SEMI-EQUIVARIANT CONDITIONAL NORMALIZING FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the problem of learning conditional distributions of the form $p(G|\hat{G})$, where G and $\hat{G}$ are two 3D graphs, using continuous normalizing flows. We derive a semi-equivariance condition on the flow which ensures that conditional invariance to rigid motions holds. We demonstrate the effectiveness of the technique in the molecular setting of receptor-aware ligand generation.

1 INTRODUCTION

Data consisting of sets of three-dimensional points appear in a number of scientific and engineering settings; examples include molecular chemistry, high energy physics, and computer vision. In many cases, these point sets can be effectively described by a 3D graph, that is a graph whose vertices correspond to points in $\mathbb{R}^3$. In this work, we are concerned with learning conditional distributions of the form $p(G|\hat{G})$, where G and $\hat{G}$ are two 3D graphs. Construction of such distributions is useful in various scenarios: in the molecular setting, $\hat{G}$ will represent a receptor / target and G will represent a ligand; in the setting of shape completion in computer vision, $\hat{G}$ will represent the part of the point cloud that we have been given, and G will represent the completion of this point cloud.

We present a method for learning such conditional distributions based on continuous normalizing flows. A critical aspect of the problem is to learn a distribution which respects the correct invariances. Specifically, there must be an invariance to rigid motions which is a kind of "conditional invariance" which is expressed jointly in terms of G and $\hat{G}$. Our main theoretical result is that this conditional invariance can be satisfied by a novel form of *semi-equivariance* of the normalizing flow. We then empirically demonstrate the utility of our technique in the molecular setting: our method allows for effective receptor-aware ligand generation.

Related Work Continuous normalizing flows originated in the work of Chen et al. (2018a) and have been extended in many ways, for example (Zhang & Zhao, 2022; Onken et al., 2021; Ghosh et al., 2020). Köhler et al. (2019) and Rezende et al. (2019) presented equivariant normalizing flows which respect natural symmetries. Liu et al. (2019) introduced graph normalizing flows to obtain generative models of graph structures. Köhler et al. (2020) and Satorras et al. (2021a) constructed equivariant graph normalizing flows, by incorporating equivariant graph neural networks into an ODE framework to obtain invertible equivariant functions. A variety of works have tackled the problem of receptor-aware ligand generation, including methods based on VAEs, e.g. (Ragoza et al., 2022); autoregressive models, e.g. (Luo et al., 2021; Liu et al., 2022; Peng et al., 2022; Drotár et al., 2021); diffusion, e.g. (Schneuing et al., 2022; Lin et al., 2022); and reinforcement learning, e.g. (Li et al., 2021; Thomas et al., 2021; Fialková et al., 2021; Fu et al., 2022).

2 SEMI-EQUIVARIANT CONDITIONAL NORMALIZING FLOWS

2.1 NOTATION AND GOAL

Notation A 3D graph is given by $G = (N, V, E, A)$, where N is the number of vertices; V is the list of vertices; E is the list of edges; and A is the set of global graph properties, i.e. properties which apply to the entire graph. The vertex list[1] is $V = (\mathbf{v}_i)_{i=1}^N$ where each vertex is specified by

[1] We use lists, rather than sets, so as to make the action of permutations clear.

a vector $\mathbf{v}_i = (\mathbf{x}_i, \mathbf{h}_i)$; in which $\mathbf{x}_i \in \mathbb{R}^3$ is the position of the vertex, and $\mathbf{h}_i \in \mathbb{R}^{d_h}$ contains the properties of the vertex. The edges in the graph G are undirected, and the edge list E is specified by a neighbourhood relationship. Specifically, if η_i is the set of vertex i's neighbours, then we write $E = (\mathbf{e}_{ij})_{i<j:j\in\eta_i}$. The vector $\mathbf{e}_{ij} \in \mathbb{R}^{d_e}$ contains the properties of the edge connecting vertices i and j. Finally, the graph properties are given by K individual properties, i.e. $A = (\mathbf{a}_1, \ldots, \mathbf{a}_K)$.

Rigid Transformations The action of a rigid transformation $T \in E(3)$ on a graph G is given by $TG = (TN, TV, TE, TA)$ where $TV = (T\mathbf{v}_i)_{i=1}^N$ with $T\mathbf{v}_i = (T\mathbf{x}_i, \mathbf{h}_i)$. The other variables are unaffected by T; that is $TN = N$, $TE = E$, and $TA = A$.

Permutations The action of a permutation $\pi \in \mathbb{S}_N$ on a graph G with $N(G) = N$ is given by $\pi G = (\pi N, \pi V, \pi E, \pi A)$ where $\pi V = (\mathbf{v}_{\pi_i})_{i=1}^N$ and $\pi E = \left(\mathbf{e}_{\pi_i \pi_j}\right)_{i<j:j\in\eta_i}$. The other variables are unaffected by π; that is, $\pi N = N$ and $\pi A = A$.

Goal We assume that we have both a base graph and a complement graph, each of which is specified by a 3D graph. We denote the base graph as $\hat{G} = (\hat{N}, \hat{V}, \hat{E}, \hat{A})$ and the complement graph as $G = (N, V, E, A)$. Our goal is to learn a conditional generative model: given the base graph, we would like to generate possible complement graphs. Formally, we want to learn $p(G|\hat{G})$. We want our generative model to respect two types of symmetries, expressed mathematically as:

$$p(TG|T\hat{G}) = p(G|\hat{G}) \ \forall T \in E(3) \quad \text{and} \quad p(\pi G|\hat{\pi}\hat{G}) = p(G|\hat{G}) \ \forall \pi \in \mathbb{S}_N, \ \hat{\pi} \in \mathbb{S}_{\hat{N}} \tag{1}$$

The first condition says that if we transform both the complement graph and the base graph with the same rigid transformation, the probability should not change. The second condition says that permuting the order of either the complement graph or the base graph should not affect the probability.

General Approach We design our generative model $p(G|\hat{G})$ by using a Markov decomposition:

$$p(G|\hat{G}) = p(N, V, E, A|\hat{G}) = p(N|\hat{G}) \cdot p(V|N, \hat{G}) \cdot p(E|N, V, \hat{G}) \cdot p(A|N, V, E, \hat{G}) \tag{2}$$

In this paper, we focus on the first two terms on the right-hand side of the equation: the Number Distribution $p(N|\hat{G})$ and the Vertex Distribution $p(V|N, \hat{G})$. The latter two terms may also be specified (e.g. see Appendix C), but are not the focus of our investigation in this paper.

2.2 Two Flavours of EGNNs

In order to incorporate the relevant invariance properties, it will be helpful to use Equivariant Graph Neural Networks, also known as EGNNs (Satorras et al., 2021b). We now introduce two separate flavours of EGNNs, one which applies to the base graph alone, and a second which applies to the combination of the base graph and the complement graph.

Base Graph EGNN This is the standard EGNN which is described in (Satorras et al., 2021b), applied to the base graph. As we are referring to the base graph, we use hatted variables:

$$\hat{\mathbf{m}}_{ij}^\ell = \hat{\phi}_e(\hat{\mathbf{h}}_i^\ell, \hat{\mathbf{h}}_j^\ell, \|\hat{\mathbf{x}}_i^\ell - \hat{\mathbf{x}}_j^\ell\|^2, \|\hat{\mathbf{x}}_i^0 - \hat{\mathbf{x}}_j^0\|^2, \hat{\mathbf{e}}_{ij}, \{\hat{\mathbf{a}}_k\}) \qquad \hat{b}_{ij}^\ell = \sigma(\hat{\phi}_b(\hat{\mathbf{m}}_{ij}^\ell)) \qquad \hat{\mathbf{m}}_i^\ell = \sum_{j\in\hat{\eta}_i} \hat{b}_{ij}^\ell \hat{\mathbf{m}}_{ij}^\ell$$

$$\hat{\mathbf{x}}_i^{\ell+1} = \hat{\mathbf{x}}_i^\ell + \hat{\phi}_x(\hat{\mathbf{m}}_{ij}^\ell) \sum_{j\neq i} \frac{(\hat{\mathbf{x}}_i^\ell - \hat{\mathbf{x}}_j^\ell)}{\|\hat{\mathbf{x}}_i^\ell - \hat{\mathbf{x}}_j^\ell\| + 1} \qquad \hat{\mathbf{h}}_i^{\ell+1} = \hat{\mathbf{h}}_i^\ell + \hat{\phi}_h(\hat{\mathbf{h}}_i^\ell, \hat{\mathbf{m}}_i^\ell) \tag{3}$$

The particular Base Graph EGNN is thus specified by the functions $\hat{\phi}_e, \hat{\phi}_b, \hat{\phi}_x, \hat{\phi}_h$.

Conditional EGNN It is possible to design a joint EGNN on the base graph and the complement graph, by constructing a single graph to capture both. The main problem with this approach is that in many applications, the base graph can be much larger than the complement graph; for example, in the molecular scenario the receptor is 1-2 orders of magnitude larger than the ligand. As a result, this naive approach will lead to a situation in which the complement graph is "drowned out" by the base graph, making it difficult to learn about the complement graph. We therefore take a different approach: we compute summary signatures of the base graph based on the Base Graph EGNN, and use these as input to a Complement Graph EGNN. Our signatures will be based on the feature variables $\hat{\mathbf{h}}_j^\ell$ from each layer $\ell = 1, \ldots, \hat{L}$. These variables are invariant to rigid body transformations by construction; furthermore, we can introduce permutation invariance by averaging, that is by using

$\hat{\mathbf{h}}^{\ell}_{av} = \Sigma_{j=1}^{\hat{N}} \hat{\mathbf{h}}^{\ell}_j / \hat{N}$. Thus, the base graph at layer ℓ of the complement graph's EGNN is summarized by the signature $\hat{\mathbf{g}}^{\ell}$ which depends on both $\{\hat{\mathbf{h}}^{\ell}_{av}\}$, as well as the time t of the normalizing flow ODE (to be introduced shortly in Section 2.4):

$$\hat{\mathbf{g}}^0 = \phi_g^0\left(\hat{\mathbf{h}}^1_{av}, \ldots, \hat{\mathbf{h}}^{\hat{L}}_{av}, t\right) \qquad \text{and} \qquad \hat{\mathbf{g}}^{\ell} = \phi_g^{\ell}\left(\hat{\mathbf{g}}^{\ell-1}\right) \quad \ell = 1, \ldots, L \tag{4}$$

The Conditional EGNN is then specified by a form similar to the equations in (3), but in which the functions $\phi_e, \phi_b, \phi_x, \phi_h$ (now non-hatted, as we are referring to the complement graph) each have an additional dependence on the base graph signatures $\{\hat{\mathbf{g}}^{\ell}\}_{\ell=1}^{L}$. Further details are in Appendix A.

2.3 The Number Distribution: $p(N|\hat{G})$

Construction Given the invariance conditions described in Equation (1), we propose the following distribution. Let $\boldsymbol{\zeta}_N$ indicate a one-hot vector, where the index corresponding to N is filled in with a 1. Based on the output of the base graph EGNN, compute

$$p(N|\hat{G}) = \boldsymbol{\zeta}_N^T F_2\left(\Sigma_{i=1}^{\hat{N}} F_1(\hat{\mathbf{h}}^L_i)/\hat{N}\right) \tag{5}$$

where F_1 and F_2 are multilayer perceptrons, and the last layer of F_2 is a softmax of size equal to the maximum number of atoms allowed. Since we use the $\hat{\mathbf{h}}^L_i$ vectors (and not the $\hat{\mathbf{x}}^L_i$ vectors), we have rigid motion invariance, as $T\hat{\mathbf{h}}^L_i = \hat{\mathbf{h}}^L_i$. Since we use an average, we have permutation invariance.

Loss Function The loss function is straightforward: it is simply the negative log-likelihood of the number distribution, i.e. $L(\theta) = \mathbb{E}_{G,\hat{G}}[-\log p(N|\hat{G};\theta)]$.

2.4 The Vertex Distribution: $p(V|N,\hat{G})$ via Continuous Normalizing Flows

General Notation Define a vectorization operation on the vertex list V, which produces a vector $\mathbf{v}$; we refer to this as a *vertex vector*. Recall that $V = (\mathbf{v}_i)_{i=1}^N$ where $\mathbf{v}_i = (\mathbf{x}_i, \mathbf{h}_i)$. Let

$$\mathbf{x} = \texttt{concat}(\mathbf{x}_1, \ldots, \mathbf{x}_N) \qquad \mathbf{h} = \texttt{concat}(\mathbf{h}_1, \ldots, \mathbf{h}_N) \qquad \mathbf{v} = \texttt{concat}(\mathbf{x}, \mathbf{h}) \tag{6}$$

The vertex vector $\mathbf{v} \in \mathbb{R}^{d_v^N}$ where $d_v^N = (d_h + 3)N$. We denote the mapping from the vertex list V to the vertex vector $\mathbf{v}$ as the vectorization operation $\texttt{vec}(\cdot)$: we write $\mathbf{v} = \texttt{vec}(V)$ and $V = \texttt{vec}^{-1}(\mathbf{v})$. We have already described the action of rigid body transformations T and permutations π on the vertex list V in Section 2.1. It is easy to extend this to vertex vectors $\mathbf{v}$ using the $\texttt{vec}$ operation; we have $T\mathbf{v} = \texttt{vec}(T\texttt{vec}^{-1}(\mathbf{v}))$ and $\pi\mathbf{v} = \texttt{vec}(\pi\texttt{vec}^{-1}(\mathbf{v}))$.

Given the above, it is sufficient for us to describe the distribution $p_{vec}(\mathbf{v}|\hat{G})$ from which the vertex distribution $p(V|N,\hat{G})$ follows directly, $p(V|N,\hat{G}) = p_{vec}(\texttt{vec}(V)|\hat{G})$. Note that we have suppressed N in the condition in $p_{vec}(\cdot)$, as $\mathbf{v}$ is a vector of dimension d_v^N, so the N dependence is already implicitly encoded.

Complex-to-Complement Mapping and Semi-Equivariance Let γ be a function which takes as input the complex consisting of both the complement graph G and base graph $\hat{G}$, and outputs a new vertex list V' for the complement graph G:

$$V' = \gamma(G, \hat{G}) \tag{7}$$

We refer to γ as a *Complex-to-Complement Mapping*. A rigid body transformation $T \in E(3)$ consists of a rotation and translation; let the rotation be denoted as T_{rot}. Then we say that γ is *rotation semi-equivariant* if

$$\gamma(T_{rot}G, T\hat{G}) = T_{rot}\gamma(G, \hat{G}) \quad \text{for all } T \in E(3) \tag{8}$$

γ is said to be *permutation semi-equivariant* if

$$\gamma(\pi G, \hat{\pi}\hat{G}) = \pi\gamma(G, \hat{G}) \quad \text{for all } \pi \in \mathbb{S}_N \text{ and } \hat{\pi} \in \mathbb{S}_{\hat{N}} \tag{9}$$

Note in the definitions of both types of semi-equivariance, the differing roles played by the complement graph and base graph; as the equivariant behaviour only applies to the complement graph, we have used the term semi-equivariance.

Validity	
Ours	GraphBP
99.87%	99.75%

ΔBinding	
Ours	GraphBP
35.7%	22.76%

Table 1: Comparison of molecule validity and ΔBinding between proposed method and GraphBP.

Conditional Flow Let γ be a Complex-to-Complement Mapping. If $\mathbf{v}$ is a vertex vector, define $G_{\mathbf{v}}$ to be the graph G with the vertex set replaced by $\texttt{vec}^{-1}(\mathbf{v})$. Then the following ordinary differential equation is referred to as a *Conditional Flow*:

$$\frac{d\mathbf{u}}{dt} = \texttt{vec}\left(\gamma(G_{\mathbf{u}}, \hat{G})\right), \quad \text{with } \mathbf{u}(0) = \mathbf{z} \tag{10}$$

where the initial condition $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is a Gaussian random vector of dimension d_v^N, and the ODE is run until $t = 1$. $\mathbf{u}(1)$ is thus the output of the Conditional Flow.

Vertex Distributions with Appropriate Invariance We now have the necessary ingredients to construct a distribution $p_{vec}(\mathbf{v}|\hat{G})$ which yields a vertex distribution $p(V|N, \hat{G})$ that satisfies the required invariance conditions. The following is our main result (proof in Appendix B):

Theorem (Invariant Vertex Distribution)**.** *Let $\mathbf{u}(1)$ be the output of a Conditional Flow specified by the Complex-to-Complement Mapping γ. Let the mean position of the base graph be given by $\hat{\mathbf{x}}_{av} = \frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}} \hat{\mathbf{x}}_i$, and define the following quantities*

$$\alpha = \frac{N}{N + \hat{N}} \quad \Omega_{\hat{G}} = \begin{bmatrix} \mathbf{I}_{3N} - \frac{\alpha}{N}\mathbf{1}_{N\times N} \otimes \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{d_h N} \end{bmatrix} \quad \omega_{\hat{G}} = \begin{bmatrix} -(1-\alpha)\mathbf{1}_{N\times 1} \otimes \hat{\mathbf{x}}_{av} \\ \mathbf{0} \end{bmatrix} \tag{11}$$

where $\otimes$ indicates the Kronecker product. Finally, let $\mathbf{v} = \Omega_{\hat{G}}^{-1}\left(\mathbf{u}(1) - \omega_{\hat{G}}\right)$. Suppose that γ is both rotation semi-equivariant and permutation semi-equivariant. Then the resulting distribution on $\mathbf{v}$, that is $p_{vec}(\mathbf{v}|\hat{G})$, yields a vertex distribution $p(V|N, \hat{G}) = p_{vec}(\texttt{vec}(V)|\hat{G})$ that satisfies the invariance conditions in Equation (1).

Designing the Complex-to-Complement Mapping For the Complex-to-Complement Mapping, we choose to use the Conditional EGNN; that is, $V' = \gamma(G, \hat{G})$ is given by $v_i' = (\mathbf{x}_i^L, \mathbf{h}_i^L)$, the output of the EGNN's final layer. In practice, we use a slightly modified version of the foregoing, that is $v_i' = (\mathbf{x}_i^L - \mathbf{x}_i, \mathbf{h}_i^L)$, which we have found to converge well empirically. The rotation semi-equivariance of γ (both versions - ordinary and modified) follows straightforwardly from the rotation semi-equivariance of EGNNs and the rotation invariance of the base graph signatures $\{\hat{\mathbf{g}}^\ell\}_{\ell=1}^L$. Similarly, the permutation semi-equivariance follows from the permutation equivariance of EGNNs and the permutation invariance of the base graph signatures.

Loss Function The loss function and its optimization are implemented using standard techniques for continuous normalizing flows (Chen et al., 2018b; Grathwohl et al., 2018; Chen et al., 2018a). If the feature vectors contain discrete variables, then techniques based on variational dequantization (Ho et al., 2019) and argmax flows (Hoogeboom et al., 2021) can be used, for ordinal and categorical features respectively. This is parallel to the treatment in (Satorras et al., 2021a).

3 Applications to Target-Aware Molecule Generation

We now apply the theory we have developed to the problem of target-aware structure based molecule generation, in which the base graph plays the role of the receptor, while the complement graph plays the role of the ligand; vertices are atoms and edges are bonds. We train our model on the CrossDocked2020 dataset (Francoeur et al., 2020). Further details are included in Appendix D.

The validity is defined as the percentage of molecules that are chemically valid, in the sense that they can be sanitized by RDKit, see (Landrum, 2016). As shown in Table 1, our model produces ligands with a validity of 99.87%, surpassing the baseline, GraphBP (Liu et al., 2022). ΔBinding measures the measures the percentage of generated molecules that have higher predicted binding affinity to the target binding site than the corresponding reference molecule. As can be seen in Table 1, our method attains ΔBinding = 35.7% vs. 22.76% for GraphBP, which corresponds to a relative improvement of 56.81%. Further details are included in Appendix E.

REFERENCES

Pedro J Ballester and John BO Mitchell. A machine learning approach to predicting protein–ligand binding affinity with applications to molecular docking. Bioinformatics, 26(9):1169–1175, 2010.

Paul S Charifson, Joseph J Corkery, Mark A Murcko, and W Patrick Walters. Consensus scoring: A method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins. Journal of medicinal chemistry, 42(25):5100–5109, 1999.

Changyou Chen, Chunyuan Li, Liqun Chen, Wenlin Wang, Yunchen Pu, and Lawrence Carin Duke. Continuous-time flows for efficient inference and density estimation. In International Conference on Machine Learning, pp. 824–833. PMLR, 2018a.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. Advances in neural information processing systems, 31, 2018b.

Tiejun Cheng, Xun Li, Yan Li, Zhihai Liu, and Renxiao Wang. Comparative assessment of scoring functions on a diverse test set. Journal of chemical information and modeling, 49(4):1079–1093, 2009.

Tiejun Cheng, Qingliang Li, Zhigang Zhou, Yanli Wang, and Stephen H Bryant. Structure-based virtual screening for drug discovery: a problem-centric review. The AAPS journal, 14(1):133–141, 2012.

Robert S DeWitte and Eugene I Shakhnovich. Smog: de novo design method based on simple, fast, and accurate free energy estimates. 1. methodology and supporting evidence. Journal of the American Chemical Society, 118(47):11733–11744, 1996.

Pavol Drotár, Arian Rokkum Jamasb, Ben Day, Cătălina Cangea, and Pietro Liò. Structure-aware generation of drug-like molecules. arXiv preprint arXiv:2111.04107, 2021.

Jacob D Durrant and J Andrew McCammon. Nnscore 2.0: a neural-network receptor–ligand scoring function. Journal of chemical information and modeling, 51(11):2897–2903, 2011.

Vendy Fialková, Jiaxi Zhao, Kostas Papadopoulos, Ola Engkvist, Esben Jannik Bjerrum, Thierry Kogej, and Atanas Patronov. Libinvent: reaction-based generative scaffold decoration for in silico library design. Journal of Chemical Information and Modeling, 62(9):2046–2063, 2021.

Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode: the world of jacobian and kinetic regularization. In International conference on machine learning, pp. 3154–3164. PMLR, 2020.

Paul G Francoeur, Tomohide Masuda, Jocelyn Sunseri, Andrew Jia, Richard B Iovanisci, Ian Snyder, and David R Koes. Three-dimensional convolutional neural networks and a cross-docked data set for structure-based drug design. Journal of chemical information and modeling, 60(9):4200–4215, 2020.

Tianfan Fu, Wenhao Gao, Connor W Coley, and Jimeng Sun. Reinforced genetic algorithm for structure-based drug design. arXiv preprint arXiv:2211.16508, 2022.

Arnab Ghosh, Harkirat Behl, Emilien Dupont, Philip Torr, and Vinay Namboodiri. Steer: Simple temporal regularization for neural ode. Advances in Neural Information Processing Systems, 33: 14831–14843, 2020.

Holger Gohlke, Manfred Hendlich, and Gerhard Klebe. Knowledge-based scoring function to predict protein-ligand interactions. Journal of molecular biology, 295(2):337–356, 2000.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. arXiv preprint arXiv:1810.01367, 2018.

Mahmudulla Hassan, Daniel Castaneda Mogollon, Olac Fuentes, et al. Dlscore: A deep learning model for predicting protein-ligand binding affinities. 2018.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In International Conference on Machine Learning, pp. 2722–2730. PMLR, 2019.

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.

Sheng-You Huang and Xiaoqin Zou. Scoring and lessons learned with the csar benchmark using an improved iterative knowledge-based scoring function. Journal of chemical information and modeling, 51(9):2097–2106, 2011.

Nicholas A Hummell, Alexey V Revtovich, and Natalia V Kirienko. Novel immune modulators enhance caenorhabditis elegans resistance to multiple pathogens. Msphere, 6(1):e00950–20, 2021.

Arian Rokkum Jamasb, Ramon Viñas Torné, Eric J Ma, Yuanqi Du, Charles Harris, Kexin Huang, Dominic Hall, Pietro Lio, and Tom Leon Blundell. Graphein-a python library for geometric deep learning and network analysis on biomolecular structures and interaction networks. In ICML 2022 2nd AI for Science Workshop, 2022.

Douglas B Kitchen, Hélène Decornez, John R Furr, and Jürgen Bajorath. Docking and scoring in virtual screening for drug discovery: methods and applications. Nature reviews Drug discovery, 3(11):935–949, 2004.

David Ryan Koes, Matthew P Baumgartner, and Carlos J Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. Journal of chemical information and modeling, 53(8):1893–1904, 2013.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. In International conference on machine learning, pp. 5361–5370. PMLR, 2020.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: sampling configurations for multi-body systems with symmetric energies, 2019. URL https://arxiv.org/abs/1910.00753.

Greg Landrum. Rdkit: Open-source cheminformatics software. 2016. URL https://github.com/rdkit/rdkit/releases/tag/Release_2016_09_4.

Yibo Li, Jianfeng Pei, and Luhua Lai. Structure-based de novo drug design using 3d deep generative models. Chemical science, 12(41):13664–13675, 2021.

Haitao Lin, Yufei Huang, Meng Liu, Xuanjing Li, Shuiwang Ji, and Stan Z Li. Diffbp: Generative diffusion of 3d molecules for target protein binding. arXiv preprint arXiv:2211.11214, 2022.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/1e44fdf9c44d7328fecc02d677ed704d-Paper.pdf.

Meng Liu, Youzhi Luo, Kanji Uchino, Koji Maruhashi, and Shuiwang Ji. Generating 3d molecules for target protein binding. arXiv preprint arXiv:2204.09410, 2022.

Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein–ligand interaction scoring functions. Accounts of chemical research, 50 (2):302–309, 2017.

Shitong Luo, Jiaqi Guan, Jianzhu Ma, and Jian Peng. A 3d generative model for structure-based drug design. Advances in Neural Information Processing Systems, 34:6229–6239, 2021.

Campbell McInnes. Virtual screening strategies in drug discovery. Current opinion in chemical biology, 11(5):494–502, 2007.

Andrew T McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. Journal of cheminformatics, 13(1):1–20, 2021.

Jens Meiler, Michael Müller, Anita Zeidler, and Felix Schmäschke. Generation and evaluation of dimension-reduced amino acid parameter representations by artificial neural networks. Molecular modeling annual, 7(9):360–369, 2001.

Ingo Muegge. A knowledge-based scoring function for protein-ligand interactions: Probing the reference state. Perspectives in Drug Discovery and Design, 20(1):99–114, 2000.

Derek Onken, Samy Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow: Fast and accurate continuous normalizing flows via optimal transport. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 9223–9232, 2021.

Xingang Peng, Shitong Luo, Jiaqi Guan, Qi Xie, Jian Peng, and Jianzhu Ma. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. arXiv preprint arXiv:2205.07249, 2022.

Matthew Ragoza, Tomohide Masuda, and David Ryan Koes. Generating 3d molecules conditional on receptor binding sites with deep generative models. Chemical science, 13(9):2701–2713, 2022.

Anthony K Rappé, Carla J Casewit, KS Colwell, William A Goddard III, and W Mason Skiff. Uff, a full periodic table force field for molecular mechanics and molecular dynamics simulations. Journal of the American chemical society, 114(25):10024–10035, 1992.

Danilo Jimenez Rezende, Sébastien Racanière, Irina Higgins, and Peter Toth. Equivariant hamiltonian flows, 2019. URL `https://arxiv.org/abs/1909.13739`.

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

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In International conference on machine learning, pp. 9323–9332. PMLR, 2021b.

Arne Schneuing, Yuanqi Du, Charles Harris, Arian Jamasb, Ilia Igashov, Weitao Du, Tom Blundell, Pietro Lió, Carla Gomes, Max Welling, Michael Bronstein, and Bruno Correia. Structure-based drug design with equivariant diffusion models. arXiv preprint arXiv:2210.13695, 2022.

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, 2018.

Morgan Thomas, Robert T Smith, Noel M O'Boyle, Chris de Graaf, and Andreas Bender. Comparison of structure-and ligand-based scoring functions for deep generative models: a gpcr case study. Journal of cheminformatics, 13(1):1–20, 2021.

Oleg Trott and Arthur J Olson. Autodock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of computational chemistry, 31(2):455–461, 2010.

Renxiao Wang, Yipin Lu, and Shaomeng Wang. Comparative evaluation of 11 scoring functions for molecular docking. Journal of medicinal chemistry, 46(12):2287–2303, 2003.

Gregory L Warren, C Webster Andrews, Anna-Maria Capelli, Brian Clarke, Judith LaLonde, Millard H Lambert, Mika Lindvall, Neysa Nevins, Simon F Semus, Stefan Senger, et al. A critical assessment of docking programs and scoring functions. Journal of medicinal chemistry, 49(20): 5912–5931, 2006.

Maciej Wójcikowski, Michał Kukiełka, Marta M Stepniewska-Dziubinska, and Pawel Siedlecki. Development of a protein–ligand extended connectivity (plec) fingerprint and its application for binding affinity predictions. Bioinformatics, 35(8):1334–1341, 2019.

Hong Zhang and Wenjun Zhao. Pnode: A memory-efficient neural ode framework based on high-level adjoint differentiation. arXiv preprint arXiv:2206.01298, 2022.

A THE CONDITIONAL EGNN

The base graph at layer ℓ of the complement graph's EGNN is summarized by the signature $\hat{\mathbf{g}}^\ell$ which depends on both $\{\hat{\mathbf{h}}^\ell_{av}\}$ and the ODE time t:

$$\hat{\mathbf{g}}^0 = \phi^0_g\left(\hat{\mathbf{h}}^1_{av}, \ldots, \hat{\mathbf{h}}^{\hat{L}}_{av}, t\right) \qquad \text{and} \qquad \hat{\mathbf{g}}^\ell = \phi^\ell_g\left(\hat{\mathbf{g}}^{\ell-1}\right) \quad \ell = 1, \ldots, L \tag{12}$$

Note that there are $L+1$ separate functions $\{\phi^\ell_g\}_{\ell=0}^L$.

These invariant base graph signatures $\{\hat{\mathbf{g}}^\ell\}_{\ell=1}^L$ are then naturally incorporated into the Conditional EGNN as follows:

$$\mathbf{m}^\ell_{ij} = \phi_e\left(\mathbf{h}^\ell_i, \mathbf{h}^\ell_j, \|\mathbf{x}^\ell_i - \mathbf{x}^\ell_j\|^2, \|\mathbf{x}^0_i - \mathbf{x}^0_j\|^2, \hat{\mathbf{g}}^\ell, t\right) \qquad b^\ell_{ij} = \sigma(\phi_b(\mathbf{m}^\ell_{ij}, \hat{\mathbf{g}}^\ell)) \qquad \mathbf{m}^\ell_i = \sum_{j=1}^N b^\ell_{ij}\mathbf{m}^\ell_{ij}$$

$$\mathbf{x}^{\ell+1}_i = \mathbf{x}^\ell_i + \left(\sum_{j\neq i} \frac{(\mathbf{x}^\ell_i - \mathbf{x}^\ell_j)}{\|\mathbf{x}^\ell_i - \mathbf{x}^\ell_j\| + 1}\right)\phi_x(\mathbf{m}^\ell_{ij}, \hat{\mathbf{g}}^\ell) \qquad \mathbf{h}^{\ell+1}_i = \mathbf{h}^\ell_i + \phi_h(\mathbf{h}^\ell_i, \mathbf{m}^\ell_i, \hat{\mathbf{g}}^\ell) \tag{13}$$

The particular Conditional EGNN is thus specified by the functions $\{\phi^\ell_g\}_{\ell=0}^L, \phi_e, \phi_b, \phi_x, \phi_h$.

B PROOF OF THE INVARIANT VERTEX DISTRIBUTION THEOREM

In this section, we prove the Invariant Vertex Distribution Theorem presented in Section 2.4. We begin with two lemmata, after which the proof of the theorem is presented.

Lemma 1. *Let the mean position of the base graph be given by $\hat{\mathbf{x}}_{av} = \frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}}\hat{\mathbf{x}}_i$, and define the following quantities*

$$\alpha = \frac{N}{N+\hat{N}} \qquad \Omega_{\hat{G}} = \begin{bmatrix} \mathbf{I}_{3N} - \frac{\alpha}{N}\mathbf{1}_{N\times N}\otimes\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{d_h N}\end{bmatrix} \qquad \omega_{\hat{G}} = \begin{bmatrix} -(1-\alpha)\mathbf{1}_{N\times 1}\otimes\hat{\mathbf{x}}_{av} \\ \mathbf{0}\end{bmatrix}$$

where $\otimes$ indicates the Kronecker product. Given the following mapping:

$$\mathbf{v} = \Omega^{-1}_{\hat{G}}\left(\mathbf{u} - \omega_{\hat{G}}\right) \tag{14}$$

Let the inverse mapping be denoted by $\Gamma^1_{\hat{G}}$, i.e. $\mathbf{u} = \Gamma^1_{\hat{G}}(\mathbf{v})$. For any rigid transformation $T \in E(3)$, which consists of both a rotation and a translation, denote the transformation consisting only of the rotation of T as $T_{rot} \in O(3)$. Then

$$\Gamma^1_{T\hat{G}}(T\mathbf{v}) = T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v}).$$

Furthermore, for any permutations $\pi \in \mathbb{S}_N$ and $\hat{\pi} \in \mathbb{S}_{\hat{N}}$, then

$$\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}) = \pi\Gamma^1_{\hat{G}}(\mathbf{v}).$$

Proof: The mapping $\mathbf{u} = \Gamma^1_{\hat{G}}(\mathbf{v})$ is given by

$$\mathbf{u} = \Omega_{\hat{G}}\mathbf{v} + \omega_{\hat{G}} \tag{15}$$

Let us denote the parts of $\mathbf{u}$ corresponding to the coordinates and the features as $\mathbf{x}^\mathbf{u}$ and $\mathbf{h}^\mathbf{u}$, respectively; and use similar notation for $\mathbf{v}$. Then we have that

$$\mathbf{h}^\mathbf{u} = \mathbf{h}^\mathbf{v} \tag{16}$$

and

$$\mathbf{x}^\mathbf{u} = \left(\mathbf{I}_{3N} - \frac{\alpha}{N}\mathbf{1}_{N\times N}\otimes\mathbf{I}_3\right)\mathbf{x}^\mathbf{v} - (1-\alpha)\mathbf{1}_{N\times 1}\otimes\hat{\mathbf{x}}_{av} \tag{17}$$

Breaking down this last equation by vertex gives

$$\begin{aligned}\mathbf{x}^\mathbf{u}_i &= \mathbf{x}^\mathbf{v}_i - \frac{\alpha}{N}\sum_{j=1}^N \mathbf{x}^\mathbf{v}_j - (1-\alpha)\hat{\mathbf{x}}_{av} \\ &= \mathbf{x}^\mathbf{v}_i - (\alpha\mathbf{x}^\mathbf{v}_{av} + (1-\alpha)\hat{\mathbf{x}}_{av}) \\ &= \mathbf{x}^\mathbf{v}_i - \bar{\mathbf{x}}^\mathbf{v}\end{aligned} \tag{18}$$

where $\mathbf{x}^{\mathbf{v}}_{av}$ is the average coordinate position of $\mathbf{x}^{\mathbf{v}}$, and $\bar{\mathbf{x}}^{\mathbf{v}}$ indicates the average of all vertices in the entire complex, i.e. taking both the complement graph and the base graph together.

Now, let us examine what happens when we apply the rigid transformation T to both $\mathbf{v}$ and the base graph $\hat{G}$; that is, let us examine

$$\tilde{\mathbf{u}} = \Gamma^1_{T\hat{G}}(T\mathbf{v}) \tag{19}$$

In the case of the features $\mathbf{h}$, they are invariant by design; thus

$$\begin{aligned} \mathbf{h}^{\tilde{\mathbf{u}}} &= \mathbf{h}^{\mathbf{Tv}} \\ &= \mathbf{h}^{\mathbf{v}} \\ &= \mathbf{h}^{\mathbf{u}} \end{aligned} \tag{20}$$

where the last line follows from Equation (16). In the case of the coordinates, the transformation is as follows:

$$\mathbf{x}^{T\mathbf{v}}_i = R\mathbf{x}^{\mathbf{v}}_i + t \tag{21}$$

where $R \in O(3)$ is the rotation matrix, and $t \in \mathbb{R}^3$ the translation vector, corresponding to rigid motion T. As we apply T to the base graph $\hat{G}$, this has the effect of applying this transformation to each of the base graph vertices, and hence to their mean and the mean of the entire complex:

$$\hat{\mathbf{x}}^{T\hat{G}}_{av} = R\hat{\mathbf{x}}^{\hat{G}}_{av} + t \quad \Rightarrow \quad \bar{\mathbf{x}}^{\mathbf{Tv}} = R\bar{\mathbf{x}}^{\mathbf{v}} + t \tag{22}$$

Thus, following Equation (18), and substituting $T\mathbf{v}$ and $T\hat{G}$ in place of $\mathbf{v}$ and $\hat{G}$, we get

$$\begin{aligned} \mathbf{x}^{\tilde{\mathbf{u}}}_i &= \mathbf{x}^{T\mathbf{v}}_i - \bar{\mathbf{x}}^{T\mathbf{v}} \\ &= R\mathbf{x}^{\mathbf{v}}_i + t - (R\bar{\mathbf{x}}^{\mathbf{v}} + t) \\ &= R\left(\mathbf{x}^{\mathbf{v}}_i - \bar{\mathbf{x}}^{\mathbf{v}}\right) \\ &= R\mathbf{x}^{\mathbf{u}}_i \end{aligned} \tag{23}$$

Combining Equations (20) and (23), we have that

$$\tilde{\mathbf{u}} = T_{rot}\mathbf{u} \tag{24}$$

Since $\mathbf{u} = \Gamma^1_{\hat{G}}(\mathbf{v})$ and $\tilde{\mathbf{u}} = \Gamma^1_{T\hat{G}}(T\mathbf{v})$, we have shown that $\Gamma^1_{T\hat{G}}(T\mathbf{v}) = T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v})$, as desired.

In the case of the permutations, let us now set

$$\tilde{\mathbf{u}} = \Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}) \tag{25}$$

It is easy to see that $\hat{\pi}$ has no effect; the only place the base graph enters is through the quantities $\hat{N}$ and $\hat{\mathbf{x}}_{av}$, both of which are permutation-invariant. For the features, we now have

$$\begin{aligned} \mathbf{h}^{\tilde{\mathbf{u}}} &= \mathbf{h}^{\pi\mathbf{v}} \\ &= \pi\mathbf{h}^{\mathbf{v}} \\ &= \pi\mathbf{h}^{\mathbf{u}} \end{aligned} \tag{26}$$

That is, the features are simply reordered according to π. With regard to the coordinates, we have that

$$\begin{aligned} \mathbf{x}^{\tilde{\mathbf{u}}}_i &= \mathbf{x}^{\pi\mathbf{v}}_i - \bar{\mathbf{x}}^{\pi\mathbf{v}} \\ &= \mathbf{x}^{\mathbf{v}}_{\pi(i)} - \bar{\mathbf{x}}^{\mathbf{v}} \\ &= \mathbf{x}^{\mathbf{u}}_{\pi(i)} \end{aligned} \tag{27}$$

The coordinates are also therefore simply reordered according to π. Summarizing, we have that

$$\tilde{\mathbf{u}} = \pi\mathbf{u} \tag{28}$$

This is exactly equal to

$$\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}) = \pi\Gamma^1_{\hat{G}}(\mathbf{v})$$

which concludes the proof. □

Lemma 2. *Let $\mathbf{u}(1)$ be the output of a Conditional Flow specified by the Complex-to-Complement Mapping γ which is rotation semi-equivariant and permutation semi-equivariant. This Conditional Flow maps the initial condition $\mathbf{z}$ to $\mathbf{u}(1)$; let the inverse mapping be denoted by $\Gamma^2_{\hat{G}}$, i.e. $\mathbf{z} = \Gamma^2_{\hat{G}}(\mathbf{u}(1))$. Then*

$$\Gamma^2_{T\hat{G}}(T_{rot}\mathbf{u}) = T_{rot}\Gamma^2_{\hat{G}}(\mathbf{u})$$

Furthermore, for any permutations $\pi \in \mathbb{S}_N$ and $\hat{\pi} \in \mathbb{S}_{\hat{N}}$, then

$$\Gamma^2_{\hat{\pi}\hat{G}}(\pi\mathbf{u}) = \pi\Gamma^2_{\hat{G}}(\mathbf{u})$$

Proof: Our first goal is to show that $\Gamma^2_{T\hat{G}}(T_{rot}\mathbf{u}) = T_{rot}\Gamma^2_{\hat{G}}(\mathbf{u})$. Let us define $F_{\hat{G}}$ to be the inverse of $\Gamma^2_{\hat{G}}$, and let $\mathbf{u} = F_{\hat{G}}(\mathbf{z})$. Then

$$\begin{aligned}
\Gamma^2_{T\hat{G}}(T_{rot}\mathbf{u}) = T_{rot}\Gamma^2_{\hat{G}}(\mathbf{u}) \quad &\Leftrightarrow \quad F^{-1}_{T\hat{G}}(T_{rot}F_{\hat{G}}(\mathbf{z})) = T_{rot}F^{-1}_{\hat{G}}(F_{\hat{G}}(\mathbf{z})) \\
&\Leftrightarrow \quad F^{-1}_{T\hat{G}}(T_{rot}F_{\hat{G}}(\mathbf{z})) = T_{rot}\mathbf{z} \\
&\Leftrightarrow \quad F_{T\hat{G}}(T_{rot}\mathbf{z}) = T_{rot}F_{\hat{G}}(\mathbf{z})
\end{aligned} \tag{29}$$

Thus, it is sufficient to show that $F_{T\hat{G}}(T_{rot}\mathbf{z}) = T_{rot}F_{\hat{G}}(\mathbf{z})$. For convenience, we shall set

$$\mathbf{u}(1) = F_{\hat{G}}(\mathbf{z}) \quad \text{and} \quad \tilde{\mathbf{u}}(1) = F_{T\hat{G}}(T_{rot}\mathbf{z}) \tag{30}$$

In this case, $\mathbf{u}(1)$ is defined by the ODE

$$\frac{d\mathbf{u}}{dt} = \texttt{vec}\left(\gamma(G_{\mathbf{u}}, \hat{G})\right) \quad \text{with } \mathbf{u}(0) = \mathbf{z} \tag{31}$$

whereas $\tilde{\mathbf{u}}(1)$ is defined by the ODE

$$\frac{d\tilde{\mathbf{u}}}{dt} = \texttt{vec}\left(\gamma(G_{\tilde{\mathbf{u}}}, T\hat{G})\right) \quad \text{with } \tilde{\mathbf{u}}(0) = T_{rot}\mathbf{z} \tag{32}$$

Now, let us define $\breve{\mathbf{u}}(t) = T^{-1}_{rot}\tilde{\mathbf{u}}(t)$, so that $\tilde{\mathbf{u}}(t) = T_{rot}\breve{\mathbf{u}}(t)$. In this case, we have that:

1. $\breve{\mathbf{u}}(0) = T^{-1}_{rot}\tilde{\mathbf{u}}(0) = T^{-1}_{rot}T_{rot}\mathbf{z} = \mathbf{z}$.
2. $\frac{d\tilde{\mathbf{u}}}{dt} = T_{rot}\frac{d\breve{\mathbf{u}}}{dt}$.
3. $\texttt{vec}\left(\gamma(G_{\tilde{\mathbf{u}}}, T\hat{G})\right) = \texttt{vec}\left(\gamma(G_{T_{rot}\breve{\mathbf{u}}}, T\hat{G})\right) = T_{rot}\texttt{vec}\left(\gamma(G_{\breve{\mathbf{u}}}, \hat{G})\right)$, where the last equality is from the definition of rotation semi-equivariance of γ.

Plugging the above three results into the flow for $\tilde{\mathbf{u}}$ in Equation (32) yields

$$\begin{aligned}
T_{rot}\frac{d\breve{\mathbf{u}}}{dt} = T_{rot}\texttt{vec}\left(\gamma(G_{\breve{\mathbf{u}}}, \hat{G})\right) \quad &\text{with} \quad \breve{\mathbf{u}}(0) = \mathbf{z} \\
\Rightarrow \quad \frac{d\breve{\mathbf{u}}}{dt} = \texttt{vec}\left(\gamma(G_{\breve{\mathbf{u}}}, \hat{G})\right) \quad &\text{with} \quad \breve{\mathbf{u}}(0) = \mathbf{z}
\end{aligned} \tag{33}$$

But this is precisely identical to the flow described in Equation (31); thus, we have that

$$\breve{\mathbf{u}}(t) = \mathbf{u}(t) \quad \text{for all } t \tag{34}$$

But $\tilde{\mathbf{u}}(t) = T_{rot}\breve{\mathbf{u}}(t)$ so that $\tilde{\mathbf{u}}(t) = T_{rot}\mathbf{u}(t)$, and in particular $\tilde{\mathbf{u}}(1) = T_{rot}\mathbf{u}(1)$. Comparing with Equation (30) completes rigid motion part of the proof.

Let us now turn to permutations; the proof is similar, but we repeat it in full for completeness. Our goal is to show that $\Gamma^2_{\hat{\pi}\hat{G}}(\pi\mathbf{u}) = \pi\Gamma^2_{\hat{G}}(\mathbf{u})$. Let us define $F_{\hat{G}}$ to be the inverse of $\Gamma^2_{\hat{G}}$, and let $\mathbf{u} = F_{\hat{G}}(\mathbf{z})$. Then

$$\begin{aligned}
\Gamma^2_{\hat{\pi}\hat{G}}(\pi\mathbf{u}) = \pi\Gamma^2_{\hat{G}}(\mathbf{u}) \quad &\Leftrightarrow \quad F^{-1}_{\hat{\pi}\hat{G}}(\pi F_{\hat{G}}(\mathbf{z})) = \pi F^{-1}_{\hat{G}}(F_{\hat{G}}(\mathbf{z})) \\
&\Leftrightarrow \quad F^{-1}_{\hat{\pi}\hat{G}}(\pi F_{\hat{G}}(\mathbf{z})) = \pi\mathbf{z} \\
&\Leftrightarrow \quad F_{\hat{\pi}\hat{G}}(\pi\mathbf{z}) = \pi F_{\hat{G}}(\mathbf{z})
\end{aligned} \tag{35}$$

Thus, it is sufficient to shows that $F_{\hat{\pi}\hat{G}}(\pi\mathbf{z}) = \pi F_{\hat{G}}(\mathbf{z})$. For convenience, we shall set

$$\mathbf{u}(1) = F_{\hat{G}}(\mathbf{z}) \quad \text{and} \quad \tilde{\mathbf{u}}(1) = F_{\hat{\pi}\hat{G}}(\pi\mathbf{z}) \tag{36}$$

In this case, $\mathbf{u}(1)$ is defined by the ODE

$$\frac{d\mathbf{u}}{dt} = \texttt{vec}\left(\gamma(G_{\mathbf{u}}, \hat{G})\right) \quad \text{with } \mathbf{u}(0) = \mathbf{z} \tag{37}$$

whereas $\tilde{\mathbf{u}}(1)$ is defined by the ODE

$$\frac{d\tilde{\mathbf{u}}}{dt} = \texttt{vec}\left(\gamma(G_{\tilde{\mathbf{u}}}, \hat{\pi}\hat{G})\right) \quad \text{with } \tilde{\mathbf{u}}(0) = \pi\mathbf{z} \tag{38}$$

Now, let us define $\check{\mathbf{u}}(t) = \pi^{-1}\tilde{\mathbf{u}}(t)$, so that $\tilde{\mathbf{u}}(t) = \pi\check{\mathbf{u}}(t)$. In this case, we have that:

1. $\check{\mathbf{u}}(0) = \pi^{-1}\tilde{\mathbf{u}}(0) = \pi^{-1}\pi\mathbf{z} = \mathbf{z}$.
2. $\frac{d\tilde{\mathbf{u}}}{dt} = \pi\frac{d\check{\mathbf{u}}}{dt}$.
3. $\texttt{vec}\left(\gamma(G_{\tilde{\mathbf{u}}}, \hat{\pi}\hat{G})\right) = \texttt{vec}\left(\gamma(G_{\pi\check{\mathbf{u}}}, \hat{\pi}\hat{G})\right) = \pi\texttt{vec}\left(\gamma(G_{\check{\mathbf{u}}}, \hat{G})\right)$, where the last equality is from the definition of permutation semi-equivariance of γ.

Plugging the above three results into the flow for $\tilde{\mathbf{u}}$ in Equation (38) yields

$$\begin{aligned} \pi\frac{d\check{\mathbf{u}}}{dt} &= \pi\texttt{vec}\left(\gamma(G_{\check{\mathbf{u}}}, \hat{G})\right) \quad \text{with} \quad \check{\mathbf{u}}(0) = \mathbf{z} \\ \Rightarrow \quad \frac{d\check{\mathbf{u}}}{dt} &= \texttt{vec}\left(\gamma(G_{\check{\mathbf{u}}}, \hat{G})\right) \quad \text{with} \quad \check{\mathbf{u}}(0) = \mathbf{z} \end{aligned} \tag{39}$$

But this is precisely identical to the flow described in Equation (37); thus, we have that

$$\check{\mathbf{u}}(t) = \mathbf{u}(t) \quad \text{for all } t \tag{40}$$

But $\tilde{\mathbf{u}}(t) = \pi\check{\mathbf{u}}(t)$ so that $\tilde{\mathbf{u}}(t) = \pi\mathbf{u}(t)$, and in particular $\tilde{\mathbf{u}}(1) = \pi\mathbf{u}(1)$. Comparing with Equation (36) completes the proof. □

Theorem. *Let $\mathbf{u}(1)$ be the output of a Conditional Flow specified by the Complex-to-Complement Mapping γ. Let the mean position of the base graph be given by $\hat{\mathbf{x}}_{av} = \frac{1}{\hat{N}}\sum_{i=1}^{\hat{N}}\hat{\mathbf{x}}_i$, and define the following quantities*

$$\alpha = \frac{N}{N+\hat{N}} \qquad \Omega_{\hat{G}} = \begin{bmatrix} \mathbf{I}_{3N} - \frac{\alpha}{N}\mathbf{1}_{N\times N}\otimes\mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{d_h N} \end{bmatrix} \qquad \omega_{\hat{G}} = \begin{bmatrix} -(1-\alpha)\mathbf{1}_{N\times 1}\otimes\hat{\mathbf{x}}_{av} \\ \mathbf{0} \end{bmatrix} \tag{41}$$

where $\otimes$ indicates the Kronecker product. Finally, let

$$\mathbf{v} = \Omega_{\hat{G}}^{-1}\left(\mathbf{u}(1) - \omega_{\hat{G}}\right) \tag{42}$$

Suppose that γ is both rotation semi-equivariant and permutation semi-equivariant. Then the resulting distribution on $\mathbf{v}$, that is $p_{vec}(\mathbf{v}|\hat{G})$, yields a vertex distribution $p(V|N,\hat{G}) = p_{vec}(\texttt{vec}(V)|\hat{G})$ that satisfies the invariance conditions in Equation (1).

Proof: The Conditional Flow maps from the Gaussian random variable $\mathbf{z}$ to the variable $\mathbf{u}(1)$. As this flow is a normalizing flow, it is invertible, so let us denote the inverse mapping by Γ^2:

$$\mathbf{z} = \Gamma^2_{\hat{G}}(\mathbf{u}(1)) \tag{43}$$

Note that the dependence on the base graph $\hat{G}$ is denoted using a subscript, as the invertibility does not apply to the base graph, but only to the complement graph. Equation (42) maps from the variable $\mathbf{u}(1)$ to the variable $\mathbf{v}$; let us denote its inverse mapping by Γ^1:

$$\mathbf{u}(1) = \Gamma^1_{\hat{G}}(\mathbf{v}) \tag{44}$$

In this case, we have that

$$\mathbf{z} = \Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v})) \equiv \Gamma_{\hat{G}}(\mathbf{v}) \tag{45}$$

Now, our goal is to show that the following condition holds:

$$p(TV|N, T\hat{G}) = p(V|N, \hat{G}) \quad \text{for } T \in E(3) \tag{46}$$

Using the p_{vec} notation, this translates to

$$p_{vec}(T\mathbf{v}|T\hat{G}) = p_{vec}(\mathbf{v}|\hat{G}) \tag{47}$$

Now, from Equation (45), the fact that Γ is invertible, and the change of variables formula, we have that

$$p_{vec}(\mathbf{v}|\hat{G}) = p_{\mathbf{z}}(\Gamma_{\hat{G}}(\mathbf{v}))|\det J_{\Gamma_{\hat{G}}}(\mathbf{v})| \tag{48}$$

where $p_{\mathbf{z}}(\cdot)$ is the Gaussian distribution from $\mathbf{z}$ is sampled; and $J_{\Gamma_{\hat{G}}}(\cdot)$ is the Jacobian of $\Gamma_{\hat{G}}(\cdot)$. Since $\Gamma_{\hat{G}} = \Gamma^2_{\hat{G}} \circ \Gamma^1_{\hat{G}}$, this can be expanded as

$$p_{vec}(\mathbf{v}|\hat{G}) = p_{\mathbf{z}}(\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v})))|\det J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))||\det J_{\Gamma^1_{\hat{G}}}(\mathbf{v})| \tag{49}$$

using the chain rule, and the fact that determinant of a product is the product of determinants. Plugging this into Equation (47), we must show that

$$\begin{aligned} & p_{\mathbf{z}}(\Gamma^2_{T\hat{G}}(\Gamma^1_{T\hat{G}}(T\mathbf{v})))|\det J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{T\hat{G}}(T\mathbf{v}))||\det J_{\Gamma^1_{T\hat{G}}}(T\mathbf{v})| \\ = \quad & p_{\mathbf{z}}(\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v})))|\det J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))||\det J_{\Gamma^1_{\hat{G}}}(\mathbf{v})| \quad \text{for } T \in E(3) \end{aligned} \tag{50}$$

A rigid transformation $T \in E(3)$ consists of both a rotation and a translation. For brevity, denote the transformation consisting only of the rotation of T as $T_{rot} \in O(3)$. Now, from Lemma 1, we have that

$$\Gamma^1_{T\hat{G}}(T\mathbf{v}) = T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v}) \tag{51}$$

From Lemma 2, we have that

$$\Gamma^2_{T\hat{G}}(T_{rot}\mathbf{u}) = T_{rot}\Gamma^2_{\hat{G}}(\mathbf{u}) \tag{52}$$

Combining Equations (51) and (52) gives that

$$\begin{aligned} p_{\mathbf{z}}(\Gamma^2_{T\hat{G}}(\Gamma^1_{T\hat{G}}(T\mathbf{v}))) &= p_{\mathbf{z}}(\Gamma^2_{T\hat{G}}(T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v}))) \\ &= p_{\mathbf{z}}(T_{rot}\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v}))) \\ &= p_{\mathbf{z}}(\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v}))) \end{aligned} \tag{53}$$

where the last line follows from the rotation invariance of the Gaussian distribution.

Note that

$$\begin{aligned} J_{\Gamma^1_{T\hat{G}}}(T\mathbf{v}) &= \frac{\partial}{\partial \mathbf{v}}\left(\Gamma^1_{T\hat{G}}(T\mathbf{v})\right) \\ &= \frac{\partial}{\partial \mathbf{v}}\left(T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v})\right) \\ &= T_{rot}\frac{\partial}{\partial \mathbf{v}}\left(\Gamma^1_{\hat{G}}(\mathbf{v})\right) \\ &= T_{rot}J_{\Gamma^1_{\hat{G}}}(\mathbf{v}) \end{aligned} \tag{54}$$

Now, T_{rot} can be represented by the $d_v^N \times d_v^N$ block diagonal matrix given by

$$T_{rot} = \begin{bmatrix} \mathbf{1}_{N\times 1} \otimes R & 0 \\ 0 & \mathbf{I}_{d_h N} \end{bmatrix} \tag{55}$$

where $R \in O(3)$, the top-left block corresponds to the coordinates $\mathbf{x}$ and the bottom-right block corresponds to the feature $\mathbf{h}$. Thus,

$$\begin{aligned} \det\Big(J_{\Gamma^1_{T\hat{G}}}(T\mathbf{v})\Big) &= \det\Big(T_{rot}J_{\Gamma^1_{\hat{G}}}(\mathbf{v})\Big) \\ &= \det(T_{rot})\det\Big(J_{\Gamma^1_{\hat{G}}}(\mathbf{v})\Big) \\ &= \det(R)^N \det(\mathbf{I}_{d_h N})\det\Big(J_{\Gamma^1_{\hat{G}}}(\mathbf{v})\Big) \\ &= \pm\det\Big(J_{\Gamma^1_{\hat{G}}}(\mathbf{v})\Big) \end{aligned} \tag{56}$$

where the second line follows from the fact that the determinant of a product is the product of determinants; the third line from the fact that the determinant of a block diagonal matrix is the product of the determinants of the blocks; and the fourth line from the fact that the determinant of a rotation matrix is ± 1.

To simplify $J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{T\hat{G}}(T\mathbf{v}))$, note that

$$\begin{aligned}\Gamma^2_{T\hat{G}}(\mathbf{u}) &= \Gamma^2_{T\hat{G}}(T_{rot}T_{rot}^{-1}\mathbf{u}) \\ &= T_{rot}\Gamma^2_{\hat{G}}(T_{rot}^{-1}\mathbf{u})\end{aligned} \tag{57}$$

where we have used Equation (52). Thus,

$$\begin{aligned}J_{\Gamma^2_{T\hat{G}}}(\mathbf{u}) &= \frac{\partial}{\partial \mathbf{u}}\left(\Gamma^2_{T\hat{G}}(\mathbf{u})\right) \\ &= \frac{\partial}{\partial \mathbf{u}}\left(T_{rot}\Gamma^2_{\hat{G}}(T_{rot}^{-1}\mathbf{u})\right) \\ &= T_{rot}J_{\Gamma^2_{T\hat{G}}}(T_{rot}^{-1}\mathbf{u})T_{rot}^{-1}\end{aligned} \tag{58}$$

We wish to plug in $\mathbf{u} = \Gamma^1_{T\hat{G}}(T\mathbf{v})$. Note that from Equation (51), $\Gamma^1_{T\hat{G}}(T\mathbf{v}) = T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v})$. Thus,

$$\begin{aligned}J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{T\hat{G}}(T\mathbf{v})) &= J_{\Gamma^2_{T\hat{G}}}(T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v})) \\ &= T_{rot}J_{\Gamma^2_{T\hat{G}}}(T_{rot}^{-1}T_{rot}\Gamma^1_{\hat{G}}(\mathbf{v}))T_{rot}^{-1} \\ &= T_{rot}J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))T_{rot}^{-1}\end{aligned} \tag{59}$$

where in the second line we substituted Equation (58). Taking determinants gives

$$\begin{aligned}\det\left(J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{T\hat{G}}(T\mathbf{v}))\right) &= \det\left(T_{rot}J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))T_{rot}^{-1}\right) \\ &= \det\left(T_{rot}^{-1}T_{rot}J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))\right) \\ &= \det\left(J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))\right)\end{aligned} \tag{60}$$

where in the second line, we used the fact that permuting the order of a matrix multiplication does not affect the determinant.

Combining Equations (53), (56), and (60), we finally arrive at:

$$\begin{aligned}&p_{\mathbf{z}}(\Gamma^2_{T\hat{G}}(\Gamma^1_{T\hat{G}}(T\mathbf{v})))|\det J_{\Gamma^2_{T\hat{G}}}(\Gamma^1_{T\hat{G}}(T\mathbf{v}))||\det J_{\Gamma^1_{T\hat{G}}}(T\mathbf{v})| \\ &\quad = \quad p_{\mathbf{z}}(\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v})))|\det J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))||\det J_{\Gamma^1_{\hat{G}}}(\mathbf{v})|\end{aligned} \tag{61}$$

which is exactly Equation (50). Thus, we have shown that $p(TV|N, T\hat{G}) = p(V|N, \hat{G})$, as desired.

Let us turn now to the permutation case, which is quite similar. Similar to Equation (50), we need to show

$$\begin{aligned}&p_{\mathbf{z}}(\Gamma^2_{\hat{\pi}\hat{G}}(\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v})))|\det J_{\Gamma^2_{\hat{\pi}\hat{G}}}(\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}))||\det J_{\Gamma^1_{\hat{\pi}\hat{G}}}(\pi\mathbf{v})| \\ &\quad = \quad p_{\mathbf{z}}(\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v})))|\det J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))||\det J_{\Gamma^1_{\hat{G}}}(\mathbf{v})| \quad \text{for } \pi \in \mathbb{S}_n \text{ and } \hat{\pi} \in \mathbb{S}_{\hat{N}}\end{aligned} \tag{62}$$

From Lemmata 1 and 2, we have that

$$\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}) = \pi\Gamma^1_{\hat{G}}(\mathbf{v}) \quad \text{and} \quad \Gamma^2_{\hat{\pi}\hat{G}}(\pi\mathbf{u}) = \pi\Gamma^2_{\hat{G}}(\mathbf{u}) \tag{63}$$

Thus

$$\begin{aligned}p_{\mathbf{z}}(\Gamma^2_{\hat{\pi}\hat{G}}(\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}))) &= p_{\mathbf{z}}(\Gamma^2_{\hat{\pi}\hat{G}}(\pi\Gamma^1_{\hat{G}}(\mathbf{v}))) \\ &= p_{\mathbf{z}}(\pi\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v}))) \\ &= p_{\mathbf{z}}(\Gamma^2_{\hat{G}}(\Gamma^1_{\hat{G}}(\mathbf{v})))\end{aligned} \tag{64}$$

where the last line follows from the permutation-invariance of the Gaussian distribution $p_{\mathbf{z}}(\cdot)$.

In a manner parallel to the derivation of Equation (54), we can show that

$$J_{\Gamma^1_{\hat{\pi}\hat{G}}}(\pi\mathbf{v}) = \boldsymbol{\pi} J_{\Gamma^1_{\hat{G}}}(\mathbf{v}) \tag{65}$$

where $\boldsymbol{\pi}$ now indicates the permutation matrix associated with the permutation π. Thus, we have that

$$\det\left(J_{\Gamma^1_{\hat{\pi}\hat{G}}}(\pi\mathbf{v})\right) = \det(\boldsymbol{\pi})\det\left(J_{\Gamma^1_{\hat{G}}}(\mathbf{v})\right) = \pm\det\left(J_{\Gamma^1_{\hat{G}}}(\mathbf{v})\right) \tag{66}$$

where we have used the fact that a permutation matrix has determinant of ± 1. Similarly, in a manner parallel to the derivation of Equation (59), we can show that

$$J_{\Gamma^2_{\hat{\pi}\hat{G}}}(\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v})) = \boldsymbol{\pi} J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))\boldsymbol{\pi}^{-1} \tag{67}$$

so that

$$\begin{aligned}\det\left(J_{\Gamma^2_{\hat{\pi}\hat{G}}}(\Gamma^1_{\hat{\pi}\hat{G}}(\pi\mathbf{v}))\right) &= \det\left(\boldsymbol{\pi} J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))\boldsymbol{\pi}^{-1}\right)\\ &= \det\left(\boldsymbol{\pi}^{-1}\boldsymbol{\pi} J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))\right)\\ &= \det\left(J_{\Gamma^2_{\hat{G}}}(\Gamma^1_{\hat{G}}(\mathbf{v}))\right)\end{aligned} \tag{68}$$

Combining Equations (64), (66), and (68) yields Equation (62); completing the proof. □

C THE EDGE AND PROPERTIES DISTRIBUTIONS

The Edge Distribution Given the invariance conditions described in Equation (1), we propose a distribution which displays *conditional independence*: $p\left(E = (\mathbf{e}_{ij})_{i<j:j\in\eta_i} | N, V, \hat{G}\right) = \prod_{i<j:j\in\eta_i} p(\mathbf{e}_{ij}|N, V, \hat{G})$. We opt for conditional independence for two reasons: (1) The usual Markov decomposition of the probability distribution with terms of the form $p(\mathbf{e}_{ij}|\mathbf{e}_{<ij}, N, V, \hat{G})$ implies a particular ordering of the edges, and is therefore not permutation-invariant. (2) V is a deterministic and invertible function of the flow's noise vector $\mathbf{z}$; thus, conditioning on V is the same as conditioning on $\mathbf{z}$. If E is a deterministic (but not necessarily invertible) function of $\mathbf{z}$, then conditional independence is correct.

To compute $p(\mathbf{e}_{ij}|N, V, \hat{G})$, we use a second Conditional EGNN. The key distinction between this network and the Conditional EGNN used in computing the vertex distribution is the initial conditions. In the case of the vertex distribution, the initial conditions are $\mathbf{x}_i^1 = \mathbf{0}$ and $\mathbf{h}_i^1 = \mathbf{0}$. In the current case of the edge distribution, we are given V (we are conditioning on it); thus, we take the initial conditions to be $\tilde{\mathbf{x}}_i^1 = \mathbf{x}_i(V)$ and $\tilde{\mathbf{h}}_i^1 = \mathbf{h}_i(V)$. In other words, the initial values are given the vertex list V itself.

Given this second Conditional EGNN, we can compute the edge distribution as

$$p\left(\mathbf{e}_{ij}|N, V, \hat{G}\right) = \mathbf{e}_{ij}^T \texttt{MLP}\left(\tilde{\mathbf{m}}_{ij}^L\right) \tag{69}$$

in the case of categorical properties (where MLP's output is a softmax with d_e entries); analogous expressions exist for ordinal or continuous properties. It is straightforward to see that this distribution satisfies the invariance properties in Equation (1). The corresponding loss function is a simple cross-entropy loss (or regression loss for non-categorical properties).

The Property Distribution We propose the following distribution. We use a standard Markov decomposition: $p(A|N, V, E, \hat{G}) = \prod_{k=1}^{K} p\left(\mathbf{a}_k|\mathbf{a}_{1:(k-1)}, N, V, E, \hat{G}\right)$. Let

$$\boldsymbol{\xi}_h = \frac{1}{N}\sum_{i=1}^{N}\texttt{MLP}\left(\tilde{\mathbf{h}}_i^L\right) \qquad \boldsymbol{\xi}_e = \frac{1}{|E|}\sum_{i<j:j\in\eta_i}\texttt{MLP}\left(\mathbf{e}_{ij}\right) \qquad \boldsymbol{\xi}_{a,k} = \texttt{MLP}\left(\sum_{j=1}^{k-1} W_j\mathbf{a}_j\right) \tag{70}$$

where the matrices $W_1, \ldots W_K$ all have the same number of rows. Then we set

$$p\left(\mathbf{a}_k \middle| \mathbf{a}_{1:(k-1)}, N, V, E, \hat{G}\right) = \mathbf{a}_k^T \texttt{MLP}\left(\texttt{concat}\left(\boldsymbol{\xi}_h, \boldsymbol{\xi}_e, \boldsymbol{\xi}_{a,k}\right)\right) \tag{71}$$

in the case of categorical properties; analogous expressions exist for ordinal or continuous properties. Note that the only item which changes for the different properties k is the vector $\boldsymbol{\xi}_{a,k}$. It is easy to see that this distribution satisfies the invariance properties in Equation (1). The corresponding loss function is a simple cross-entropy loss (or regression loss for non-categorical properties).

D EXPERIMENTAL SETUP

Molecular Dataset Datasets with a large number of receptor-ligand complexes are critical to our endeavour. Many models have relied on the high quality PDBbind dataset which curates the Protein Data Bank (PDB) (Liu et al., 2017); however, for the training of generative models, this dataset is relatively small. CrossDocked2020 (Francoeur et al., 2020) is the first large-scale standardized dataset for training ML models with ligand poses cross-docked against non-cognate receptor structure, greatly expanding the number of poses available for training. The dataset is organized by clustering of similar binding pockets across the PDB; each cluster contains ligands cross-docked against all receptors in the pocket. Each receptor-ligand structure also contains information indicating the nature of the docked pair, such as root mean squared deviation (RMSD) to the reference crystal pose and Vina cross-docking score (Trott & Olson, 2010) as implemented in Smina (Koes et al., 2013). The dataset contains 22.5 million poses of ligands docked into multiple similar binding pockets across the PDB. We use the authors' suggested split into training and validation sets (Francoeur et al., 2020). This dataset contains docked receptor-ligand pairs whose binding pose RMSD is lower than 2Å. Based on considerations of training duration, we keep only those data points whose ligand has 30 atoms or fewer with atom types in {C, N, O, F}. We also keep only data points with potentially valid ligand-protein binding properties, i.e. whose ligands do not contain duplicate vertices and whose predicted Vina scores (Trott & Olson, 2010) are within distribution. The refined datasets consist of 132,863 training data points and 63,929 validation data points.

Features The ligand features that we wish to predict include the atom type $\in$ {C, N, O, F} (categorical); the stereo parity $\in$ {not stereo, odd, even} (categorical); and charge $\in \{-1, 0, +1\}$ (ordinal). The receptor features that are used are computed with the Graphein library (Jamasb et al., 2022). The vertex features include: the atom type $\in$ {C, N, O, S, "other"} where "other" is a catch-all for less common atom types[2] (categorical); the Meiler Embeddings (Meiler et al., 2001) (continuous $\in \mathbb{R}^7$). The bond (edge) properties include: the bond order $\in$ {Single, Double, Triple} (categorical); covalent bond length (continuous $\in \mathbb{R}$). The receptor overall graph properties ($\hat{A}$) contain the weight of all chains contained within a polypeptide structure, see (Jamasb et al., 2022).

Training Training the model takes approximately 18 days using a single NVIDIA A100 GPU for 39 epochs. We proactively stopped the training procedure when the negative log-likelihood (NLL) term reached a low improvement rate of between epochs. We train with the Adam optimizer, weight decay of 10^{-12}, batch size of 128, and learning rate of 2×10^{-4}. To improve the stability of the continuous flow model and to deal with ODE stiffness issues we use the ODE regularization term described in (Finlay et al., 2020) with a value of 10^{-3}; and perform gradient clipping, where the clipping term is set by calculating a moving average of 50 steps of the normalizing flows gradient-norm. This is parallel to the treatment in (Satorras et al., 2021a). We use dopri5 (Runge-Kutta 4(5) of Dormand-Prince) as the ODE solver with relative (rtol) and absolute (atol) error tolerance values of 10^{-4}.

E DETAILED EXPERIMENTAL RESULTS

We first evaluate the utility of our technique as a conditional generative model. Given a receptor, we generate ligands which may successfully bind to that receptor. To show that our method is competitive as a generative model we compare against Liu et al. (2022) – one of the recent works that showed promising results in similar settings. We then demonstrate the utility of our technique for indicating the nature of the interaction of ligands with their corresponding binding site. The trained model provides the conditional probability, $p(G|\hat{G})$, which defines the likelihood of a binding event occurring given a ligand, based on the presence of a specific target receptor.

Conditional Generative Model We perform inference using the method suggested in the baseline (Liu et al., 2022). Given a receptor, we sample from the learned distribution, which generates the ligands' vertices; we then then apply OpenBabel (Hummell et al., 2021) to construct bonds. Evaluation follows the standard procedure (Ragoza et al., 2022; Liu et al., 2022). First, the receptor target is computed by taking all of the atoms in the receptor that are less than 15Å from the center of mass of the reference ligand. We then generate 100 ligands for each reference binding site in

[2]Specifically: Na, Mg, P, Cl, K, Ca, Co, Cu, Zn, Se, Cd, I, Hg.

Validity	
Ours	GraphBP
99.87%	99.75%

Bond Length Distribution			
	Ref. Mols.	Ours	GraphBP
mean	1.42	1.45	1.65
std	0.08	0.10	0.95

(a) Ligand validity and bond length distribution

ΔBinding	
Ours	GraphBP
35.7%	22.76%

Predicted Affinity Distribution			
	Ref. Mols.	Ours	GraphBP
mean	5.09	4.56	4.31
std	1.16	1.05	1.03

(b) ΔBinding and predicted affinity distribution

Table 2: Comparison of molecule validity and ΔBinding between proposed method and GraphBP.

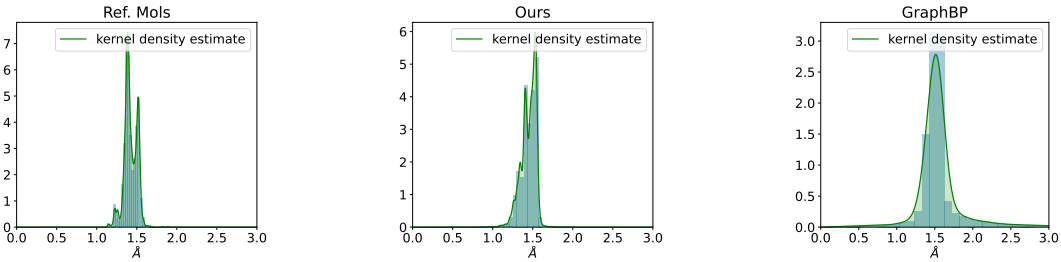

Figure 1: Normalized histogram of relative distances between atoms

the evaluation set, and compute statistics (i.e. validity and ΔBinding, see below) on this set of samples. As in (Ragoza et al., 2022; Liu et al., 2022), 10 target receptors for evaluation; each target receptor has multiple associated ligands, leading to 90 (receptor, reference-ligand) pairs. We train the baseline technique (Liu et al., 2022) on our filtered dataset. More specifically, the training continues for 100 epochs, using the hyperparameters given in the paper, with one exception: we set the atom number range of the autoregressive generative process according to the atom distribution of the filtered dataset.

Validity The validity is defined as the percentage of molecules that are chemically valid among all generated molecules. A molecule is valid if it can be sanitized by RDKit; for an explanation of the sanitization procedure, see (Landrum, 2016). As shown in Table 2(a), our model produces ligands with a validity of 99.87%, surpassing the baseline, GraphBP. We also compute the distribution of bond distances of the two methods, and compare this to distribution of the reference ligands; see Figure 1. Our method's distribution is considerably closer to the reference distribution than GraphBP; some non-trivial fraction of the time, GraphBP produces unusual, very high bond distances. (In fact, we have discarded values higher than 10Å on the GraphBP plot so as to display the distributions on similar scales.) This impression is reinforced in Table 2(a) which compares the mean and standard deviation of these distributions.

Binding Affinity A more interesting measure than validity is ΔBinding, which measures the measures the percentage of generated molecules that have higher predicted binding affinity to the target binding site than the corresponding reference molecule. To compute binding affinities, we follow the procedure used by GraphBP. Briefly, we refine the generated 3D molecules by Universal Force Field (UFF) minimization (Rappé et al., 1992); then, Vina minimization and CNN scoring are applied to both generated and reference molecules by using gnina, a molecular docking program (McNutt et al., 2021). As can be seen in Table 2(b), our result improves significantly on the baseline. Raw GraphBP attains ΔBinding = 13.45%. By playing with the minimum and the maximum atom number of the baseline autoregressive model, we were able to improve this to 22.76%; however, note that this results in a reduction in validity from 99.75% to 99.54%. Our method attains ΔBinding = 35.7%, which is a relative improvement of 56.81% over the better of the two GraphBP scores.

Qualitative Results We show examples of generated ligands in Figure 2, along with their chemical structures. Note that the structures of the generated molecules differ substantially from the reference molecules, indicating that the model has indeed learn to generalize to interesting novel structures.

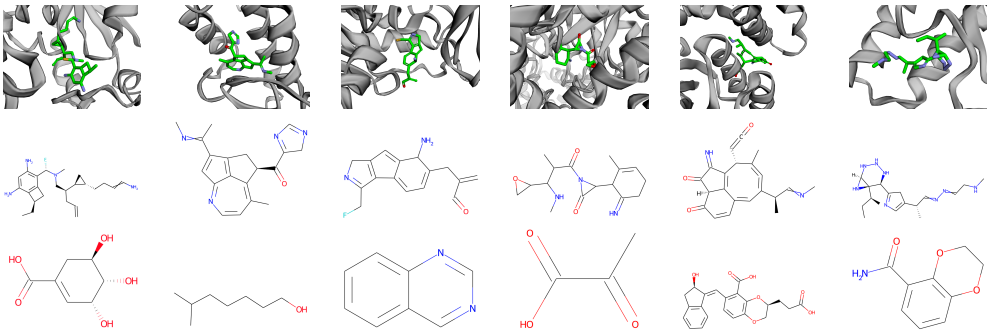

Figure 2: Comparison between generated 3D molecules for target binding-site and reference molecules. Receptor IDs, left to right: 1zyu, 2qu9, 2hw1, 3lcg, 3zt3, 5lvq. Top: generated ligand (colour) + receptor. Middle: generated ligand chemical structure. Bottom: reference ligand chemical structure.

Binding Likelihood The CrossDocked2020 dataset (Francoeur et al., 2020) includes quantitative measures indicating the quality of the binding of each docked receptor-ligand structure: (i) root mean squared deviation (RMSD) to the reference crystal pose; (ii) Vina cross-docking score (Trott & Olson, 2010) as implemented in Smina (Koes et al., 2013). Scoring functions that represent and predict ligand-protein interactions are important for applications in structure-based drug discovery (e.g. energy minimization, molecular dynamics simulations, and hit identification/lead optimization (DeWitte & Shakhnovich, 1996; McInnes, 2007; Charifson et al., 1999)), and particularly important for molecular docking where one seeks to predict the probability that binding occurs given a specified orientation and conformation (i.e. pose) of a ligand with respect to a target receptor (Su et al., 2018; Wang et al., 2003; Kitchen et al., 2004; Warren et al., 2006; Cheng et al., 2009; Huang & Zou, 2011; Trott & Olson, 2010; Cheng et al., 2012). We show how the conditional probability of our model may also be used as such indicator. As we have seen in Section 2, our method is not trained on either of the binding quality indicators (Vina / RMSD), but instead is the result of learning a conditional probability distribution from ligand-receptor pairs; therefore it may be viewed as a complementary scoring method. Specifically, given the conditional probability $p(G|\hat{G})$ from the trained model, its negative log-likelihood (NLL) may be used a scoring method. Such an approach can be considered as a member of the family of knowledge-based methods (Muegge, 2000; Gohlke et al., 2000; Durrant & McCammon, 2011; Huang & Zou, 2011; Ballester & Mitchell, 2010; Hassan et al., 2018; Wójcikowski et al., 2019) which are constructed from entirely non-physical statistical potentials derived from known receptor-ligand complexes.

Vina and RMSD We split the 63,929 validation data points into three different groups according their Vina scores: (a) Vina $\in [0, 5]$ (b) Vina $\in (5, 9]$ (c) Vina $\in (9, 15]$. We randomly sample 10,000 receptor-ligand complexes from each group. Using the trained model, we calculate the average NLL for each group separately, where lower NLL indicates better conditional likelihood. The results in Table 3(a) show good correspondence between the Vina scores (higher values describe better affinity) with the binding likelihood outcomes of our model (lower NLL is better).

We then repeat this experiment, but using RMSD values instead of Vina scores. Again, we divide the validation set into three groups according to their RMSD values: (a) RMSD $\in [0, 1]$ (b) RMSD $\in (1, 1.5]$ (c) RMSD $\in (1, 2]$ (all values in Angstroms). Again, we see a good correspondence between the RMSD values (lower values describe a better connected complex) with the binding likelihood outcomes of our model (lower NLL is better).

Rigid Transformations The nature of binding between receptor and corresponding ligand is affected by the position of the ligand in the binding site. The ligand pose for each ligand-receptor pair in the validation data points was refined using the UFF force-field and then optimized with respect to the receptor structure using the Vina scoring function. We randomly select 10,000 validation data points and apply a rigid body transformation solely to the minimized ligand-graph pose. Any such changes to the ligand pose with respect to the binding site potentially affect and reduce the binding affinity. The results in Table 3(b) show the NLL under varying degrees of rigid transformations. The more significant the transformation, the lower the binding likelihood (i.e. the higher the NLL) gets.

Vina Score ↑			
	0 < Vina < 5	5 < Vina < 9	9 < Vina < 15
NLL ↓	-77.83	-144.30	-180.37

RMSD ↓			
	0 < RMSD < 1	1 < RMSD < 1.5	1.5 < RMSD < 2
NLL ↓	-137.70	-135.72	-130.67

(a) Ligand-receptor Vina score and RMSD vs average negative log-likelihood (NLL).

NLL under rigid transformation ↓						
		Rotation $\theta_x \sim U(-\frac{\pi}{u}, \frac{\pi}{u}), \theta_y \sim U(-\frac{\pi}{2u}, \frac{\pi}{2u}), \theta_z \sim U(-\frac{\pi}{u}, \frac{\pi}{u})$				
			$u = 20$	$u = 10$	$u = \frac{20}{3}$	$u = 5$
Translation $\tau \sim U\left([-t, t]^3\right)$		-134.34	-132.18	-124.15	-110.45	-91.94
	$t = 2$Å	-133.98	-130.80	-122.49	-109.22	-90.75
	$t = 5$Å	-129.28	-126.28	-118.57	-104.94	-86.02
	$t = 10$Å	-113.99	-111.80	-103.74	-91.15	-71.83
	$t = 15$Å	-88.85	-87.26	-76.85	-66.19	-45.89

(b) Average negative log-likelihood (NLL) under varying degrees of 3D rigid body transformations (translation and rotation) of ligand pose. $\theta_x, \theta_y, \theta_z$ are rotational Euler angles and $\tau \in \mathbb{R}^3$ is translation factor.

Table 3: Binding Likelihood. ↓ (↑) indicates stronger binding for lower (higher) values.