Broderick, David Baker, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

[52] Vincenzo Tschinke and Nissim Claude Cohen. The newlead program: a new method for the design of candidate structures from pharmacophoric hypotheses. *Journal of medicinal chemistry*, 36(24):3863–3870, 1993.

[53] Aaron M Virshup, Julia Contreras-García, Peter Wipf, Weitao Yang, and David N Beratan. Stochastic voyages into uncharted chemical space produce a representative library of all possible drug-like compounds. *Journal of the American Chemical Society*, 135(19):7296–7303, 2013.

[54] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28 (1):31–36, 1988.

[55] Minkai Xu, Wujie Wang, Shitong Luo, Chence Shi, Yoshua Bengio, Rafael Gomez-Bombarelli, and Jian Tang. An end-to-end framework for molecular conformation generation via bilevel programming. In *International Conference on Machine Learning*, pp. 11537–11547. PMLR, 2021.

[56] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. *arXiv preprint arXiv:2203.02923*, 2022.

[57] Yuyao Yang, Shuangjia Zheng, Shimin Su, Chao Zhao, Jun Xu, and Hongming Chen. Syntalinker: automatic fragment linking with deep conditional transformer neural networks. *Chemical science*, 11(31):8312–8322, 2020.

[58] Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J.C. Burges, Ainur Yessenalina, and Qiang Liu. Computational approaches to sentence completion. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 601–610, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `https://aclanthology.org/P12-1063`.

A Appendix

Algorithm 1 Training

Input: linker $\boldsymbol{x}$, context $\boldsymbol{u}$, neural network φ
Sample $t \sim \mathcal{U}(0, \ldots, T),\ \boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$
$\boldsymbol{z}_t \leftarrow \alpha_t \boldsymbol{x} + \sigma_t \boldsymbol{\epsilon}_t$
$\hat{\boldsymbol{\epsilon}}_t \leftarrow \varphi(\boldsymbol{z}_t, \boldsymbol{u}, t)$
Minimize $\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}_t\|^2$

Algorithm 2 Sampling

Input: context $\boldsymbol{u}$, neural network φ
Center context $\boldsymbol{u}$ at $f(\boldsymbol{u})$
Sample $\boldsymbol{z}_T \sim \mathcal{N}(0, \boldsymbol{I})$
for t in $T,\ T-1,\ \ldots\ 1$:
    Sample $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \boldsymbol{I})$
    $\hat{\boldsymbol{\epsilon}}_t \leftarrow \varphi(\boldsymbol{z}_t, \boldsymbol{u}, t)$
    $\boldsymbol{z}_{t-1} \leftarrow (1/\overline{\alpha}_t) \cdot \boldsymbol{z}_t - \overline{\sigma}_t^2/(\overline{\alpha}_t \sigma_t) \cdot \hat{\boldsymbol{\epsilon}}_t + \varsigma_t \cdot \boldsymbol{\epsilon}$
end for
Sample $\boldsymbol{x} \sim p(\boldsymbol{x}|\boldsymbol{z}_0, \boldsymbol{u})$

A.1 Parameters of the True Denoising Process

The parameters of the true denoising process $q(\boldsymbol{z}_{t-1}|\boldsymbol{x}, \boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{z}_{t-1}; \boldsymbol{\mu}_t(\boldsymbol{x}, \boldsymbol{z}_t), \varsigma_t^2 \boldsymbol{I})$ are given as:

$$\boldsymbol{\mu}_t(\boldsymbol{x}, \boldsymbol{z}_t) = \frac{\overline{\alpha}_t \sigma_{t-1}^2}{\sigma_t^2} \boldsymbol{z}_t + \frac{\alpha_s \overline{\sigma}_t^2}{\sigma_t^2} \boldsymbol{x} \ \text{ and } \ \varsigma_t = \frac{\overline{\sigma}_t \sigma_{t-1}}{\sigma_t}. \tag{5}$$

A.2 Equivariance of the Conditional Diffusion Model

Proposition 1. *Consider a prior noise distribution $p(\boldsymbol{z}_T|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{z}_T; f(\boldsymbol{u}), \boldsymbol{I})$ and transition distributions $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u}) = q(\boldsymbol{z}_{t-1}|\hat{\boldsymbol{x}}, \boldsymbol{z}_t)$, where q is an isotropic Gaussian, $f : \mathbb{R}^{M\times 3} \to \mathbb{R}^3$ is a function operating on 3D point clouds, and $\hat{\boldsymbol{x}}$ is an approximation computed by the neural network φ according to Equation (1). Let the conditional denoising probabilistic model p be a Markov chain defined as*

$$p(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{u}) = p(\boldsymbol{z}_T|\boldsymbol{u}) \prod_{t=1}^{T} p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u}). \tag{6}$$

If f is O(3)-equivariant and φ is equivariant to joint O(3)-transformations of $\boldsymbol{z}_t$ and $\boldsymbol{u}$, then $p(\boldsymbol{z}_0|\boldsymbol{u})$ is O(3)-equivariant.

Proof. O(3)-equivariance of function f and the fact that q is isotropic Gaussian distribution implies O(3)-equivariance of the prior distribution:

$$p(\boldsymbol{R}\boldsymbol{z}_T|\boldsymbol{R}\boldsymbol{u}) = \mathcal{N}(\boldsymbol{R}\boldsymbol{z}_T|f(\boldsymbol{R}\boldsymbol{u}), \boldsymbol{I}) = \mathcal{N}(\boldsymbol{R}\boldsymbol{z}_T|\boldsymbol{R}f(\boldsymbol{u})) = \mathcal{N}(\boldsymbol{z}_T|f(\boldsymbol{u})) = p(\boldsymbol{z}_T|\boldsymbol{u}).$$

Likewise, O(3)-equivariance of function φ and Equation (1) imply O(3)-equivariance of all transition probabilities $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u})$.

To obtain the distribution $p(\boldsymbol{z}_0|\boldsymbol{u})$ of data point $\boldsymbol{z}_0$, we can consider joint distribution $p(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{u})$ and marginalize it by $\boldsymbol{z}_{1\ldots T}$:

$$p(\boldsymbol{z}_0|\boldsymbol{u}) = \int p(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{u}) d\boldsymbol{z}_{1\ldots T} = \int p(\boldsymbol{z}_T|\boldsymbol{u}) \prod_{t=0}^{T-1} p(\boldsymbol{z}_t|\boldsymbol{z}_{t+1}, \boldsymbol{u}) d\boldsymbol{z}_{1\ldots T}.$$

Having prior and all transition distributions equivariant, it is now trivial to show O(3)-equivariance of $p(\boldsymbol{z}_0|\boldsymbol{u})$:

$$
\begin{aligned}
p(\boldsymbol{R}\boldsymbol{z}_0|\boldsymbol{R}\boldsymbol{u}) &= \int p(\boldsymbol{R}\boldsymbol{z}_T|\boldsymbol{R}\boldsymbol{u}) \prod_{t=0}^{T-1} p(\boldsymbol{R}\boldsymbol{z}_t|\boldsymbol{R}\boldsymbol{z}_{t+1}, \boldsymbol{R}\boldsymbol{u}) d\boldsymbol{z}_{1\ldots T} \\
&= \int p(\boldsymbol{z}_T|\boldsymbol{u}) \prod_{t=0}^{T-1} p(\boldsymbol{R}\boldsymbol{z}_t|\boldsymbol{R}\boldsymbol{z}_{t+1}, \boldsymbol{R}\boldsymbol{u}) d\boldsymbol{z}_{1\ldots T} \quad \text{(equivariant prior } p(\boldsymbol{z}_T|\boldsymbol{u})) \\
&= \int p(\boldsymbol{z}_T|\boldsymbol{u}) \prod_{t=0}^{T-1} p(\boldsymbol{z}_t|\boldsymbol{z}_{t+1}, \boldsymbol{u}) d\boldsymbol{z}_{1\ldots T} \quad \text{(equivariant transition kernels } p(\boldsymbol{z}_t|\boldsymbol{z}_{t+1}, \boldsymbol{u})) \\
&= \int p(\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T|\boldsymbol{u}) d\boldsymbol{z}_{1\ldots T} = p(\boldsymbol{z}_0|\boldsymbol{u}).
\end{aligned}
$$

□

A.3 Problem with Translations

Consider transition probability $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u}) = q(\boldsymbol{z}_{t-1}|\hat{\boldsymbol{x}}, \boldsymbol{z}_t)$. Translation equivariance of $p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u})$ means that

$$p(\boldsymbol{z}_{t-1} + \boldsymbol{t}|\boldsymbol{z}_t + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}) = p(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{u}) \;\; \forall \boldsymbol{t} \in \mathbb{R}^3. \tag{7}$$

More precisely,

$$\mathcal{N}(\boldsymbol{z}_{t-1} + \boldsymbol{t}; \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t} + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}), \varsigma_t^2 \boldsymbol{I}) = \mathcal{N}(\boldsymbol{z}_{t-1}; \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}), \varsigma_t^2 \boldsymbol{I}), \tag{8}$$

where

$$\hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}) = \boldsymbol{\mu}_t(\hat{\boldsymbol{x}}, \boldsymbol{z_t}) = \frac{\overline{\alpha}_t \sigma_{t-1}^2}{\sigma_t^2} \boldsymbol{z}_t + \frac{\alpha_{t-1} \overline{\sigma}_t^2}{\sigma_t^2} \hat{\boldsymbol{x}}, \tag{9}$$

and

$$\hat{\boldsymbol{x}} = \frac{1}{\alpha_t} \boldsymbol{z}_t - \frac{\sigma_t}{\alpha_t} \varphi(\boldsymbol{z_t}, \boldsymbol{u}, t). \tag{10}$$

Therefore, the mean of this distribution can be written as:

$$\hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}) = \frac{1}{\overline{\alpha}_t} \boldsymbol{z_t} - \frac{\overline{\sigma}_t^2}{\overline{\alpha}_t \sigma_t} \varphi(\boldsymbol{z_t}, \boldsymbol{u}, t). \tag{11}$$

Neural network φ is translation invariant meaning that $\varphi(\boldsymbol{z}_t + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}, t) = \varphi(\boldsymbol{z}_t, \boldsymbol{u}, t)$.

It means that:

$$
\begin{aligned}
\hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t} + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}) &= \frac{1}{\overline{\alpha}_t}(\boldsymbol{z_t} + \boldsymbol{t}) - \frac{\overline{\sigma}_t^2}{\overline{\alpha}_t \sigma_t} \varphi(\boldsymbol{z_t} + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}, t) && (12) \\
&= \frac{1}{\overline{\alpha}_t}(\boldsymbol{z_t} + \boldsymbol{t}) - \frac{\overline{\sigma}_t^2}{\overline{\alpha}_t \sigma_t} \varphi(\boldsymbol{z_t}, \boldsymbol{u}, t) && (13) \\
&= \frac{1}{\overline{\alpha}_t} \boldsymbol{z_t} - \frac{\overline{\sigma}_t^2}{\overline{\alpha}_t \sigma_t} \varphi(\boldsymbol{z_t}, \boldsymbol{u}, t) + \frac{1}{\overline{\alpha}_t} \boldsymbol{t} && (14) \\
&= \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}) + \frac{1}{\overline{\alpha}_t} \boldsymbol{t} && (15) \\
&= \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}) + \lambda_t \boldsymbol{t}. && (16)
\end{aligned}
$$

So we see that $\hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u})$ is equivariant to translations, however input and output translations **are not equal** because $\lambda \neq 1$. It means that equivariance of distributions from Equations (7) and (8) does not hold. More formally,

$$
\begin{aligned}
p(\boldsymbol{z}_{t-1} + \boldsymbol{t}|\boldsymbol{z}_t + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}) &= \mathcal{N}\left(\boldsymbol{z}_{t-1} + \boldsymbol{t}; \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}) + \lambda \boldsymbol{t}, \varsigma_t^2 \boldsymbol{I}\right) && (17) \\
&= \mathcal{N}\left(\boldsymbol{z}_{t-1}; \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}) + (\lambda - 1)\boldsymbol{t}, \varsigma_t^2 \boldsymbol{I}\right) && (18) \\
&\neq \mathcal{N}\left(\boldsymbol{z}_{t-1}; \hat{\boldsymbol{\mu}}_t(\boldsymbol{z_t}, \boldsymbol{u}), \varsigma_t^2 \boldsymbol{I}\right). && (19)
\end{aligned}
$$

We can also write that

$$p(\boldsymbol{z}_{t-1} + \boldsymbol{t} | \boldsymbol{z}_t + \boldsymbol{t}, \boldsymbol{u} + \boldsymbol{t}) = p\left(\boldsymbol{z}_{t-1} + (1-\lambda)\boldsymbol{t} | \boldsymbol{z}_t, \boldsymbol{u}\right) \neq p\left(\boldsymbol{z}_{t-1} | \boldsymbol{z}_t, \boldsymbol{u}\right). \tag{20}$$

A.4 Diffusion Model

We trained all DiffLinker models with $T = 500$ diffusion steps using polynomial noise schedule:

$$\alpha_t = (1 - 2s) \cdot (1 - (t/T)^2), \tag{21}$$

where $s = 10^{-5}$ is a precision value that helps to avoid numerically unstable situations [19].

A.5 Dynamics

EGNN takes as input a graph of atoms belonging to the linker $\boldsymbol{z}_t$ and its context $\boldsymbol{u}$ represented by feature vectors $_i \in \mathbb{R}^{\text{in}}$ and coordinates $_i \in \mathbb{R}^3$. Feature vector $_i$ consists of atom types, fragments flag and time step t. If anchors are known, additionally anchor flag is passed. If the model is conditioned on the protein pocket, additionally pocket flag is passed.

First, atom features are passed to the encoder: $_i \rightarrow$ Linear(in, nf) $\rightarrow {}^0_i$.

Next, as discussed in Section 4.2, L Equivariant Graph Convolutional Layers (EGCL) are sequentially applied. Learnable components of EGCL $\phi_e,\ \phi_h, \phi_r$ are implemented as neural networks that include fully-connected layers (FC), batch normalization layers (BN) and activations SiLU.

Message ϕ_e: takes a pair of node embeddings ${}^l_i$ and ${}^l_j$ and the squared distance $d^2_{ij} = \|_i -_j \|^2$ between these nodes and outputs a message $_{ij} \in \mathbb{R}^{\text{nf}}$:

$$\text{concat}[{}^l_i, {}^l_j, d^2_{ij}] \rightarrow \{\text{FC}(2 \cdot \text{nf} + 1, \text{nf}) \rightarrow \text{SiLU} \rightarrow \text{FC}(\text{nf}, \text{nf}) \rightarrow \text{SiLU}\} \rightarrow {}_{ij}$$

Features update ϕ_h: takes as input node embedding ${}^l_i$ and its aggregated message $_i = \sum_j {}_{ij}$ and returns the updated node embedding:

$$\text{concat}[{}^l_i, {}_i] \rightarrow \{\text{FC}(2 \cdot \text{nf}, \text{nf}) \rightarrow \text{BN} \rightarrow \text{SiLU} \rightarrow \text{FC}(\text{nf}, \text{nf}) \rightarrow \text{BN} \rightarrow \text{add}({}^l_i)\} \rightarrow {}^{l+1}_i$$

Coordinates update ϕ_r: takes the same input as ϕ_e and outputs a scalar value

$$\text{concat}[{}^l_i, {}^l_j, d^2_{ij}] \rightarrow \{\text{FC}(2 \cdot \text{nf} + 1, \text{nf}) \rightarrow \text{SiLU} \rightarrow \text{FC}(\text{nf}, \text{nf}) \rightarrow \text{SiLU} \rightarrow \text{FC}(\text{nf}, 1)\ \} \rightarrow \text{output}$$

Equivariance The equivariance of EGCLs is achieved by construction. The messages ϕ_e and the node updates ϕ_h depend only on scalar node features and distances between nodes that are E(3)-invariant. Coordinate updates ϕ_{vel} additionally depend linearly on the difference between coordinate vectors $\boldsymbol{r}^l_i$ and $\boldsymbol{r}^l_j$, which makes them E(3)-equivariant.

Training We trained separate models for ZINC, Multi-Frag and Pocket datasets. Hyper parameters of the models and average time required for training one epoch are provided in Table 2. All models were trained on a single Tesla V100-PCIE-32GB GPU using Adam with learning rate $2 \cdot 10^{-5}$ and weight decay 10^{-13}.

A.6 Linker Size Prediction

Graph neural network for predicting probabilities of the number of atoms in the prospective linker for a given set of fragments takes as input a fully-connected graph of atoms belonging to the fragments represented by feature vectors $_i \in \mathbb{R}^{\text{in}}$ and inter-atomic squared distances $d^2_{ij} = \|_i -_j \|^2$, and outputs a vector of probabilities corresponding to the predefined linker sizes $\in [0, 1]^{\text{out}}$.

First, node embeddings are computed: $_i \rightarrow$ Linear(in, nf) $\rightarrow {}^0_i$.

Next, a sequence of L Graph Convolutional Layers (GCL) is applied. Learnable components of GCL $\phi_e,\ \phi_h$ are implemented in the same way as for EGNN.

Table 2: Hyperparameters of EGNN models trained on ZINC, Multi-Fragment and Pocket datasets.

dataset	given anchors	nf	L	batch size	epochs	time per epoch, min
ZINC	no	128	8	128	300	15.2
ZINC	yes	128	8	128	300	17.1
Multi-Fragment	no	128	6	64	839	10.6
Multi-Fragment	yes	128	6	128	1240	10.1
Pocket (full atomic representation)	yes	128	6	32	420	20.3
Pocket (backbone representation)	yes	128	6	32	620	10.7
Pocket (no pocket)	yes	128	6	32	670	8.4

Table 3: Hyperparameters of SizeGNN models trained on ZINC and Multi-Fragment datasets.

dataset	in	hid	out	L	batch size	epochs	time per epoch, min
ZINC	8	256	10	5	256	53	10.6
Multi-Fragment	9	256	33	5	256	119	9.2

Finally, node embeddings ${}_i^L$ are projected onto $\mathbb{R}^{\text{out}}$, aggregated and normalized resulting in the vector of label probabilities:

$$ {}_i^L \rightarrow \{\text{FC(nf, out)} \rightarrow \text{Mean} \rightarrow \text{Softmax}\} \rightarrow $$

Training We trained two models for ZINC and Multi-Frag datasets. Hyper parameters of the models and average time required for training one epoch are provided in Table 3. Both models were trained using Adam with learning rate 10^{-4} and weight decay 10^{-13}. Both models were trained on a single Tesla V100-PCIE-32GB GPU.

A.7 3DLinker on GEOM Dataset

In order to run 3DLinker on GEOM dataset, we had to additionally filter the original test set consisting of 1,290 input fragment sets and remove examples with more than 3 disconnected fragments. For the remainder test set that included 1,172 input fragment triplets, we ran 3DLinker twice: first, to connect two randomly selected fragments (10 samples per fragment pair) and then to connect the resulting compound with the third fragment (10 samples per input). In both steps, we used the half of the original linker size. Overall, we obtained 100 samples for each input fragment triplet. We used a pre-trained 3DLinker model available at `https://github.com/YinanHuang/3DLinker`. The results are provided in Tables 1 and 4.

A.8 Evaluation Details

The principal difference between our and other methods is that we generate 3D point cloud of atoms that should be further connected with covalent bonds while other methods generate covalent bonds along with atom types. We emphasize that both DeLinker and 3DLinker employ valency rules at each generation step which makes is much easier to achieve high validity of samples. In our case, DiffLinker learns these chemical rules from the data and places atoms at the relevant distances from each other. Since the output of DiffLinker is a 3D point cloud, we need to additionally compute covalent bonds between pairs of atoms based on their types and pairwise distances. For doing that, we use OpenBabel [37].

To be consistent in the evaluation methodology, we recomputed covalent bonds using OpenBabel for all molecules in ZINC and CASF test sets. Next, for each updated molecule, we obtained linkers by removing irrelevant atoms and saved the resulting molecules and fragments in SDF and SMILES formats. Molecules saved in SDF format were considered as ground truth and used for 3D comparison (for computing RMSD and SC_{RDKit} metrics). Molecules and linkers saved in SMILES format were

considered as ground truth and used for 2D comparison (novelty and recovery rates). To evaluate other methods, we used original SMILES representations.

Our samples For each generated point cloud, we computed covalent bonds with OpenBabel, and extracted the largest connected component. Next, we obtained a linker by matching the generated molecule with the corresponding fragments (computed with OpenBabel as explained above) and removing irrelevant atoms. Finally, we kekulized the resulting linker and saved the generated molecule with recomputed covalent bonds and the corresponding linker in SDF and SMILES formats.

Metrics To compute validity, we apply sanitization and additionally check that the molecule contains all atoms from fragments. For all other metrics, we consider only a subset of valid samples. To compute novelty, we first preprocess SMILES of the linker by removing stereochemistry and canonicalizing tautomer SMILES. Then, we count how many of the resulting generated linker SMILES were represented in the training set. To compute uniqueness, we compare SMILES of whole molecules and count number of unique molecules sampled for each input pair of fragments. To compute recovery, we compare SMILES of each molecule sampled for a given pair of fragments with SMILES of the corresponding ground-truth molecule. Before comparison, we remove hydrogens and stereochemistry from molecules. To compute RMSD, we consider only recovered molecules and align them with the corresponding ground-truth molecules using RDKit function `rdkit.Chem.rdMolAlign` which returns the optimal RMSD for aligning two molecules. To compute quantitative drug-likeness (QED) [5], we used RDKit function `rdkit.Chem.QED.qed`. To compute synthetic accessibility, we used function `calculateScore` provided by Ertl & Schuffenhauer [13] in the RDKit-compatible package `sascorer.py`. For calculating the number of rings, we used RDKit function `rdkit.Chem.rdMolDescriptors.CalcNumRings`.

A.9 Additional Results

As explained in Section A.7, we had to additionally filter the original test set consisting of 1,290 input fragment sets and remove examples with more than 3 disconnected fragments. For each input fragment set, we obtained 100 samples with 3DLinker. For consistency we report DiffLinker performance on GEOM in Tables 1 and 4 computed in the same setting: 1,290 input examples and 100 samples per input. DiffLinker results computed on the full GEOM test set with 250 samples per input are provided in Tables 5 and 6.

DiffLinker results on the Pockets test set with 100 samples per input are provided in Tables 5 and 6 as well.

A.10 2D Filters

2D Filters used by Imrie et al. [24] for constructing ZINC and CASF datasets include synthetic accessibility [13], ring aromaticity (RA), and pan-assay interference compounds (PAINS) [4] criteria. RA controls the correctness of the covalent bond orders in the rings of the a linker and PAINS checks if a molecule does not contain compounds that often give false-positive results in high-throughput screens [4]. Even though we used the same datasets as in Imrie et al. [24] that were created using all three filters, we however modify the metric "Passed 2D Filters" by removing SA from it. Instead we introduce our own SA-based metric that we report separately.

Problem with SA filter used by Imrie et al. [24] and Huang et al. [21] The molecule is considered to pass the synthetic accessibility filter if its SA-score is lower than SA-score of the corresponding pair of fragments. Even though our models performed on par or better than DeLinker and 3DLinker according to all other metrics, almost all molecules generated by our models did not pass SA-filter. We investigated this issue and figured out that SMILES of fragments passed to SA filter by DeLinker and 3DLinker contained dummy atoms representing anchors. These atoms did not have any atom type assigned and therefore such molecules were considered hard to be synthesised. For almost all molecules in the test set SA-scores of fragments with dummy atoms were higher than SA-scores of

Table 4: Metrics assessing the ability of the methods to generate molecules that are chemically and geometrically similar to the reference ones.

	Method	2D Filters, %	Recovery, %	RMSD ↓	SC_{RDKit} > 0.7	> 0.8	> 0.9	Avg ↑
ZINC	DeLinker + ConfVAE + MMFF	84.88	80.2	5.48	3.73	0.61	0.09	0.49
	3DLinker (given anchors)	84.24	**94.0**	**0.10**	**99.86**	**97.05**	63.78	0.92
	3DLinker	83.72	**93.5**	**0.11**	99.83	96.22	63.63	0.92
	DiffLinker	**86.26**	82.0	0.34	99.72	94.62	**67.85**	**0.93**
	DiffLinker (given anchors)	84.36	87.2	0.32	**99.96**	**97.02**	**71.73**	**0.94**
	DiffLinker (sampled size)	**87.98**	70.7	0.34	99.37	90.21	51.90	0.90
	DiffLinker (given anchors, sampled size)	84.76	77.5	0.35	99.67	95.04	56.35	0.91
CASF	DeLinker + ConfVAE + MMFF	77.25	**52.8**	11.89	1.24	0.19	0.03	0.39
	DiffLinker	**87.73**	42.8	0.44	92.17	79.62	50.14	0.84
	DiffLinker (given anchors)	82.37	**50.2**	0.37	**95.07**	**89.05**	**60.63**	**0.86**
	DiffLinker (sampled size)	**89.27**	40.5	**0.34**	92.83	79.12	43.99	0.82
	DiffLinker (given anchors, sampled size)	82.08	48.8	**0.32**	**94.52**	**87.90**	**55.25**	**0.84**
GEOM	3DLinker	**94.10**	0.0	—	60.63	29.30	8.22	0.72
	DiffLinker	49.56	**86.8**	0.12	**96.11**	**89.09**	**73.56**	**0.93**
	DiffLinker (given anchors)	51.38	**86.2**	0.11	**95.73**	**88.24**	**72.39**	**0.93**
	DiffLinker (sampled size)	56.35	70.9	**0.07**	92.38	82.67	58.39	0.88
	DiffLinker (given anchors, sampled size)	**58.41**	70.8	**0.07**	91.63	80.98	56.16	0.88

Table 5: Performance metrics on GEOM (250 samples, full test set) and Pockets test set (100 samples). The first three metrics assess the chemical relevance of the generated molecules. The last three metrics evaluate the standard generative properties of the methods.

	Method	QED ↑	SA ↓	# Rings ↑	Valid, %	Unique, %	Novel, %
GEOM	DiffLinker	**0.48**	**2.99**	0.75	**93.4**	31.7	68.6
	DiffLinker (given anchors)	**0.49**	**3.02**	**0.79**	**93.5**	32.1	68.4
	DiffLinker (sampled size)	0.45	3.27	0.76	87.1	**57.3**	**76.1**
	DiffLinker (given anchors, sampled size)	0.46	3.33	**0.84**	88.6	**58.2**	**76.1**
Pockets	DiffLinker (pocket atoms)	0.45	3.89	**1.06**	88.7	**62.5**	**73.1**
	DiffLinker (pocket backbone)	**0.45**	**3.77**	0.95	**90.4**	60.9	72.8
	DiffLinker (unconditioned)	**0.45**	**3.83**	**1.10**	**93.6**	**61.1**	**74.9**

Table 6: Metrics assessing the ability of the methods to generate molecules that are chemically and geometrically similar to the reference ones. For GEOM (250 samples, full test set) and Pockets test set (100 samples).

	Method	2D Filters, %	Recovery, %	RMSD ↓	SC_{RDKit} > 0.7	> 0.8	> 0.9	Avg ↑
GEOM	DiffLinker	49.54	**89.0**	0.11	**95.90**	**88.81**	**73.28**	**0.93**
	DiffLinker (given anchors)	51.32	**87.9**	0.11	**95.58**	**87.78**	**72.09**	**0.93**
	DiffLinker (sampled size)	**56.11**	77.4	**0.07**	92.13	82.12	58.02	0.88
	DiffLinker (given anchors, sampled size)	**58.22**	77.1	**0.07**	91.39	80.39	55.70	0.88
Pockets	DiffLinker (pocket atoms)	**69.47**	**37.3**	**0.87**	**71.61**	**57.30**	**34.52**	**0.78**
	DiffLinker (pocket backbone)	**66.48**	**37.8**	0.89	**65.58**	**52.77**	**31.50**	**0.76**
	DiffLinker (unconditioned)	59.43	36.5	**0.89**	64.06	50.95	30.59	0.75

the whole molecules. However, for most of fragments without dummy atoms SA-scores were much lower. Figure 3 shows 4 examples of molecules and fragments with and without dummy atoms and the corresponding SA-scores. We can conclude that SA-filter proposed by authors of DeLinker shows nothing but the fact that molecules with unknown atoms are hard to be synthesized. Therefore, we considered this metric to be irrelevant and excluded it from our report. Instead, we report average Synthetic Accessibility score of full generated molecules.

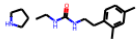
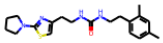
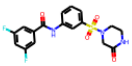
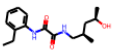
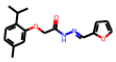
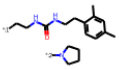
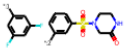
our fragments (without dummy atoms)	 2.11	2.01	2.11	2.33
full molecules	 2.29	 2.23	 2.88	 2.21
DeLinker fragments (with dummy atoms)	 3.26	 3.49	4.37	3.44

Figure 3: Synthetic accessibility scores (SA-scores) for fragments without dummy atoms (top row), full molecules (middle row) and fragments with dummy atoms (bottom row).

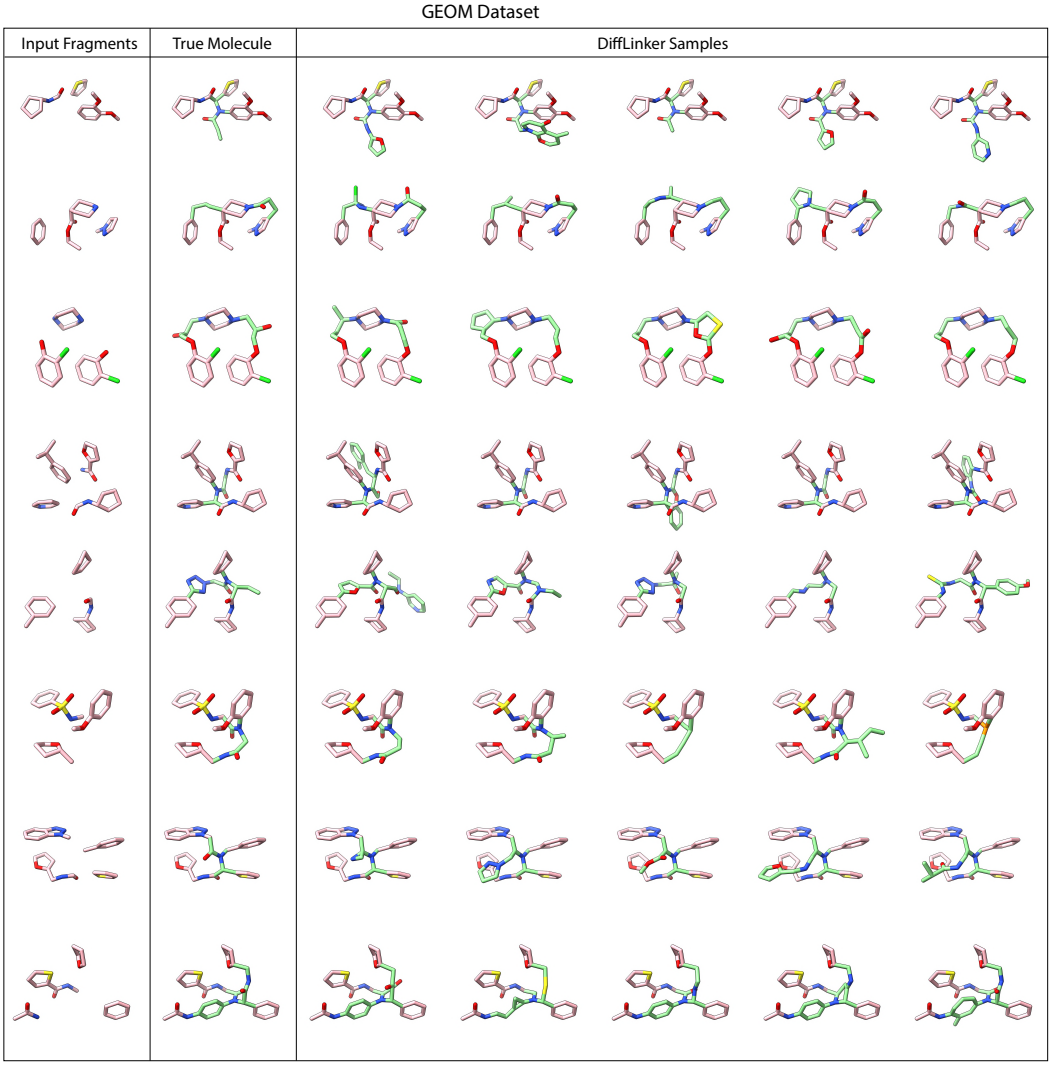


Figure 4: Examples of linkers generated by DiffLinker (sampled size) for fragments from GEOM datasets.

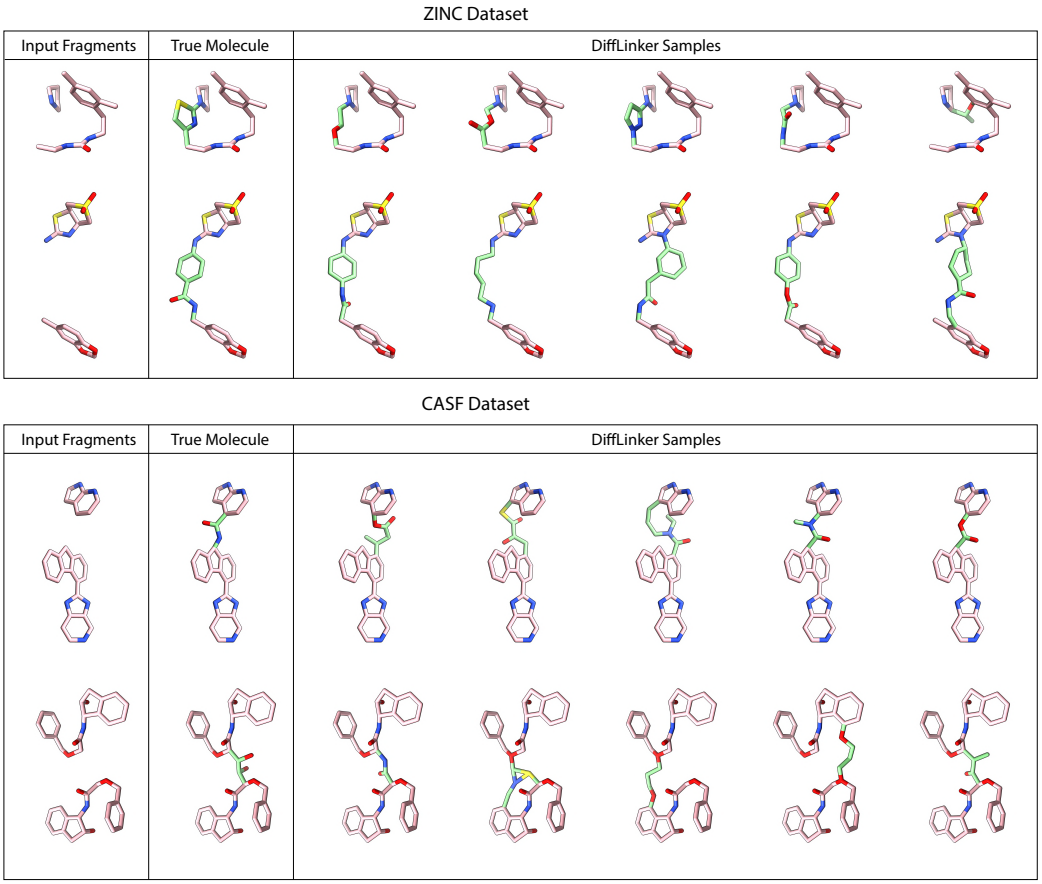


Figure 5: Examples of linkers generated by DiffLinker (sampled size) for fragments from CASF and ZINC datasets.