Guided docking as a data generation approach facilitates structure-based machine learning on kinases

Michael Backenköhler,[†,§] Joschka Groß,[‡,§] Verena Wolf,[‡] and Andrea Volkamer[*,†,¶]

†*Data Driven Drug Design, Center for Bioinformatics, Saarland University*

‡*Modeling and Simulation, Saarland University*

¶*Structural Bioinformatics and in silico Toxicology, Institute of Physiology, Universitätsmedizin Berlin*

§*equal contribution*

E-mail: volkamer@cs.uni-saarland.de

Abstract

Drug discovery pipelines nowadays rely on machine learning models to explore and evaluate large chemical spaces. While including 3D structural information is considered beneficial, structural models are hindered by the availability of protein-ligand complex structures. Exemplified for kinase drug discovery, we address this issue by generating kinase-ligand complex data using template docking for the kinase compound subset of available ChEMBL assay data. To evaluate the benefit of the created complex data, we use it to train a structure-based E(3)-invariant graph neural network (GNN). Our evaluation shows that binding affinities can be predicted with significantly higher precision by models that take synthetic binding poses into account compared to ligand or drug target interaction models only.

1 Introduction

The search space for new drugs is vast, and obtaining experimental measurements is time-consuming and expensive. Computational methods provide an time and cost efficient alternative when searching for novel drug candidates. Drug discovery pipelines can benefit immensely from molecular machine learning (ML) components. In particular, deep learning (DL) has increasingly been used to approach a multitude of tasks such as property prediction,[1–4] docking,[5,6] and generative modeling.[7–9]

The success of DL in natural language processing[10] and computer vision[11] demonstrates the importance of jointly scaling model capacity and training data quantity. For images, DALL-E[11,12] was trained on $\sim 1 \times 10^8$ text-image pairs, or GPT-3[10] using $\sim 1 \times 10^{11}$ tokens. Analogously, the potential of DL has been leveraged to some extent on molecular tasks using the wealth of available compound data alone.[13,14] Recent work using transformers[15] or graph neural networks (GNNs)[16] showed great promises when pre-training these models on several millions of unlabeled compounds represented in the SMILES language[17] or as 2D molecular graphs.

However, for molecular ML tasks that study the binding of ligands to potential protein targets, both the 3D protein structure and information regarding potential ligand binding poses play a vital role. Models have to capture large chemical spaces and identify complex protein-ligand interaction networks. The availability of such data – protein-ligand (PL) complexes with annotated affinity data – is much lower when compared to compound data – compounds with measured affinity data. PDBBind,[18] one of the most used PL-complex datasets, for example, contains merely 23 496 co-crystallized structures in its current 2020 release. This lack of large scale datasets constitutes a major challenge for approaching PL-binding tasks with deep learning.

Experimental data for training structure-based DL models Experimental data to study ligand binding – and thus protein inhibition or activation – comes mainly in two

flavors: Assay data, primarily binding assays, and protein-ligand complex data, mostly x-ray structures. In the assay it is measured to what extent a small molecule can down- or up-regulate a protein's activity. Such data is comparatively easy to produce, and consequently such datasets are much larger. The public ChEMBL dataset,[14] for example, contains more than 20 000 000 *activities*, i.e., compounds-assay measurement pairs (ChEMBL 33). While the protein data bank (PDB)[19] contains over 200 000 experimental structures, only a subset of those are ligand-bound structures and only a subset of those are annotated with affinity data (i.e. the PDBBind subset v.2020 with roughly 23 000 entries). Other data sources are BindingDB[20] and Binding MOAD.[21] Binding DB contains more than a million binding measurements, but far fewer protein-ligand crystal structures than the others, i.e., 2823 co-crystallized structures as of March 2021.[22] Binding MOAD contains 41 409 complexes. Note that as of 2024, the structural data will be integrated in RCSB PDB.[21] A more detailed overview of available datasets beyond structural data and related common methods in the context of DL-based virtual screening is given by Kimber et al.[22]

Generally, this emphasizes that more protein-ligand complex data with annotated affinities is needed. The MISATO dataset describes a recent effort to provide larger - computationally generated - data by adding data from MD simulations and QM calculations to these empirical data points.[23] As of December 2023, this includes 19 443 QM and 16 972 MD simulations. Other studies investigate into augmenting the structure space using AlphaFold or homology modelling, with two recent examples from the kinase field.[24,25] The KinCo dataset[24] focuses on kinase inhibitors and provides docked poses for 137 778 kinase-ligand complexes, annotated with bio-activity measurements. The complexes are generated using blind docking into kinase homology models covering the complete kinome tree. Gorantla *et al.*[25] recently published a study to investigate the impact of different encodings of the protein and ligand on predicting binding affinities using the Davis[26] and the KIBA[27] kinase datasets. For this study, they generated protein structures, more precisely protein contact maps, using the tools AlphaFold2, Pconsc4, and ESM-1b. Note that no ligand-bound structures were

generated.

Different data splitting strategies to test model generalizability Further care has to be taken, when these datasets are used in machine learning. Oftentimes, datasets may be heavily biased by the lead optimization process. Once a promising compound is found, slight variants are synthesized and tested to optimize characteristics leading to a highly clustered dataset. Splitting such data *randomly* almost necessarily leads to data leakage, i.e., highly similar ligands appear in both, the training and the test set. It is well-known that such uniformly random data splits tend to overestimate model generalizability.[28] In contrast, a generalizable model should also be able to reason about examples which are missing highly similar counterparts in the training data, i.e., they should be insensitive to *domain-shifts* to some extent. Thus, evaluating a model across domain-shifts is crucial to understand its true predictive power better. Therefore, it is advisable to keep *similar* groupings of samples between train, validation and test sets separate. There is a variety of ways to define such groupings for both ligands and receptors. An example in case of small molecules is to consider molecular *scaffolds* and classify molecules along those scaffolds,[1,2,28] which often is a good approximation of a temporal split.[29] The receptor, i.e. the proteins, can be split similarly using their amino acid sequence and clustering them by sequence similarity.[30]

Kinases as model systems Kinases are a prominent protein class, their binding pocket is well known and a fairly large amount of structural data is available. Kinases are involved in many diseases, especially cancer, and have thus been well studied over the last years.[31] Their three-dimensional (3D) protein structure has been resolved in over 6372 x-ray structures[32][1] and the binding mechanism is well understood. Kinases are involved in signal transduction, binding ATP in the catalytic pocket which is located between the N- and C-lobe of the kinase. To date there are over 82 FDA-approved kinase inhibitors known,[33] and most inhibitors bind competitive to the ATP, i.e., they block the catalytic pocket. KLIFS[32] provides an excellent

[1]This is the number of PDB structures in the KLIFS data base as of Sep. 2023.

data source for kinase related structural information: A multiple sequence alignment of the available protein kinases, a binding pocket definition constituted of 85 residues, as well as many additional information about the bound ligand, interactions and affinities. By splitting the PDB structures into monomers, KLIFS holds over 13 500 data points for kinase structures. However, this 3D structural complex data is not as abundant as necessary for the training of complex, modern machine learning models.

Template docking to enrich 3D training datasets In this work, we are particularly interested in evaluating the importance of 3D protein-ligand information for machine learning. We conjecture that even computed protein-ligand complex information is particularly useful because the binding mode is an explicit part of the data. This way an ML model should be able to learn more, than it would just from the ligand and/or the protein sequence alone. Such models, however, are less common mainly due to the data imbalance between assay data and the amount of available co-crystallized protein-ligand complexes.

A potential remedy explored in this work is the use of guided docking to generate additional *in silico* PL-complex data. In doing so, we restrict the application to the protein kinase family since the binding pocket is well-conserved across kinases. Our data generation approach combines structural kinase data from the KLIFS[32] database and a curated set of ligand-kinase activity measurements derived from ChEMBL.[14] The *kinodata* project, assembled from the latter a curated list of ligand-kinase activity pairs, containing over 200 000 entries. The guided docking approach uses a ligand similarity search and template docking[34] to generate the *kinodata-3D* set, a dataset of roughly 120 000 *in silico* kinase-ligand complexes, each annotated with an activity measurement. Schaller et al.[35] recently demonstrated the power of such a docking protocol in a kinase re-docking study. Here, we also use their generated re-docking data to train an additional model to predict the root-mean-square deviation (RMSD) of docking poses. This docking quality model is used to filter our dataset using different quality thresholds.

E(3)-invariant GNN model using kinodata-3D Using these generated complexes, we conducted an extensive case study that involves training both a structure-based E(3)-invariant GNN model and multiple structure-free GNN baselines for binding affinity prediction. Given the docked poses, we predict the affinity, in terms of pIC50 values, known from the underlying ChEMBL data.

The binding of a ligand to a protein does not depend on their absolute position or rotation in space. This should be reflected when building machine learning models, more precisely neural networks that incorporate 3D information. Formally, we say that our network should be $E(3)$-invariant where $E(3)$ is the Euclidean group.[36] Thus, a model reasoning about a bound complex should be invariant to rotations and translations. A natural approach is to drop information on absolute positions and instead focus on relative features. This can be achieved by restricting the network to only use inter-atomic distances as inputs rather than the full 3D-structure. For the structure-based model, we implemented an $E(3)$-invariant graph neural network (EGNN) that processes the entire docked complex as a point cloud.

We compare its predictive performance to two baseline models: A model that is similar to GraphDTA[37] using a graph isomorphism network (GIN)[38,39] for representing ligands and a sequence-based representation of the binding site, and a GNN with a similar architecture as our the EGNN barring access to 3D structure. All models are evaluated across both a pocket sequence-based cold-target and a scaffold-based cold-ligand split, as well as a random split.

Our results show that our proposed structure-based models make substantially more accurate predictions on the held-out test data, even in the presence of domain shifts in the ligand space as simulated through scaffold splits. Our data generation pipeline and our concrete application thereof are thus important steps towards unlocking the full potential of structure-based deep learning for the sake of accelerated drug discovery.

2 Data and methods

Large datasets are crucial to machine learning in general and neural networks in particular. However there is a limited availability of experimentally resolved co-crystallized structures, containing both the target protein and the bound ligand. In contrast, assay data is much more abundant since it can be experimentally generated faster and in higher throughput. To still leverage this available information for structural models, we explore the use of guided docking for the generation of ML-scale datasets. We will first introduce the template docking pipeline used to generate kinodata-3D, then we will introduce the affinity prediction models, with a focus on the structure-informed E(3)-GNN model. Finally, we propose the data splitting strategies and evaluation statistics.

2.1 Kinodata-3D: an *in silico* kinase complex dataset

In this study, we focus on the target class of kinases, where the structural mechanisms are fairly well-understood. In particular, the ATP-pocket is typically targeted and structural information, i.e., kinase-ligand complexes, is available in larger amount.

Due to the fact that docking algorithms have difficulties with regard to finding the optimal position or getting the rank correct, we use structural kinase information from the available X-ray complexes to guide the docking process in a process called template docking.[34]

In a nutshell, our pipeline to generate *in-silico* kinase-ligand complexes consists of the following steps:

1. Take a compound L with known activity on a kinase K from kinodata[2].
2. Find the complex for a similar ligand L' in the PDB database.[19]
3. Use the KLIFS entry of the PDB complex with ligand L' as a template for the ligand L in the docking run.

[2] https://github.com/openkinome/kinodata

Figure 1 provides a schematic overview of these steps, described in more detail below. In Subsection 3.1, we elaborate further on details such as the reasons, why certain steps in the pipeline did not succeed for individual data points.

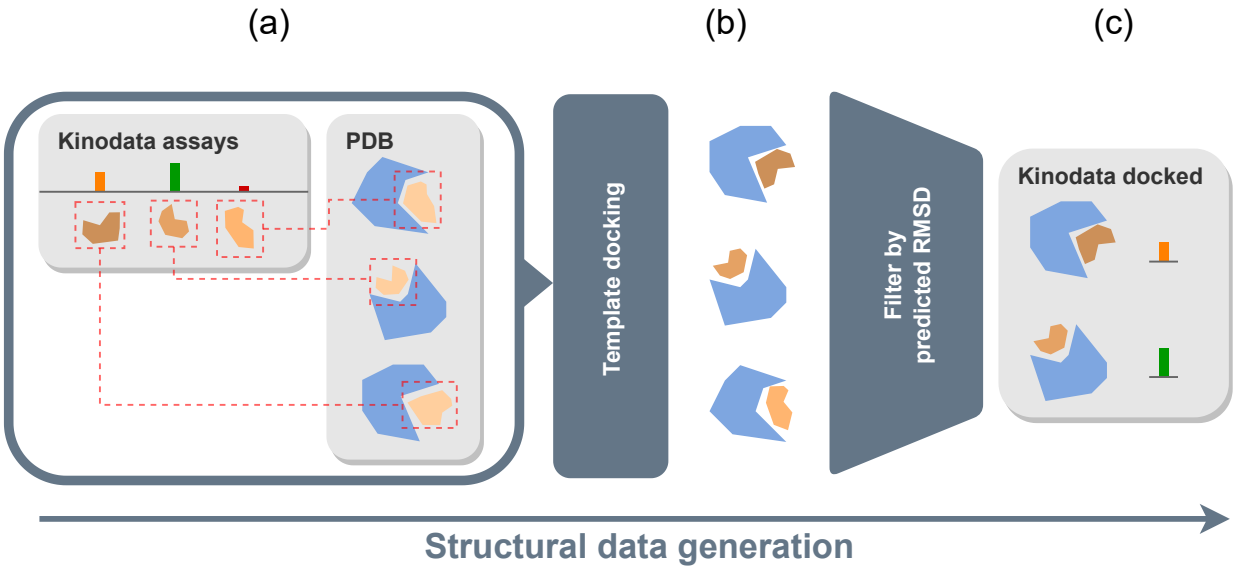

Figure 1: The pipeline for structural data generation. (a) The ligands from kinodata are used to identify PDB structures with similar co-crystallized ligands. (b) These are used as a template by the Posit docking algorithm. (c) The resulting poses are filtered by an RMSD prediction model.

Assay data – kinodata The basis for our docking dataset is publicly available assay data from Chembl 31,[14] from which we source kinase activity data via kinodata[3]. The kinodata pipeline in short: A list of 473 protein kinases is assembled by cross-checking other widely recognized sources on kinases.[30,32,40] Next, bioactivity measurements for these kinases were obtained from ChEMBL via Uniprot IDs. The resulting data was filtered following the steps suggested by Kramer and colleagues.[41,42] The resulting set consists of experimental activity values for 202 592 kinase-ligand assay pairs, including K_i, K_d, and IC_{50} values.

Finding a template (Figure 1.a) All kinase-ligand pairs are further processed by searching for an appropriate co-crystallized template complex. For a given kinase-ligand assay

[3] https://github.com/openkinome/kinodata

pair, the ligands of all PDB structures for the same kinase (matching via Uniprot-ID) are searched. The PL-complex with the most similar ligand using a Tanimoto similarity search over Morgan fingerprints with radius 2 and length 2048[43] is chosen as a template. For these template structures, we query the KLIFS database[32] for the structure of the adenosine triphosphate (ATP)-binding pocket, defined by 85 residues. Utilizing the pre-processed structures found in KLIFS, as opposed to the RCSB structures, allows us to leverage the additional kinase-specific information readily available in KLIFS more easily.

Template docking (Figure 1.b) For the docking itself, we use *OpenEye's OEDocking* tool[44] as exemplified in the *kinoml*[4] framework. This docking pipeline employs the *Posit* algorithm for template docking[34] using default settings. It aims to achieve a close alignment of the generated ligand pose in the template kinase structure with the identified most similar known bound ligand. Additionally, the torsion angle strain on the ligand molecule is added to the optimization objective. In its current form, only the top-scoring docking pose is returned for each protein-ligand pair. We ran the algorithm for each kinase-ligand pair using a timeout of 10 minutes and a memory limit of 10GB per docking run. Overall, 34 500 runs exceeded the memory limit while 17 300 reached the timeout. The docking terminated successfully in 128 000 instances.

Pose quality assessment – RMSD model (Figure 1.c) For docking runs that terminate successfully, we filter the results using ligand (fingerprint) similarity, docking score (ChemGauss 4), and Posit probability. Posit probability gauges the likelihood of having found a correct binding pose.

A recent benchmark study[35] assesses the RMSD of 39 204 cross-docked ligand poses in kinases[5]. We trained a simple neural network on this recent kinase cross-docking benchmark data to predict the binding pose RMSD – as a surrogate for docking quality – and use it to

[4] https://github.com/openkinome/kinoml

[5] Results are available at https://github.com/openkinome/kinase-docking-benchmark.

categorize the results with thresholds at 2Å, 4Å and 6Å. The lower the value, the better the docking pose is estimated to be. The lower the value, the better the docking pose is estimated to be. The model takes as input (1) the Chemgauss 4 score, (2) the Posit probability,[34] and (3) the ligand similarity (Morgan fingerprint). The network consists of two hidden layers of widths 64 and 32 with ReLU activation functions. For training we used the Adam optimizer with learning rate 0.001 on batches of size 64 for 200 epochs. The training objective was the mean squared error loss function. We performed this training once on a random train-test split with a test set size ratio of 0.2 and finally for the full dataset. The model is used to filter the dataset according to these predictions.

2.2 Affinity prediction models

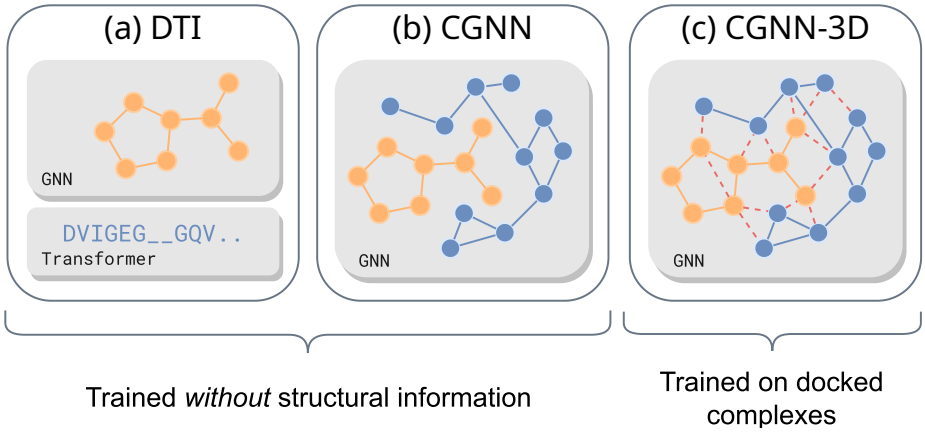

Figure 2: Affinity prediction models. We consider two non-structural baselines that make predictions based on (a) pairs of ligand molecular graphs and pocket-residue sequence pairs, and (b) the complex graphs where structural information has been removed (Subsection 2.4). (c) The main GNN utilizes the structural information generated by template-docking in the form of *complex graphs* (Subsection 2.3).

To be able to explore the performance gain when using 3D information available from an *in silico* dataset of docked complexes, we implemented four different GNN models that include or exclude complex and structure information, as well as a ligand-based model only as baseline.

2.3 E(3)-invariant message passing model using the protein-ligand complex information

Our main objective was to implement a message passing graph neural network model that uses the kinase-ligand complex as input, named complex graph here, which is invariant to rotations and transformations of the complexes.

Complex graphs We model kinase-ligand complexes as *heterogeneous, geometric graphs* (point clouds) consisting of n nodes in 3D space, $\boldsymbol{X} \in \mathbb{R}^{n\times 3}$. Since the binding pocket is known, we only use this part of the protein. Each node $i = 1, \ldots, n$ represents an atom with atomic number (chemical element) $a_i \in \mathbb{N}^+$, formal charge $q_i \in \{-2, -1, 0, 1, 2\}$, and is either part of the ligand or pocket. Note, that we do not explicitly model hydrogen atoms for performance reasons, but instead include an additional node-level feature $h_i \in \mathbb{N}^+$ that models the number of hydrogens connected to atom i. Pairs of atom nodes (i, j) within the ligand or pocket may be connected via covalent bonds whose types are encoded as $b_{ij} \in \mathbb{N}$. We only consider single-, double- and triple-bonds, encoded as $b_{ij} \in \{1, 2, 3\}$, respectively. Aromatic bonds are removed through kekulization. All potentially occurring other types of covalent bonds are pooled into one additional class $b_{ij} = 4$. For atom pairs, which are not connected by a covalent bond, we set $b_{ij} = 0$. The distance between two atoms is given by $d_{ij} = \|\boldsymbol{X}_i - \boldsymbol{X}_j\|$. The graph topology is governed by covalent bonds as well as the structural information, i.e. the pair-wise atom distances. More precisely

$$\mathcal{N}(i) = \{j \mid d_{ij} \leq d_{cut} \vee b_{ij} > 0\} \tag{1}$$

are the immediate neighbors of node i. Following related work,[4,45] the cut-off distance is set to $d_{cut} = 5$Å. For computational performance, we limit $\mathcal{N}(i)$ to only include at most the $k = 16$ nearest neighbors of i. The total number of edges is given by $m = \sum_i |\mathcal{N}(i)|\,/2$.

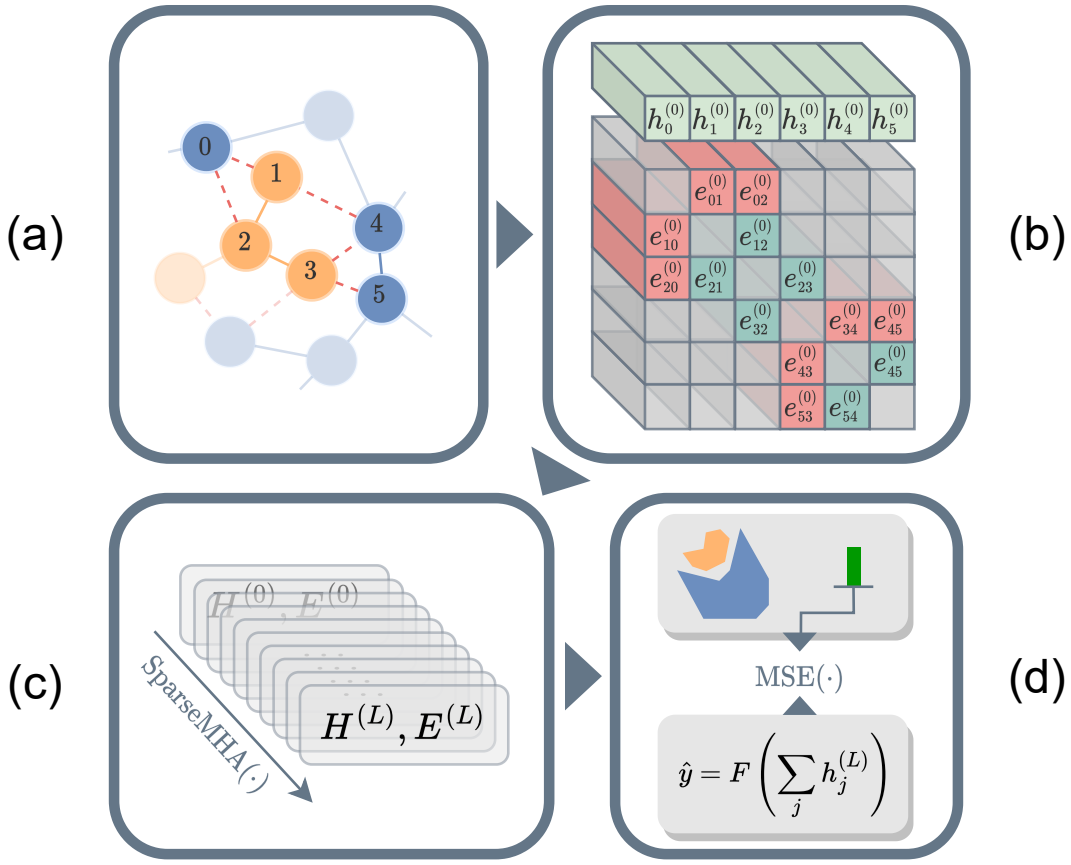

Figure 3: Schematic overview of our E(3)-invariant GNN for structural binding affinity prediction as a 4-step process. (a) Modeling the data (Section 2.3, Complex graphs), (b) deriving an initial representation (Section 2.3, Featurization), (c) (Section 2.3, $E(3)$-invariant message passing), (d) making predictions (Section 2.3, Readout) and training the model (Section 6).

Featurization Initial node-level features $\boldsymbol{H}^{(0)} \in \mathbb{R}^{n \times d}$ are derived from atomic numbers, formal charge and the number of hydrogen atoms, all of which are treated as a categorical feature and thus are encoded via one-hot embeddings.

Initial edge-level features $\boldsymbol{E}^{(0)} \in \mathbb{R}^{m \times d}$ consist of categorical bond-types and features derived from inter-atomic distances. In particular, the distances are embedded via a set of learnable Gaussian kernels. For further details regarding featurization, see appendix 5.1.

$E(3)$-invariant message passing Given a complex graph and initial features as described above, we employ a form of sparse multi-headed attention (SparseMHA) as our message

passing scheme

$$\boldsymbol{H}^{(l+1)} = \text{LayerNorm}\left(\boldsymbol{H}^{(l)} + \text{MLP}_H^{(l)}(\tilde{\boldsymbol{H}}^{(l)})\right) \tag{2}$$

$$\boldsymbol{E}^{(l+1)} = \text{LayerNorm}\left(\boldsymbol{E}^{(l)} + \text{MLP}_E^{(l)}(\boldsymbol{\alpha}^{(l)})\right) \tag{3}$$

$$\tilde{\boldsymbol{H}}^{(l)}, \boldsymbol{\alpha}^{(l)} = \text{SparseMHA}^{(l)}\left(\boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l)}, \boldsymbol{E}^{(l)}\right). \tag{4}$$

$$\tag{5}$$

This operation is repeated across multiple layers $l = 0, \ldots, L-1$. $\text{MLP}_H^{(l)}$ and $\text{MLP}_E^{(l)}$ are shallow multi-layer perceptrons (MLPs) that operate on the last dimension of their respective input tensors, i.e., node- and edge-wise. They both use one hidden layer, sigmoid linear unit (SiLU)[46] as an activation function and have output size d. LayerNorm$(\cdot)$ denotes layer-normalization.[47]

Put simply, SparseMHA$(\cdot)$ implements standard multi-headed scaled dot-product attention,[48] but attention between nodes is constrained by the complex graph neighborhoods (Equation 1). The model only computes and uses attention weights between nodes that are within cutoff distance of one another or connected by a covalent bond. Exploiting sparsity in this way is beneficial mainly due to its lower time- and memory-complexity when implemented with sparse tensors. For more details see appendix 5.2.

Readout We obtain the binding affinity predictions from the final node representations via a global sum readout and an MLP with scalar output as

$$\hat{y} = \text{MLP}\left(\text{LayerNorm}\left(\sum_{i=1}^{n} \boldsymbol{H}_i^{(L)}\right)\right). \tag{6}$$

The MLP has 2 hidden layers and also uses the SiLU activation function. The final prediction $\hat{y}$ is $E(3)$-invariant because $\boldsymbol{H}_i^{(L)}$ depends only on atomic distances, which themselves are $E(3)$-invariant, but on no other structural information.

We will refer to this model, operating on complex graphs and using 3D structural infor-

mation, with the acronym CGNN-3D, see Figure 3.

2.4 Structure-free baseline models

To evaluate the effect of using docking-based structural information we implement two structure-free baselines. An ablative model derived from CGNN-3D and a drug target interaction (DTI) model.

CGNN-3D ablative baseline For the ablative baseline, we aim to construct a model that reuses the neural network architecture of CGNN-3D. However, this model should not have access to structural and docking information such as atom positions or similar ligand information. As such, the CGNN baseline closely resembles the CGNN-3D model except for two changes:

1. Neighborhoods for message passing are determined by covalent bonds only, i.e., $\mathcal{N}(i) = \{j \mid b_{ij} > 0\}$.

2. The edge featurization resulting in $\boldsymbol{E}^{(0)}$ does *not* make use of atomic distances.

Note that this implies $j \notin \mathcal{N}(i)$ if j is a pocket-atom and i is a ligand-atom and vice-versa. Despite this, the model is still able to capture relationships between the ligand- and pocket-atoms, since the readout module (6) includes *all* complex-atoms. The hyperparameter configuration is similar to the CGNN-3D model, as well. This way, the model corresponds closely to CGNN-3D, only differing in its lack of access to 3D structural information.

DTI model using ligand and protein information independently The ligand's molecular graph is encoded using a GIN.[38] Atoms and bonds of the molecular graph are featurized as described in Section 2.3. Final predictions are obtained using the exact same readout module defined in Equation 6. In parallel to the ligand's graph, the pocket residue sequence is processed. Residue types are one-hot encoded and the corresponding sequence is processed using a shallow transformer-like model.[48] The final fixed-size residue sequence and

ligand representation are concatenated and processed using the readout module defined in Equation 6. A more detailed, technical description of the DTI baseline is given in appendix 5.

2.5 Data split and cross-validation

When estimating the generalization performance of a data-driven model, the way data is split into training and test examples plays a crucial role. To better understand the performance of our models, we implemented three splitting schemes: Random, ligand-scaffold, and cold-target split.

Cross-validation We do five-fold cross-validation for each model and splitting type. Thus, we evaluate each model on five distinct random data splits, five distinct scaffold-grouped and five distinct pocket-grouped data splits.

For the random split, we split the data randomly in five-folds for cross-validation, using the `KFold` iterator in `scikit-learn`.[49] In the other two cases, we first find an informed partitioning of the data into groups and then carry out five-fold cross-validation (`GroupKFold` iterator). In all three cases, it is guaranteed that either each sample (random) or each group is contained in *exactly* one fold.

Ligand-scaffold partitioning For the ligand-side, we base splits on *molecular scaffolds* which are often used for this purpose.[1,2,28,29] Intuitively, we group together molecules that are similar, i.e., that share specific core motifs or scaffolds. More precisely, we group all docked kinase-ligand complexes by the generic Murcko scaffold[50] of their ligands. Figure 4 provides an example of a partitioning arising from the molecular scaffold.

Cold-target partitioning For the kinases, we first group the docked protein-ligand complexes based on their *pocket sequences*, defined by the 85 KLIFS pocket residues.[32] The same kinase can be found in complex with different ligands, thus, the different instances, i.e., pdb

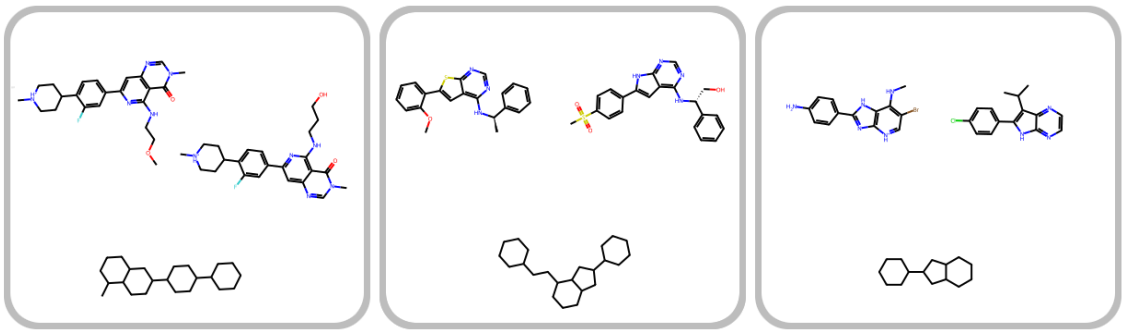

Figure 4: A partitioning of a set of molecules using their Murcko scaffolds. Top line, examples of two molecules of a group, bottom line, generic Murcko scaffold of these molecules.

structures, of the same kinase must be grouped together. To make this partitioning more robust, i.e., to account for evolutionary similarity between kinases, we further cluster the unique sequences by similarity. For this, we calculate the sequence similarity using the block substitution matrix (BLOSUM) similarity matrix,[51] derived from comparisons of closely related protein sequences. The corresponding sequence similarity matrix is used to compute clusters based on affinity propagation.[52] It selects a subset of exemplary data points that effectively capture the characteristics of the entire dataset and then clusters the remaining data points based on their closest resemblance to these exemplar points.

3 Results and discussion

3.1 Generated dataset analysis

Losses in the template docking pipeline While the docking pipeline works well for a large number of kinase-ligand combinations, some steps cause problems for individual cases. The throughput of all states is schematically illustrated in Figure 5. Our pipeline searches for the most similar ligand co-crystallized against the same target to be used as template during docking. The lookup of available co-crystallized PDB structures is the first possible point of "failure". On one side, crystal structures are only available for roughly half of the

kinome, on the other side, even if several structures are available for a kinase, they may not be bound to a ligand. More precisely, when searching for the kinase Uniprot IDs in the PDB databank, no structures are returned for 1450 queries (*no similar PDB*). Next, for a found most similar co-crystallized kinase structure, we search the entires in KLIFS since additional meta data is used from there. The matching of PDB structures to KLIFS failed in 22 264 cases (*no KLIFS entry*). Causes for aborting the process at this stage are that the kinase may not be contained in KLIFS due to (i) different definitions of the term kinase itself (e.g. JAK2 regulatory pseudokinase domain (PDB: 8EX1) or Phosphatidylinositol 5-Phosphate 4-Kinase (PDB: 7QPN)) or (ii) multiple ligands bound to a kinase, where our crude approach always picks the most similar ligand, which may not be the one with the kinase activity (Mitogen-activated protein kinase kinase 1 (PDB: 3ORN), selected ligand: 3OR, KLIFS ligand: ANP). The last step, i.e., the Posit template docking, reached the timeout 10 minutes in 17 280 cases and the memory limit of 10GB in 34 467 cases. This leaves us with 127 628 docking poses.

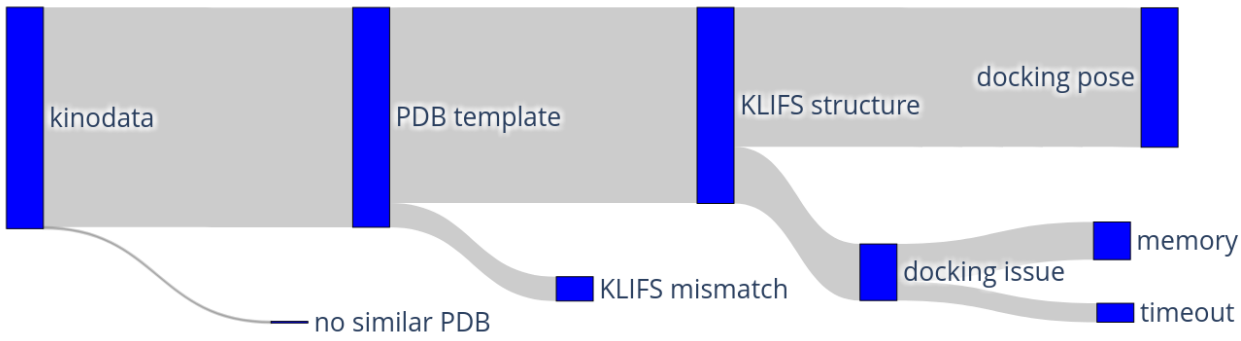

Figure 5: Figure shows the losses of structures during pose generation in the docking pipeline. From initially 201 142 pairs, we could generate a set of 118 586 complexes composing kinodata-3D.

Dataset statistics After deduplication with respect to kinase-ligand tuples, the kinodata-3D dataset encompassing *all* docked complexes contains 118 577 labeled kinase-ligand pairs. These labels consist of 102 571 pIC50 values, 9069 pKi values, and 6946 pKd values. Figure 6 shows how the activity measurements are distributed across activity type. The pIC50 values

are normal distributed in the range of [2,12] — the higher the more active — with a mean at pIC50 of 7.3, which translates to a high affinity binder with an IC50 of 50nM. Note that for the remainder of the study, we only used the subset of pairs with available pIC50 values, due to its quantity. The kinase-ligand pairs consist of ≈ 254 distinct kinases and ≈ 80 000 distinct ligand compounds. Figure 7 illustrates which parts of the kinome tree are covered.

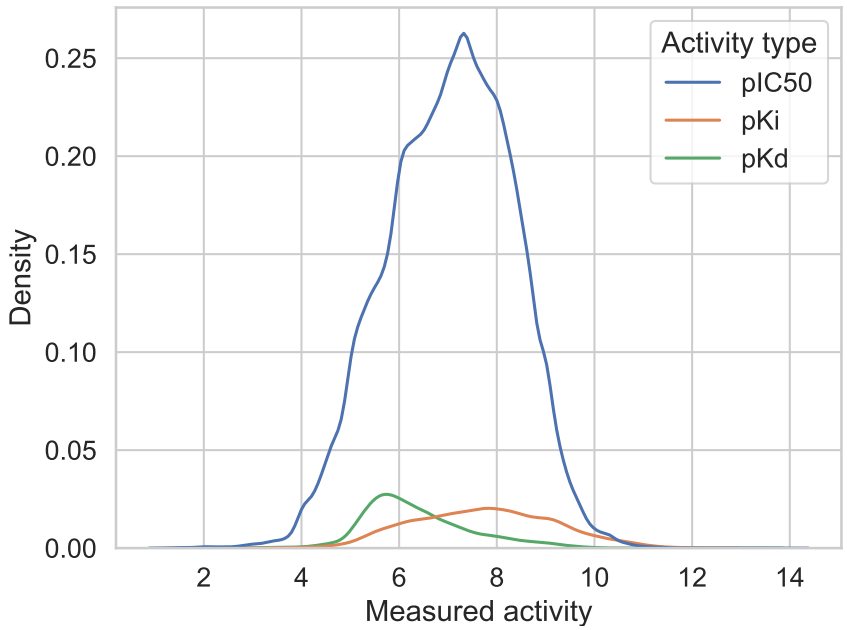

Figure 6: Distribution of kinodata-3D conditioned on activity.

Clearly, the kinome tree is not covered uniformly, due to differences in therapeutic interest, ease of crystallization or other facts. The focus on certain kinases let to large amounts of data for single kinases, such as EGFR and PKA (see Figure 7). Note that the issue of uncharted kinase x-ray structure territory could be overcome by using structural modelling techniques.[24,25] In KinCo, they generated 137 778 kinase-ligand complexes using blind docking into kinase homology models. While this way, the can cover the complete kinome tree, the blind docking process might suffer in accuracy. We instead rely on template docking and only considers kinases with known x-ray structrue from KLIFS.[32] Template docking cannot achieve the same coverage as blind docking but is particularly well-suited to find accurate

poses in cases where template data is available.[35]

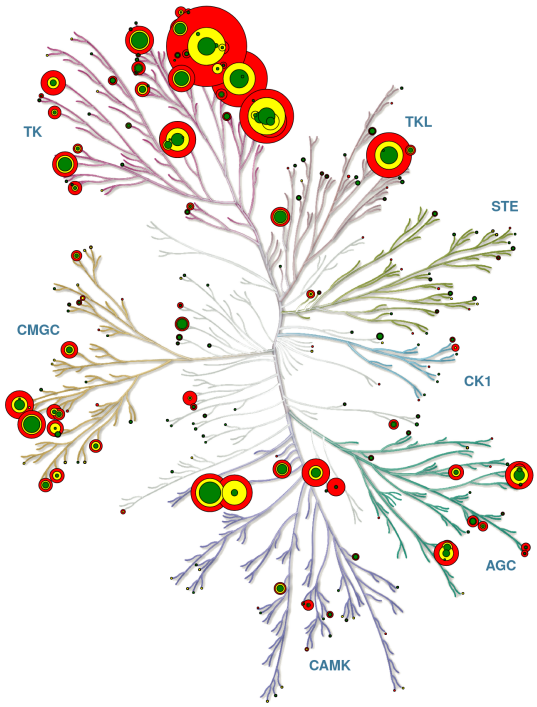

Figure 7: The kinase tree and the kinase coverage. The size of the circles indicates how many activities of the given kinase are in the kinodata-3D dataset for predicted RMSD values of 6Å (red), 4Å (yellow), and 2Å (green). Illustration reproduced courtesy of Cell Signaling Technology, Inc. (www.cellsignal.com).

Docking quality estimation We trained a simple neural network to predict the docking RMSD on the re-docking kinase dataset containing 39 204 samples by Schaller *et al.*[35] using docking score, posit probability and fingerprint similarity as features, as described in Subsection 2.1. We apply this additional model instead of using the docking score, the posit probability, or the ligand similarity only because the correlations between those values and the RMSD was lower. The Chemgauss docking score, for example, only has a Pearson correlation[6] of approx. 0.10. While the posit probability and the fingerprint similarity have better correlations, -0.56 and -0.44, respectively, combining all three features in an NN model gives a much more reliable performance. The Pearson correlation between predicted

[6]This excludes overflow values of the score.

RMSD and true RMSD between the pose and the co-crystallized ligand is 0.70. This can be rationalized by the empirical distributions between the individual features and the RMSD in Figure 8. The individual distributions tend to be more complex and bimodal. With one of the modes corresponding to higher quality docking results (i.e. low RMSD values). A joint model can capture the interplay of these features better and thus provide us with more reliable predictions.

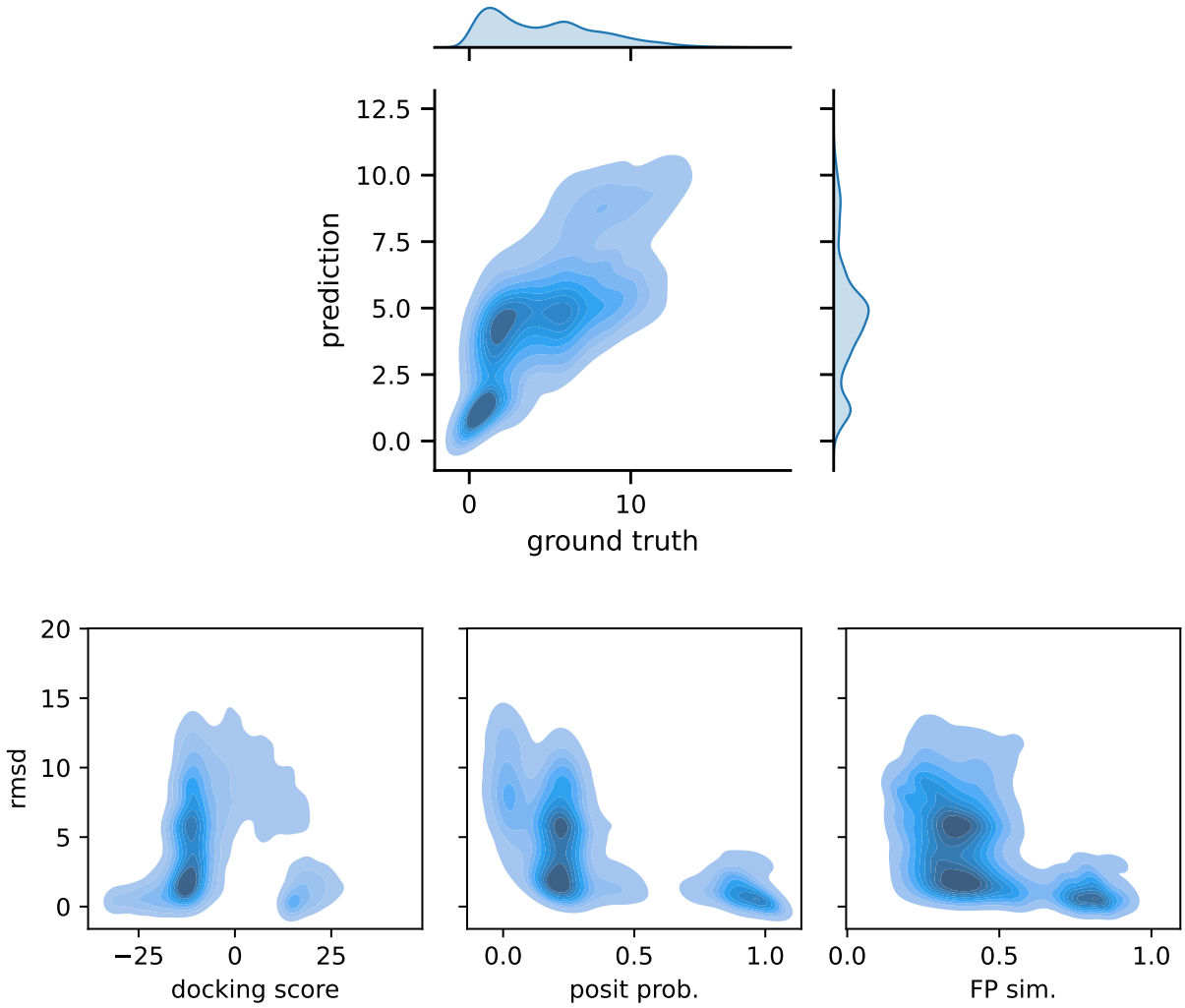

Figure 8: The predicted docking RMSD versus the ground truth in Ångstrom on a random test set and the individual features' values (Chemgauss4 score, posit probability, fingerprint similarity) versus the ground truth RMSD. The model was trained and tested on re-docking data by Schaller et al.[35] The Pearson correlation between predicted and true RMSD is approx. 0.70. Test set performance of the first model with an 80/20 split is illustrated.

Since we are interested in assessing the impact of data quality on the performance of structural affinity prediction, we also filter our dataset based on the predicted docking RMSD of complexes. To this end, we apply the docking RMSD predictor (described in Subsection 2.1) to all complexes in our dataset.

Figure 9 (a) shows that the data distribution subject to predicted docking RMSD is bimodal. The RMSD model predicts an RMSD $\leq$ 2Å for roughly 25% of the data. A successful docking often is characterized by an RMSD below 2Å.[34,53,54] At 4Å, the docked ligand still tends to cover similar area as the actual ligand pose. We also include the threshold of 6Å to assess model performance including poses which likely are of poorer quality. Note that for our DL models, before applying the splitting scheme (see Subsection 2.5), we create three separate subsets of our data based on an RMSD cutoff of 2Å, 4Å and 6Å. The resulting subsets consist of 27 595, 50 097, and 95 959 entries, respectively.

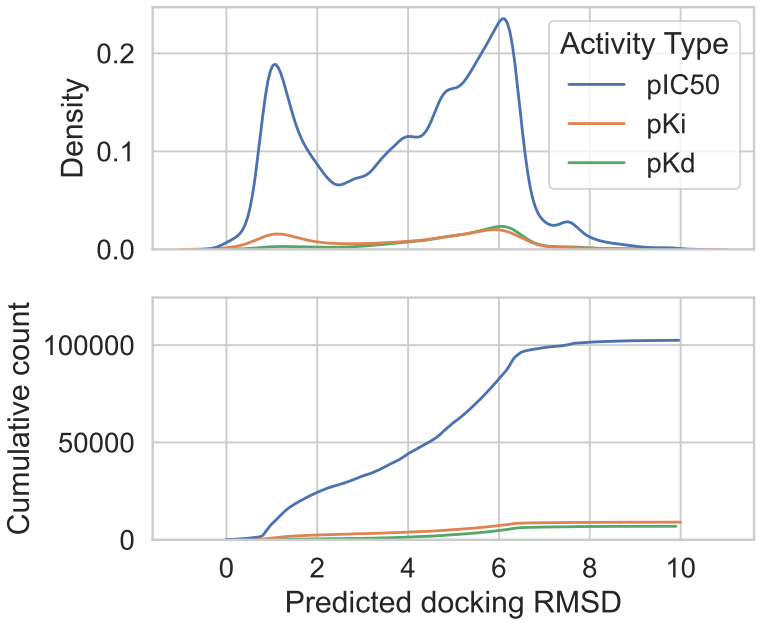

Figure 9: Distribution of kinodata-3D conditioned on predicted docking RMSD.

These only include complexes with a predicted docking RMSD less than or equal to their associated cutoffs and thus partition our data into (estimated) high-, medium- and low-quality data subsets. The predicted RMSD is a typical estimate of (re-)docking quality:

If the RMSD of a docked pose compared to the empirical pose is low, it is most certainly of higher quality. Conversely, if the RMSD is high, the docked pose deviates more from the empirical pose and is not as informative wrt. the actual binding modes.

3.2 Binding affinity prediction

Each model is trained and evaluated on five folds (see Subsection 2.5) for each of the three data subsets – grouped by predicted RMSD – as described above. Data splitting takes place after the RMSD subset operation. In addition, we restrict the affinity prediction task to the prediction of pIC50 values since they are the most abundant. The models are trained to minimize the mean squared error between their prediction and the ground truth activity using the AdamW[55] optimizer. In addition to reporting the performance of all models on the test splits visually, using boxplots, we conduct statistical tests to assess if the model performances differ significantly. For further details regarding the training protocol and statistical evaluation see appendix 6 and 7, respectively.

Type of train-test split highly impacts reported model performance Figure 10 summarizes the test performance of all models in terms of mean absolute error (MAE) and Pearson correlation. We choose these two performance metrics because Pearson correlation, as a unit-free measure, is well-suited for judging a model's generalizability, whereas the relative difference between two models is more easily detectable in terms of their MAE.

While a Pearson correlation of $\rho \approx 0.72$ (Figure 10) on *randomly* held-out test data might suggest that the respective model exhibits strong generalization, the metrics observed on the *scaffold* split should be considered as more realistic estimates thereof. This overly optimistic behavior of random splits has previously been shown by several other studies.[56–59] All models still generalize fairly well on the scaffold split ($\rho \approx 0.60$). On the contrary, the precision degrades significantly when tested on a sequence-based pocket split ($\rho \leq 0.5$ in almost all cases).

Predicting binding affinities for novel ligands for a known query kinase seems to be a less challenging task compared to making predictions for complexes that involve an unknown kinase. This suggests that, even in the presence of *in silico* structural information, the trained models rather capture kinase-specific binding mechanisms.

The CGNN-3D model outperforms baseline models in most set-ups In the *scaffold* and *random* split setting, the structure-based CGNN-3D model achieves the best (lowest) MAE across the board. Due to the more realistic set-up when using scaffold-splits, the following discussion will focus on the scaffold splits only. With an average MAE of 0.72 when including good and medium docking results (RMSD $\leq$ 2Å and 4Å), the CGNN-3D makes, on average, substantially more accurate predictions than both structure-free baseline models. In both these settings, our statistical analysis provides significant p-values (see Section 7). In comparison to the DTI baseline (mean MAE of 0.78), the improvement relative to the ablated CGNN baseline is slightly higher, at around 0.80 (for RMSD $\leq$ 4Å). This suggests that the specific choice of architecture and parameterization also matters. Using docked poses of higher estimated quality (RMSD), the observed improvement, both in terms of correlation and MAE, is much stronger. These observations support the main hypothesis that *in silico* structural information can be useful for training structure-based machine learning methods.

Our results stand out from other works such as recently published by Gorantla *et al.*,[25] in which the authors observe no significant difference between DTI models with and without access to structure-based protein encodings. The key advantage of CGNN-3D over their structure-informed DTI models is access to structural information that directly *relates ligands and kinases* in the form of predicted complexes. Their approach is similar to comparing our DTI model to the ablative CGNN model, where we see no clear difference. Our improved CGNN-3D results suggest that going one step further in generating joint protein-ligand conformations can indeed aid machine learning. The key advantage of CGNN-3D over their

DTI models is access to structural information that directly *relates ligands and kinases*.

Note that in the pocket split setting, where all methods exhibit relatively poor performance, the DTI baseline model tends to perform better, regardless of the RMSD cut-offs.

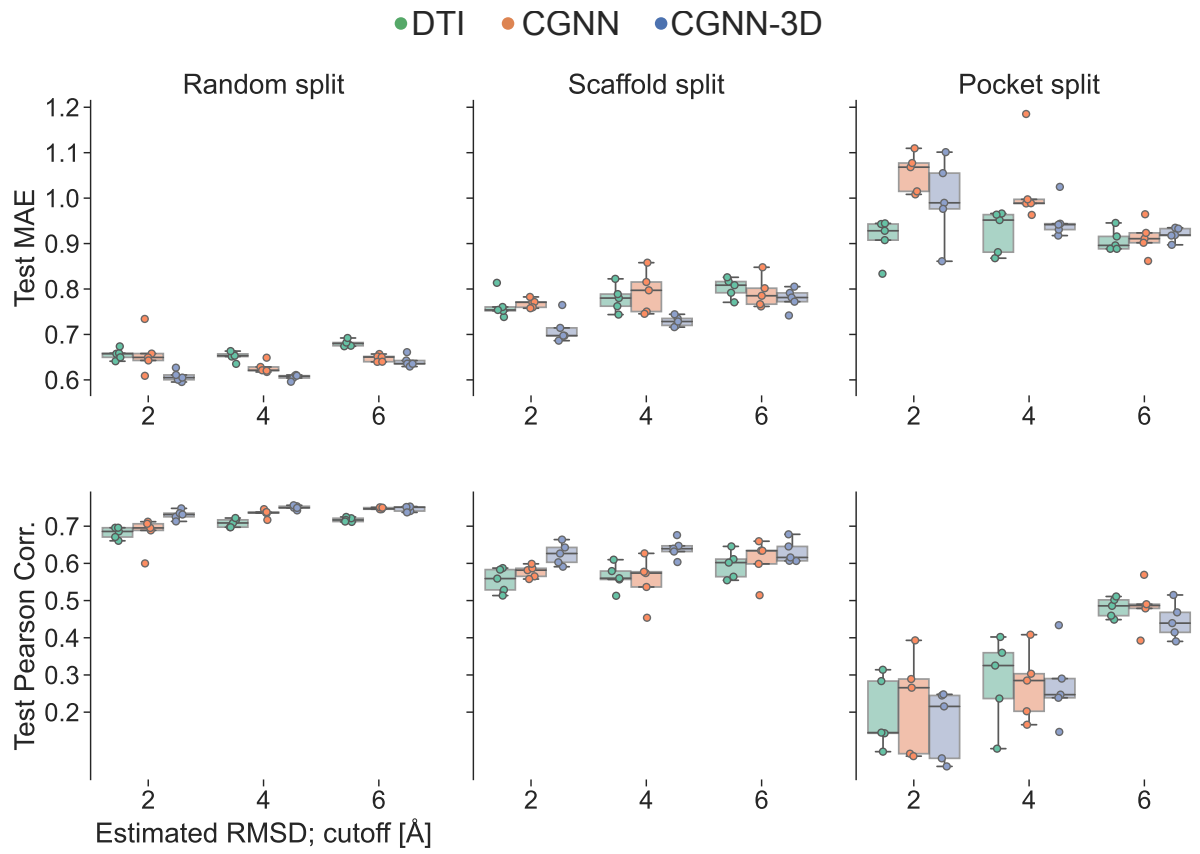

Figure 10: Model performance for different split types and RMSD-based data subsets. Test mean absolute error, Pearson correlation between ground truth and predicted test activity. Each dot represents one tested model.

4 Conclusion and future work

This work is a step towards unlocking the full potential of structure-based deep learning for drug design with focus on kinase-related tasks. Our main contributions are (i) increasing the training data quantity by a computational complex data generation approach, and (ii) a concrete show case on how using this structural data improves model performance compared to structure-free baselines. Based on the structural kinase database KLIFS and publicly

available assay data sourced from ChEMBL, we generated kinodata-3D: a large dataset ($\sim 1 \times 10^5$) of labeled, docked ligand-kinase complexes. In order to evaluate its utility, we conducted a rigorous evaluation, training a structure-based DL model, i.e. an E(3)-invariant GNN, as well as multiple structure-free baselines on the task of binding affinity prediction. We found that access to the *in silico* 3D-structures provided by kinodata-3D is beneficial to the training of structure-based GNNs. This benefit persists in case of ligand-space domain-shifts, whereas results for out-of-domain kinases are less conclusive. Our evaluation also underpins the evident assumption that the benefit scales with the quality of docked poses. Thus, we can expect the utility of our data generation approach to grow together with the ability of docking tools to find accurate ligand poses.

In considering future avenues for our project, several possibilities emerge for enhancing the dataset and extending its applications. One promising direction involves augmenting the dataset by retaining multiple docking poses for each protein-ligand pair. Furthermore, the dataset could be expanded to encompass non-binders and their corresponding decoy poses, presenting an opportunity to significantly enhance machine learning models and mitigate the risk of learning spurious patterns.

Another prospective avenue involves applying the dataset to different machine learning models, with a particular focus on generative models. Leveraging the 3D information contributed by docking, these models could be utilized for ligand generation. Additionally, the evaluated affinity prediction model opens up possibilities for future research, such as applying explainability methods to gain insights into the specific ligand features contributing to high affinity and the underlying binding mechanisms. This way, one could gain insights on which parts of ligands contribute most to high affinity and which binding mechanisms are exploited. Both these directions offer valuable opportunities to broaden the pool of candidate ligands, providing a complementary approach to traditional screening pipelines that primarily rank known compounds.

In order to enable other researchers to leverage our insights and data more easily, we

made the dataset and all code freely available. This way, the kinodata-3D dataset can also be used for training other DL-models, such as an ongoing kinase-specific generative models that use pocket information for ligand generation building on the KinFragLib.[60]

Summarizing, our study underscores the enhancement in predicting binding affinities achieved by incorporation synthetic binding poses. This opens avenues for future studies to scale up the incorporating of structural information in DL models and by that the accuracy.

Data and Software Availability

The kinodata-3D generation pipeline is available on Github [7]. The raw kinodata-3D dataset containing all docked complexes is published through Zenodo [8]. Our code used to preprocess the data into a form amenable to machine learning and for training as well as evaluating the binding affinity models is hosted in a separate Github repository [9]. For the sake of guaranteed reproducibility, we also publish the exact preprocessed version of kinodata-3D and data splits used in the training and evaluation of our models, also on Zenodo [10].

Acknowledgement

We express our gratitude to OpenEye, Cadence Molecular Sciences for granting us the privilege of utilizing their tools under the Free Public Domain Research License. We are thankful for funding from Saarland University for the NextAID project through which the positions of MB and JG are supported. AV acknowledges financial support from a BIH Einstein Visiting Professor Fellowship for John Chodera which layed the foundation for the openkinome work.

[7] https://github.com/volkamerlab/kinodata-3D
[8] https://zenodo.org/records/10410259
[9] https://github.com/volkamerlab/kinodata-3D-affinity-prediction
[10] https://zenodo.org/records/10410594

Supporting Information Available

5 CGNN-3D description

5.1 Node and edge featurization

Node featurization Atomic numbers, formal charge and the number of hydrogens per atom, i.e. node, are each one-hot encoded and then concatenated, resulting in a vector z_i. $\boldsymbol{H}_i^{(0)} = W_h^\top z_i$ is the result of applying a learned linear transformation W_h.

Edge featurization Inter-atomic distances are embedded via a set of Gaussian kernels, i.e.,

$$\begin{aligned} \boldsymbol{d}'_{ij} &= \left(g_1(d_{ij}), \ldots, g_d(d_{ij})\right) W_d \in \mathbb{R}^d \\ \text{where } g_k(x) &= \frac{\exp(-(x-\mu_k)^2)}{\sigma_k}, \end{aligned}$$

and μ, $\sigma \in \mathbb{R}^d$ and $W_d \in \mathbb{R}^{d\times d}$ are trainable parameters. The values in μ are initialized linearly spaced between 0 and d_{cut}. σ is initialized to contain ones. Finally, given one-hot encoded bond types b_{ij},

$$\boldsymbol{E}_{ij}^{(0)} = W_b^\top b_{ij} + \boldsymbol{d}'_{ij}$$

where W_b is also a learned linear transformation.

5.2 Sparse multi-headed attention

SparseMHA has two outputs, n value-aggregations, i.e., the standard output of self-attention, and the corresponding unnormalized attention weights across *all H attention heads*

$$\begin{aligned} \tilde{\boldsymbol{H}}^{(l)}, \boldsymbol{\alpha}^{(l)} &= \text{SparseMHA}^{(l)}\left(\boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l)}, \boldsymbol{H}^{(l)}, \boldsymbol{E}^{(l)}\right) \\ \text{where } \tilde{\boldsymbol{H}}^{(l)}, \boldsymbol{\alpha}^{(l)} &\in \mathbb{R}^{n\times d}, \mathbb{R}^{n\times n\times H}. \end{aligned}$$

These are calculated as

$$\tilde{\boldsymbol{H}}_i^{(l)} = \sum_{h=1}^{H} \sum_{j \in \mathcal{N}(i)} \boldsymbol{\omega}^{(h,l)} \left(\boldsymbol{H}_i^{(l)}, \boldsymbol{H}_j^{(l)}, \boldsymbol{E}_{i,j}^{(l)} \right) \boldsymbol{\phi}^{(h,l)} \left(\boldsymbol{H}_j^{(l)} \right) \tag{7}$$

$$\boldsymbol{\alpha}_{ij}^{(l)} = \left(w_{ij}^{(1,l)}, \ldots, w_{ij}^{(H,l)} \right) \tag{8}$$

where

$$\boldsymbol{\omega}^{(h,l)} \left(\boldsymbol{H}_i^{(l)}, \boldsymbol{H}_j^{(l)}, \boldsymbol{E}_{i,j}^{(l)} \right) = \frac{\exp(w_{ij}^{(h,l)})}{\sum_{k \in \mathcal{N}(i)} \exp(w_{ik}^{(h,l)})} \tag{9}$$

$$\boldsymbol{\phi}^{(h,l)}(\boldsymbol{H}_j^{(l)}) = V_j^{(h,l)} \tag{10}$$

$$w_{ij}^{(h,l)} = \frac{Q_i^{(h,l)} \left(K_j^{(h,l)} + B_{i,j}^{(h,l)} \right)^\top}{\sqrt{d/H}} \tag{11}$$

$$Q^{(h,l)}, K^{(h,l)}, V^{(h,l)} = \boldsymbol{H}^{(l)} W_Q^{(h,l)}, \boldsymbol{H}^{(l)} W_K^{(h,l)}, \boldsymbol{H}^{(l)} W_V^{(h,l)} \tag{12}$$

$$B^{(h,l)} = \boldsymbol{E}^{(l)} W_B^{(h,l)}. \tag{13}$$

Equations 11 and 13 show how we use edge-representation $\boldsymbol{E}^{(l)}$ to bias the scaled-dot product attention mechanism.

The linear projections $W_Q^{(h,l)}, W_K^{(h,l)}, W_V^{(h,l)} \in \mathbb{R}^{d \times \frac{d}{H}}$ and $W_B^{(h,l)} \in \mathbb{R}^{d \times \frac{d}{h}}$ used to compute queries, keys, values and attention biases within each attention head $h = 1, \ldots, H$, are learnable parameters. We assume that the model dimension d is evenly divisible by the number of heads H. Our implementation never materializes the full n by n edge-representation tensor, or attention-weight matrix, but works entirely with sparse coordinate-format matrices/tensors instead.

5.3 DTI baseline details

The DTI baseline consists of two components: a GIN to encode the molecular ligand graph and a transformer for the pocket residues.

Ligand GIN The ligand GIN receives the same $\boldsymbol{H}^{(0)}$ and $\boldsymbol{E}^{(0)}$ (restricted to the ligand molecular graph) as input, as the ablative CGNN model. One GIN message passing layer then carries out the operations

$$\begin{aligned}\boldsymbol{H}^{(l)} &= \text{LayerNorm}\left(\boldsymbol{H}^{(l-1)} + \tilde{\boldsymbol{H}}^{(l-1)}\right) \\ \tilde{\boldsymbol{H}}_i^{(l-1)} &= \text{MLP}^{(l)}\left(\boldsymbol{H}_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \text{ReLU}\left(\boldsymbol{H}_j^{(l-1)} + \boldsymbol{E}_{i,j}^{(0)}\right)\right).\end{aligned}$$

Row-wise layer normalization is added for the sake of a fairer comparison to the CGNN-* models.

Residue transformer All pocket sequences considered in this paper are made up of 85 residues. For the DTI baseline we represent them using a fix residue dictionary of size c and one-hot encoding as a matrix $\boldsymbol{X}_P \in \mathbb{R}^{85\times c}$. The residue transformer applies an initial linear transformation $W_0 \in \mathbb{R}^{d\times c}$ and adds learned positional encodings $W_P \in \mathbb{R}^{85\times d}$ resulting in an initial sequence representation

$$\boldsymbol{H}_P^{(0)} = \boldsymbol{X}_P W_0^\top + W_P \in \mathbb{R}^{85\times d}.$$

One layer of the residue transformer carries out the operation

$$\begin{aligned}\boldsymbol{H}_P^{(l)} &= \text{LayerNorm}\left(\tilde{\boldsymbol{H}}_P^{(l-1)} + \text{MLP}^{(l)}\left(\tilde{\boldsymbol{H}}_P^{(l-1)}\right)\right) \\ \tilde{\boldsymbol{H}}_P^{(l-1)} &= \text{MHA}\left(\boldsymbol{H}_P^{(l-1)}\right)\end{aligned}$$

where $\text{MHA}(\cdot)$ denotes multi-headed self attention.[48]

DTI baseline The DTI baseline uses a L'-layer residue transformer as described above and uses the ligand GIN. Its predictions are obtained as

$$\hat{y} = \text{MLP}\left(\text{LayerNorm}\left(\sum_{i=1}^{n} \boldsymbol{H}_i^{(L)} \parallel \sum_{i=1}^{85} \boldsymbol{H}_{P_i}^{(L')}\right)\right)$$

where $\parallel$ denotes concatenation, resembling the readout module as defined in Equation 6.

6 Model training

All models were implemented in PyTorch[61] and PyTorch Geometric.[62] The training and evaluation pipelines were implemented using PyTorch Lightning.[63] Models are trained to minimize the mean squared error (MSE) between predicted and ground truth affinities via L2-regularized stochastic gradient-based optimization as implemented by the PyTorch version of the AdamW[55] optimizer. During training, we monitor the validation loss on a randomly held-out subset of the training data. If it does not improve for M_{decay} epochs, we decrease the learning rate by a fixed factor. If it does not improve for $M_{stop} > M_{decay}$ epochs, we stop the training early. Our hyperparameter choices are fixed across all experiments and are documented in Table 1.

Table 1: Model and training hyperparameters.

(a) Training hyperparameters

Related symbol	Descr.	Value
-	`Optimizer`	AdamW[55]
-	`Learning rate (LR)`	1×10^{-4}
-	`Weight decay`	3×10^{-6}
-	`Batch size`	128
M_{decay}	`LR decay patience (# epochs)`	10
-	`LR decay factor`	0.9
M_{stop}	`Early stopping patience (# epochs)`	20
-	`max. # training epochs`	300

(b) CGNN-* hyperparameters

Related symbol	Descr.	Value
d	`# hidden channels`	256
L, SparseMHA$(\cdot)^{(l)}$	`# message passing layers`	3
H	`# attention heads`	8
MLP$_E^{(l)}(\cdot)$, MLP$_E^{(l)}(\cdot)$	`# hidden layers`	2

(c) DTI hyperparameters

Related symbol	Descr.	Value
d	`# hidden channels`	256
L	`# GIN message passing layers`	3
MLP$^{(l)}(\cdot)$	`# hidden layers`	2
d	`# hidden channels`	256
L'	`# residue transformer layers`	3

7 Statistical analysis

We test whether the differences between the CGNN-3D model and the non-structural baselines are statistically significant. To this end, we consider the observed test MAE as a dependent variable in a series of experiments. The combination of a particular split type and RMSD cutoff value corresponds to one of nine experiments with five subjects (one for each CV fold). The model under evaluation on a CV fold is treated as a within-factor.

All results presented in this section are computed using the Pingouin Python package.[64]

General impact of model type We first conduct a repeated measures analysis of variance (ANOVA)[65] for each experiment to assess if the type of model used has a significant effect on

Table 2: Adjusted p-values of repeated measure ANOVA testing for the effect of model type on test MAE.

	Split Type		
RMSD cutoff [Å]	Random	Scaffold	Pocket
2	0.0655	**0.015**	**0.0419**
4	**0.0001**	**0.0109**	**0.0454**
6	**0.006**	0.3695	0.69

the test MAE in general. Each ANOVA tests the null hypothesis that the mean test MAE is equal for all levels of the model factor. Since our test MAE data grouped by model type does not satisfy the sphericity[66] property required for ANOVA, p-values are adjusted using the Greenhouse-Geisser method.[67] We account for the multiplicity of our tests by additionally correcting p-values using the Bonferroni-Holm method.[68] The final adjusted ANOVA p-values are reported in Table 2.

Pairwise comparison post-hoc tests Aside from the general effect of model type, we also want to directly compare pairs of models. To this end, we conduct pairwise comparison t-tests[65] with Bonferroni-Holm correction within an experiment, for all combinations of split type and RMSD cutoff value. The resulting p-values are reported in Figure 11. Notably, we find a statistically significant difference ($p < 0.05$) between CGNN-3D and both baselines in the (Scaffold split, RMSD $\leq$ 2Å) and (Scaffold Split, RMSD $\leq$ 4Å) setting.

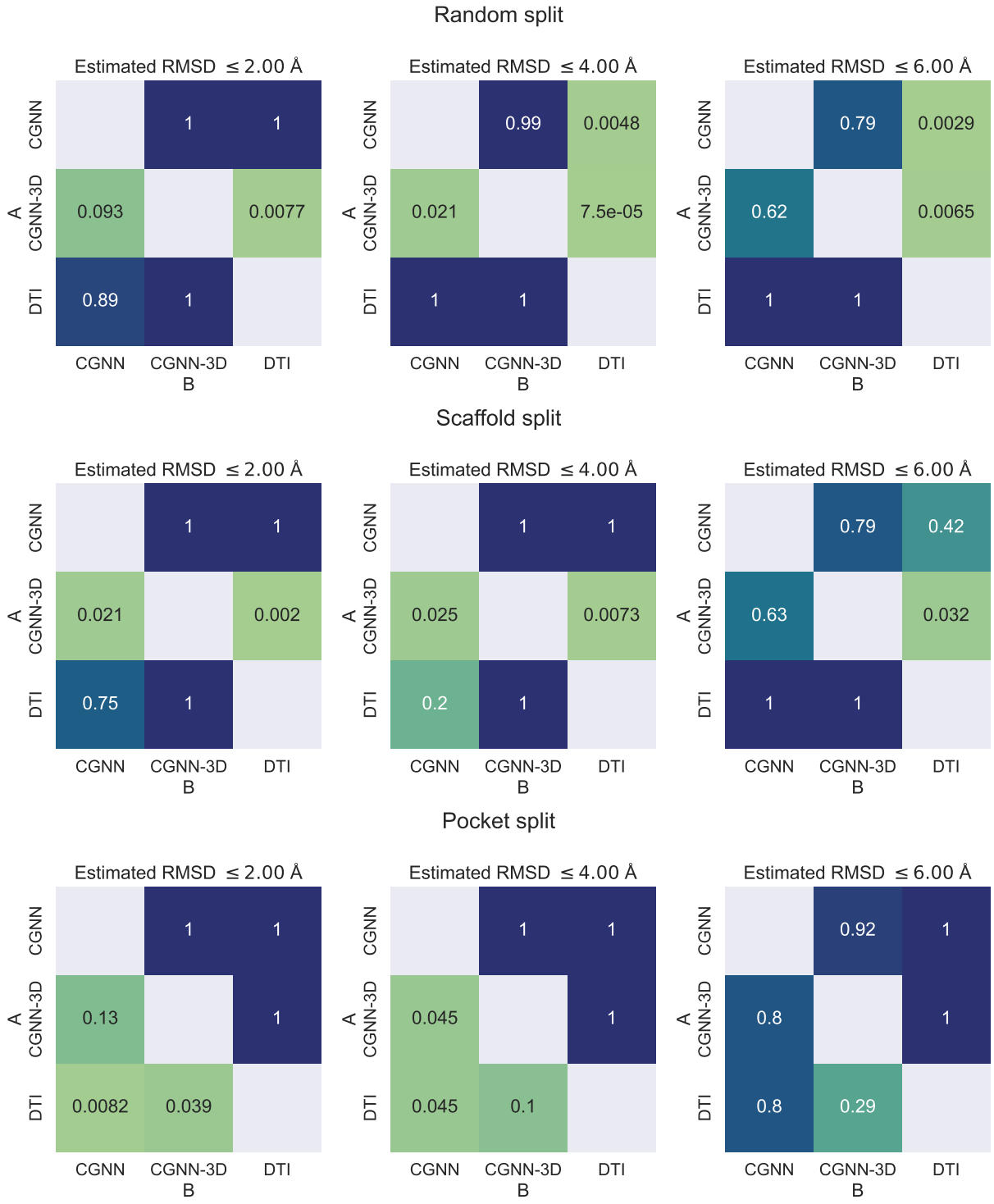

Figure 11: p-values for pairwise comparison tests. We test if the mean MAE of model A and B come from the same distribution with a one-sided alternative that tests if the MAE of model A is less than the MAE of model B (model A is better than model B in this particular metric).

8 Acronyms

3D three-dimensional

ANOVA analysis of variance

BLOSUM block substitution matrix

DTI drug target interaction

EGNN $E(3)$-invariant graph neural network

GNN graph neural network

MLP multi-layer perceptron

NN neural network

ML machine learning

RMSD root-mean-square deviation

DL deep learning

ATP adenosine triphosphate

GIN graph isomorphism network

MAE mean absolute error

MSE mean squared error

SiLU sigmoid linear unit

References

(1) Heid, E.; Greenman, K. P.; Chung, Y.; Li, S.-C.; Graff, D. E.; Vermeire, F. H.; Wu, H.; Green, W. H.; McGill, C. J. Chemprop: Machine Learning Package for Chemical Property Prediction. **2023**,

(2) Ramsundar, B.; Eastman, P.; Walters, P.; Pande, V.; Leswing, K.; Wu, Z. *Deep Learning for the Life Sciences*; O'Reilly Media, 2019.

(3) Doerr, S.; Majewski, M.; Pérez, A.; Kramer, A.; Clementi, C.; Noe, F.; Giorgino, T.; De Fabritiis, G. Torchmd: A deep learning framework for molecular simulations. *Journal of chemical theory and computation* **2021**, *17*, 2355–2363.

(4) Cremer, J.; Medrano Sandonas, L.; Tkatchenko, A.; Clevert, D.-A.; De Fabritiis, G. Equivariant graph neural networks for toxicity prediction. *Chemical Research in Toxicology* **2023**, *36*, 1561–1573.

(5) Sánchez-Cruz, N. Deep graph learning in molecular docking: Advances and opportunities. *Artificial Intelligence in the Life Sciences* **2023**, *3*, 100062.

(6) Corso, G.; Stärk, H.; Jing, B.; Barzilay, R.; Jaakkola, T. Diffdock: Diffusion steps, twists, and turns for molecular docking. *arXiv preprint arXiv:2210.01776* **2022**,

(7) Schneuing, A.; Du, Y.; Harris, C.; Jamasb, A.; Igashov, I.; Du, W.; Blundell, T.; Lió, P.; Gomes, C.; Welling, M., et al. Structure-based drug design with equivariant diffusion models. *arXiv preprint arXiv:2210.13695* **2022**,

(8) Peng, X.; Luo, S.; Guan, J.; Xie, Q.; Peng, J.; Ma, J. Pocket2mol: Efficient molecular sampling based on 3d protein pockets. International Conference on Machine Learning. 2022; pp 17644–17655.

(9) De Cao, N.; Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973* **2018**,

(10) Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems* **2020**, *33*, 1877–1901.

(11) Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; Chen, M. Hierarchical text-conditional image generation with clip latents. **2022**,

(12) Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J., et al. Learning transferable visual models from natural language supervision. International conference on machine learning. 2021; pp 8748–8763.

(13) Kim, S.; Chen, J.; Cheng, T.; Gindulyte, A.; He, J.; He, S.; Li, Q.; Shoemaker, B. A.; Thiessen, P. A.; Yu, B., et al. PubChem 2019 update: improved access to chemical data. *Nucleic acids research* **2019**, *47*, D1102–D1109.

(14) Mendez, D. et al. ChEMBL: towards direct deposition of bioassay data. *Nucleic Acids Research* **2018**, *47*, D930–D940.

(15) Chithrananda, S.; Grand, G.; Ramsundar, B. ChemBERTa: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885* **2020**,

(16) Rong, Y.; Bian, Y.; Xu, T.; Xie, W.; Wei, Y.; Huang, W.; Huang, J. Self-supervised graph transformer on large-scale molecular data. *Advances in Neural Information Processing Systems* **2020**, *33*, 12559–12571.

(17) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28*, 31–36.

(18) Wang, R.; Fang, X.; Lu, Y.; Wang, S. The PDBbind database: Collection of bind-

ing affinities for protein- ligand complexes with known three-dimensional structures. *Journal of medicinal chemistry* **2004**, *47*, 2977–2980.

(19) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I. N.; Bourne, P. E. The protein data bank. *Nucleic acids research* **2000**, *28*, 235–242.

(20) Gilson, M. K.; Liu, T.; Baitaluk, M.; Nicola, G.; Hwang, L.; Chong, J. BindingDB in 2015: a public database for medicinal chemistry, computational chemistry and systems pharmacology. *Nucleic acids research* **2016**, *44*, D1045–D1053.

(21) Wagle, S.; Smith, R. D.; Dominic III, A. J.; DasGupta, D.; Tripathi, S. K.; Carlson, H. A. Sunsetting Binding MOAD with its last data update and the addition of 3D-ligand polypharmacology tools. *Scientific Reports* **2023**, *13*, 3008.

(22) Kimber, T. B.; Chen, Y.; Volkamer, A. Deep Learning in Virtual Screening: Recent Applications and Developments. *International Journal of Molecular Sciences* **2021**, *22*.

(23) Siebenmorgen, T.; Menezes, F.; Benassou, S.; Merdivan, E.; Kesselheim, S.; Piraud, M.; Theis, F. J.; Sattler, M.; Popowicz, G. M. MISATO-Machine learning dataset of protein-ligand complexes for structure-based drug discovery. *bioRxiv* **2023**, 2023–05.

(24) Liu, C.; Kutchukian, P.; Nguyen, N. D.; AlQuraishi, M.; Sorger, P. K. A Hybrid Structure-Based Machine Learning Approach for Predicting Kinase Inhibition by Small Molecules. *Journal of Chemical Information and Modeling* **2023**, acs.jcim.3c00347.

(25) Gorantla, R.; Kubincova, A.; Weiße, A. Y.; Mey, A. S. From Proteins to Ligands: Decoding Deep Learning Methods for Binding Affinity Prediction. *Journal of Chemical Information and Modeling* **2023**,

(26) Davis, M. I.; Hunt, J. P.; Herrgard, S.; Ciceri, P.; Wodicka, L. M.; Pallares, G.;

Hocker, M.; Treiber, D. K.; Zarrinkar, P. P. Comprehensive analysis of kinase inhibitor selectivity. *Nature biotechnology* **2011**, *29*, 1046–1051.

(27) Tang, J.; Szwajda, A.; Shakyawar, S.; Xu, T.; Hintsanen, P.; Wennerberg, K.; Aittokallio, T. Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. *Journal of Chemical Information and Modeling* **2014**, *54*, 735–743.

(28) Wu, Z.; Ramsundar, B.; Feinberg, E. N.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; Pande, V. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* **2018**, *9*, 513–530.

(29) Yang, K.; Swanson, K.; Jin, W.; Coley, C.; Eiden, P.; Gao, H.; Guzman-Perez, A.; Hopper, T.; Kelley, B.; Mathea, M., et al. Analyzing learned molecular representations for property prediction. *Journal of chemical information and modeling* **2019**, *59*, 3370–3388.

(30) Manning, G.; Whyte, D. B.; Martinez, R.; Hunter, T.; Sudarsanam, S. The protein kinase complement of the human genome. *Science* **2002**, *298*, 1912–1934.

(31) Kooistra, A. J.; Volkamer, A. *Annual Reports in Medicinal Chemistry*; Elsevier, 2017; Vol. 50; pp 197–236.

(32) Kanev, G. K.; de Graaf, C.; Westerman, B. A.; de Esch, I. J. P.; Kooistra, A. J. KLIFS: an overhaul after the first 5 years of supporting kinase research. *Nucleic Acids Research* **2020**, *49*, D562–D569.

(33) Carles, F.; Bourg, S.; Meyer, C.; Bonnet, P. PKIDB: A Curated, Annotated and Updated Database of Protein Kinase Inhibitors in Clinical Trials. *Molecules* **2018**, *23*.

(34) Kelley, B. P.; Brown, S. P.; Warren, G. L.; Muchmore, S. W. POSIT: flexible shape-

guided docking for pose prediction. *Journal of Chemical Information and Modeling* **2015**, *55*, 1771–1780.

(35) Schaller, D.; Christ, C. D.; Chodera, J. D.; Volkamer, A. Benchmarking Cross-Docking Strategies for Structure-Informed Machine Learning in Kinase Drug Discovery. *bioRxiv* **2023**,

(36) Satorras, V. G.; Hoogeboom, E.; Welling, M. E(n) equivariant graph neural networks. International conference on machine learning. 2021; pp 9323–9332.

(37) Nguyen, T.; Le, H.; Quinn, T. P.; Nguyen, T.; Le, T. D.; Venkatesh, S. GraphDTA: Predicting drug–target binding affinity with graph neural networks. *Bioinformatics* **2021**, *37*, 1140–1147.

(38) Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* **2018**,

(39) Hu, W.; Liu, B.; Gomes, J.; Zitnik, M.; Liang, P.; Pande, V.; Leskovec, J. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* **2019**,

(40) Modi, V.; Dunbrack Jr, R. L. Kincore: a web resource for structural classification of protein kinases and their inhibitors. *Nucleic Acids Research* **2022**, *50*, D654–D664.

(41) Kramer, C.; Kalliokoski, T.; Gedeck, P.; Vulpetti, A. The experimental uncertainty of heterogeneous public K i data. *Journal of medicinal chemistry* **2012**, *55*, 5165–5173.

(42) Kalliokoski, T.; Kramer, C.; Vulpetti, A.; Gedeck, P. Comparability of mixed IC50 data–a statistical analysis. *PloS one* **2013**, *8*, e61007.

(43) Rogers, D.; Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling* **2010**, *50*, 742–754.

(44) OpenEye Scientific Software Inc. Santa Fe NM, OEDOCKING 4.2.0.2. http://www.eyesopen.com.

(45) Moon, S.; Zhung, W.; Yang, S.; Lim, J.; Kim, W. Y. PIGNet: a physics-informed deep learning model toward generalized drug–target interaction predictions. *Chemical Science* **2022**, *13*, 3661–3673.

(46) Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* **2016**,

(47) Ba, J. L.; Kiros, J. R.; Hinton, G. E. Layer Normalization. **2016**, arXiv:1607.06450 [cs, stat].

(48) Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.

(49) Pedregosa, F. et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.

(50) Bemis, G. W.; Murcko, M. A. The properties of known drugs. 1. Molecular frameworks. *Journal of medicinal chemistry* **1996**, *39*, 2887–2893.

(51) Henikoff, S.; Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences* **1992**, *89*, 10915–10919.

(52) Frey, B. J.; Dueck, D. Clustering by passing messages between data points. *science* **2007**, *315*, 972–976.

(53) Warren, G. L.; Andrews, C. W.; Capelli, A.-M.; Clarke, B.; LaLonde, J.; Lambert, M. H.; Lindvall, M.; Nevins, N.; Semus, S. F.; Senger, S., et al. A critical assessment of docking programs and scoring functions. *Journal of medicinal chemistry* **2006**, *49*, 5912–5931.

(54) Wang, Z.; Sun, H.; Yao, X.; Li, D.; Xu, L.; Li, Y.; Tian, S.; Hou, T. Comprehensive evaluation of ten docking programs on a diverse set of protein–ligand complexes: the

prediction accuracy of sampling power and scoring power. *Physical Chemistry Chemical Physics* **2016**, *18*, 12964–12975.

(55) Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* **2017**,

(56) Morger, A.; Svensson, F.; Arvidsson McShane, S.; Gauraha, N.; Norinder, U.; Spjuth, O.; Volkamer, A. Assessing the calibration in toxicological in vitro models with conformal prediction. *Journal of Cheminformatics* **2021**, *13*, 35.

(57) Sheridan, R. P. Time-Split Cross-Validation as a Method for Estimating the Goodness of Prospective Prediction. *Journal of Chemical Information and Modeling* **2013**, *53*, 783–790, PMID: 23521722.

(58) Martin, T. M.; Harten, P.; Young, D. M.; Muratov, E. N.; Golbraikh, A.; Zhu, H.; Tropsha, A. Does Rational Selection of Training and Test Sets Improve the Outcome of QSAR Modeling? *Journal of Chemical Information and Modeling* **2012**, *52*, 2570–2578, PMID: 23030316.

(59) Kapoor, S.; Narayanan, A. Leakage and the reproducibility crisis in machine-learning-based science. *Patterns* **2023**, *4*.

(60) Sydow, D.; Schmiel, P.; Mortier, J.; Volkamer, A. KinFragLib: exploring the kinase inhibitor space using subpocket-focused fragmentation and recombination. *Journal of Chemical Information and Modeling* **2020**, *60*, 6081–6094.

(61) Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **2019**, *32*.

(62) Fey, M.; Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* **2019**,

(63) Falcon, W. A. Pytorch lightning. *GitHub* **2019**, *3*.

(64) Vallat, R. Pingouin: statistics in Python. *The Journal of Open Source Software* **2018**, *3*, 1026.

(65) Montgomery, D. C. *Design and analysis of experiments*; John wiley & sons, 2017.

(66) Mauchly, J. W. Significance test for sphericity of a normal n-variate distribution. *The Annals of Mathematical Statistics* **1940**, *11*, 204–209.

(67) Greenhouse, S. W.; Geisser, S. On methods in the analysis of profile data. *Psychometrika* **1959**, *24*, 95–112.

(68) Holm, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics* **1979**, 65–70.