A Resource Availability and Licensing

All datasets are available for download from `https://www.atom3d.ai`. The corresponding code for dataset processing and model training is maintained at `https://github.com/drorlab/atom3d`, along with instructions for installing our Python package. Further details can be found in the documentation at `atom3d.readthedocs.io/en/latest`. All datasets are hosted on Zenodo under individual DOIs, and are licensed under Creative Commons (CC) licenses. Full licenses, DOIs, and other details can be found on our website and Github. The authors bear all responsibility in case of violation of rights, and confirm the CC licenses for the included datasets.

B Broader Impact

The methods and datasets presented here are intended to promote machine learning research that uses molecular structure. We expect that such research can be used in the development of new medicines and materials, as well as lead to a better understanding of human health. At the same time, the systems we propose are inherently complex, and thus can result in a greater degree of difficulty in interpreting their results, as well as preventing and assessing errors. These errors in turn can have severe consequences in both the development of new medicines, as well as in the treatment of patients using advances derived from this work. Careful evaluation of such systems results, as well as further research in increasing our ability to interpret their outputs, can help mitigate these concerns. Additionally, the benchmarks we use focus on molecules that have known structures. This is a biased set of all molecules, specifically in that they are molecules other researchers have chosen to focus on. Thus, we might expect trained methods to reflect these biases, and therefore have better performance in cases that are already being studied by the broader research community. This could lead to neglecting systems that are not as prioritized.

C Working with 3D Molecular Data Using ATOM3D

In order to facilitate the entry of new practitioners to the field of 3D molecular learning, we provide some high-level guidelines for working with the datasets we provide and for curating new ones. We provide computational tools required for these tasks in the `atom3d` package, and will be continuing to develop and support its functionality moving forward.

C.1 Assembling New Datasets

Data sources and repositories. The success of deep learning methods strongly depends on the availability of sufficient training data. Unless they have the capabilities to produce the necessary data, most scientists will rely on use public databases. The go-to repository for protein structures is the Protein Data Bank (PDB).[3] RNA structures can be found at the RNA 3D hub of Bowling Green State University.[4] An exhaustive repository for small molecules is ChEMBL,[5] though the 3D structures of small molecules are mostly not directly deposited. They can be generated by expensive quantum-chemical methods or in good approximation by cheminformatics tools such as RDKit. Many more specific databases are out there and worth being explored. `atom3d` provides methods to mine and convert data from many common formats in the field (e.g. PDB, SDF, XYZ) into our standardized dataset format.

Scope and limitations of the data. Even the most extensive databases cannot capture the large diversity of biological macromolecules or the space of potential drug molecules. It is therefore necessary to think about the scientific problem at hand and whether the available data adequately represents the range of structures that are responsible for the studied effects. An important general limitation of structural data is that molecules change conformation fluidly in real life due to thermal fluctuations and other effects. Additionally, interactions with other molecules, disordered regions, or environmental factors like pH can result in significant differences from their experimentally determined forms.

---

[3] `https://www.rcsb.org/`

[4] `http://rna.bgsu.edu/rna3dhub/nrlist/`

[5] `https://www.ebi.ac.uk/chembl/`

Incomplete or corrupted data. Structural data is rarely perfect. Experimental uncertainties are mostly caused by limited resolution of the involved techniques such as X-ray crystallography or electron cryo-microscopy. Computationally generated structures are also prone to flaws in the underlying modeling programs (e.g., molecular force fields).

These limitations can lead to problems such as unrealistic conformations, missing or duplicate atoms, non-resolved amino acid side chains, and more. One has to decide whether to keep those structures or to sanitize them using computational tools. Additionally, hydrogen atoms are often not included in the data and, if needed for the task, have to be added when assembling the dataset. The most important guideline here is to be consistent and clear in the way these issues are treated. Sometimes it can be necessary to assemble two different datasets with different treatments of missing data. By providing a standardized format and functions for processing and filtering datasets through `atom3d`, we hope to simplify this process.

C.2 Developing and Benchmarking New Algorithms

Reading and preprocessing. Algorithms represent data in various ways and a given dataset is not always compatible with the representation needed for the algorithm. For example, certain structures, residues, or atoms may need to be filtered out. Ideally, these steps are considered as dataset preparation and are separated from the algorithm itself, i.e. not hard-coded into the dataloader. This has two main advantages: (1) it saves time upon multiple reruns of the algorithm as structural data can be large and expensive to process, and (2) saving the preprocessed input dataset separately increases reproducibility, because small differences in preprocessing are often not recorded. We provide our benchmarking datasets in a format that is easy to read for most Python-based algorithms, and a simple interface for applying arbitrary transforms to convert between data formats (e.g. graph representations or voxelized grids).

Comparing algorithms. Predictions can be tested with various metrics. Depending on the prediction problem, some of the metrics grasp the scientific aims of the training better than others. It is usually recommended to stick to the metrics that are common in the field and are given in the benchmarks. As science develops, new metrics for a specific problem might come up. These should be well justified and the old metrics should still be reported alongside them to allow for a comparison. Ideally, the new metrics are calculated for older models, too. To facilitate this in advance, when benchmarking an algorithm, specific predictions should be stored and not only metrics. In `atom3d`, we provide standardized evaluation classes to calculate the appropriate metrics from the outputs of a given algorithm.

Interpretation of results. When judging the performance of an algorithm, one should take into account the experimental uncertainties both in the structures but also in the label data. While small molecules can be investigated in much detail, it will rarely be possible to get near perfect performance for tasks involving complex biological macromolecules. Over time, even held-out test sets become part of the selection process for new methods as only those methods that perform better on the test set will prevail. A measured improvement can thus be caused by minor specifics of the test set. As the field matures and performance becomes saturated, the benchmark sets will still be valid as sanity checks for new methods, but harder tasks will be the ones driving new development. We anticipate that new tasks and datasets will be added to ATOM3D as the field evolves.

D Dataset Preparation

We present a set of methods to mine task-specific atomic datasets from several large databases (e.g. PDB) as well as to filter them, split them, and convert them to a format suitable for standard machine learning libraries (esp. PyTorch and TensorFlow). We store these datasets in LMDB format, where each atom is stored as a row in a standardized data frame. This data format accurately captures the natural hierarchy of atom subgroups in biomolecules, especially proteins, and enables data loading and processing to be consistent across datasets, tasks, and computational environments. The LMDB format also allows for labels and additional metadata is stored along with the atoms dataframe for each datapoint.

To capture hierarchical information in a way that is task-specific but standardized, we define an "ensemble" to be the highest-level of structure for each example, e.g. the PDB entry for the protein. Within each ensemble, we define "subunits", which represent the specific units of structure used for that task. For example, for the paired tasks (PIP, LEP, MSP), there is one subunit corresponding to each structure in the pair; for RES, there is one subunit for each residue microenvironment, and for structure ranking (PSR, RSR), there is one subunit for each candidate 3D structure. In this way, it is simple to iterate over each dataset and extract each atomistic structure, which can then be augmented and processed into any desired format (e.g. voxelized for the 3DCNN, converted to graphs for the GNN).

In the following sections, we describe the specific steps used to mine and process each dataset.

D.1 Small Molecule Properties (SMP)

The QM9 dataset is processed from the files provided on Figshare [Ramakrishnan et al., 2014a]. The properties included in QM9 are the following:

- μ - Dipole moment (unit: D)
- α - Isotropic polarizability (unit: bohr^3)
- ϵ_{HOMO} - Highest occupied molecular orbital energy (unit: Ha, reported in eV)
- ϵ_{LUMO} - Lowest unoccupied molecular orbital energy (unit: Ha, reported in eV)
- ϵ_{gap} - Gap between HOMO and LUMO (unit: Ha, reported in eV)
- R^2 - Electronic spatial extent (unit: bohr^2)
- ZPVE - Zero point vibrational energy (unit: Ha, reported in meV)
- U_0 - Internal energy at 0 K (unit: Ha)
- U_{298} - Internal energy at 298.15 K (unit: Ha)
- H_{298} - Enthalpy at 298.15 K (unit: Ha)
- G_{298} - Free energy at 298.15 K (unit: Ha)
- C_v - Heat capacity at 298.15 K (unit: $\frac{\mathrm{cal}}{\mathrm{mol\,K}}$)

It is common to subtract the reference thermochemical energy from U_0, U_{298}, H_{298}, G_{298} to obtain:

- U_0^{at} - Atomization energy at 0K (unit: Ha, reported in eV)
- U_{298}^{at} - Atomization energy at 298.15K (unit: Ha, reported in eV)
- H_{298}^{at} - Atomization enthalpy at 298.15K (unit: Ha, reported in eV)
- G_{298}^{at} - Atomization free energy at 298.15K (unit: Ha, reported in eV)

We report metrics for these quantities in the benchmark.

As recommended by the authors of the original dataset, we exclude 3,054 molecules that do not pass a geometrical consistency test [Ramakrishnan et al., 2014b]. Additionally, we excluded all 1,398 molecules that RDKit is unable to process - as in former GNN work [Fey and Lenssen, 2019]. In this way, we ensure that all models in this work can be trained on the same data. Following previous work [Wu et al., 2018, Gilmer et al., 2017, Schütt et al., 2017, Anderson et al., 2019], we split the remaining dataset randomly in training, validation, and test set - containing 103547, 12943, and 12943 molecules, respectively.

D.2 Protein Interface Prediction (PIP)

For our test set, we download the cleaned PDB files from the DB5 dataset as provided in [Townshend et al., 2019], and convert to our standardized format. Each complex is an ensemble, with the bound/unbound ligand/receptor structures forming 4 distinct subunits of said ensemble. We use the bound forms of each complex to define neighboring amino acids (those with any heavy atoms within 6 Å of one another), and then map those onto the corresponding amino acids in the unbound forms of the complex (removing those that do not map). These neighbors are then included as the positive examples, with all other pairs being defined as negatives. At prediction time, we attempt to re-predict which possible pairings are positive or negative, downsampling negatives to achieve a 1:1 positive to negative split. We use the unbound subunits as our pair of input structures for testing. We use AUROC of these predictions as our metric to evaluate performance.

For our training set, we reproduce the Database of Interacting Protein Structures (DIPS) [Townshend et al., 2019]. Specifically, we take the snapshot of all structures in the PDB from November 20, 2015. We apply a number of filtering operations, removing structures with no protein present, structures with less than 50 amino acids, structures with worse than 3.5 Å resolution, and structures not solved by X-ray crystallography or Cryo-EM. We then split the dataset into all pairs of interacting chains. These pairs form our ensembles, with each of the two chains being one subunit. We then remove pairs with less than 500 Å^2 buried surface area as measured by the FreeSASA Python library [Mitternacht, 2016] (using total area computed the naccess classifier, including hydrogens and skipping unknown residues). Furthermore, to ensure there is no train/test contamination, we prune this set against the DB5 set defined above, removing any pairs that have a chain with more than 30% sequence identity, using the software BLASTP [Altschul et al., 1990]). We also prune the set based on structural similarity, removing any pairs in DIPS that map to corresponding SCOP [Andreeva et al., 2014] pairs of superfamilies that are also present across a pair in DB5 (i.e., we remove a DIPS pair if the first subunit in that pair has a chain with a SCOP superfamily that is present in an unbound subunit of a DB5 pair, and the second subunit in that DIPS pair also has a SCOP superfamily that is present in the other unbound subunit of that same DB5 pair). Once this pruning is done, we split the DIPS set into a training, validation, and (internal) testing set based on PDB sequence clustering at a 30% identity level, to ensure little contamination between them. We perform a 80%, 10%, 10% split for training, validation, and testing, respectively. Note this internal testing set is not used for performance reporting.

D.3 Residue Identity (RES)

Environments are extracted from a non-redundant subset of high-resolution structures from the PDB. Specifically, we use only X-ray structures with resolution $<$3.0 Å, and enforce a 60% sequence identity threshold. We then split the dataset by structure based on domain-level CATH 4.2 topology classes [Dawson et al., 2017], as described in [Anand et al., 2020]. This resulted in a total of 21147, 964, and 3319 PDB structures for the train, validation, and test sets, respectively. Rather than train on every residue for each of these structures, we balance the classes in the train set by downsampling to the frequency of the least-common amino acid (cysteine). The original class balance is preserved in the test set. In total, the train, validation, and test sets comprised 3733710, 188530, and 1261342 environments, respectively. We ignore all non-standard residues. We represent the physico-chemical environment around each residue using all C, O, N, S, and P atoms in the protein and any co-crystallized ligands or ions. All non-backbone atoms of the target residue are removed, and each environment is centered around a "virtual" Cβ position of the target residue defined using the average Cβ position over the training set.

D.4 Mutation Stability Prediction (MSP)

Mutation data are extracted from the SKEMPI 2.0 database [Jankauskaitė et al., 2019]. Non-point mutations or mutants that cause non-binding of the complex are screened out. Additionally, mutations involving a disulfide bond and mutants from the PDBs 1KBH or 1JCK are ignored due to processing difficulties. A label of 1 is assigned to a mutant if the K_d of the mutant protein is less than that of the wild-type protein, indicating better binding, and 0 otherwise. Atoms from the twenty canonical amino acids were extracted from the PDBs provided in SKEMPI using PyMOL [Schrödinger, LLC, 2015], and in silico mutagenesis is accomplished using PyRosetta [Chaudhury et al., 2010], where dihedrals within 10 Å of the mutated residue are repacked. This protocol produces 893 positive examples and 3255 negative examples. For ENN training, we use structures that are reduced to a size that is tractable for the implementation we used. To this end, we only selected the regions within a radius of 6 Å around the Cα-atom of the mutated residue. For 3DCNNs, we analogously used a radius of 25 Å. GNNs are trained on complete structures. This dataset is split by sequence identity at 30%.

D.5 Ligand Binding Affinity (LBA)

PDBBind contains X-ray structures of proteins bound to small molecule and peptide ligands. We use the "refined set" (v.2019) consisting of 4,852 complexes filtered for various quality metrics, including resolution ≤ 2.5 Å, R-factor ≤ 0.25, lack of steric clashes or covalent bonding, and more [Li et al., 2014]. We further exclude complexes with invalid ligand bonding information. The binding affinity provided in PDBBind is experimentally determined and expressed in terms of inhibition constant (K_i) or dissociation constant (K_d), both in Molar units. As in previous works [Ballester and Mitchell, 2010, Zilian and Sotriffer, 2013, Ragoza et al., 2017, Jiménez et al., 2018], we do not make the distinction between K_i and K_d, and instead predict the negative log-transformed binding affinity, or pK. The majority of prior scoring functions have used the "core set" provided by the Critical Assessment of Scoring Functions (CASF) [Su et al., 2019] as a test set for evaluating prediction performance. However, by construction every protein in this test set is at least 90% identical to several proteins in the training set. Thus, performance on this test set does not accurately represent the generalizability of a scoring function, and has been shown to overestimate the performance of machine learning models in particular [Kramer and Gedeck, 2010, Gabel et al., 2014, Li and Yang, 2017]. Therefore, to prevent overfitting to specific protein families, we create a new train/validation/test split based on a 30% sequence identity threshold to limit homologous proteins appearing in both train and test sets. Specifically, we use a cluster-based approach to ensure that no protein in the training set has $> 30\%$ sequence identity to any protein in the validation or test sets, as calculated by BLASTP. To prevent overrepresentation of any single protein family (i.e. sequence identity cluster), we additionally enforce that no single cluster represents more than 20% of the overall split. Splitting using this procedure resulted in training, validation, and test sets of size 3507, 466, and 490, respectively.

For comparison, we provide an additional, less restrictive, split based on a 60% sequence identity threshold (results in Table 8). This leads to training, validation, and test sets of size 3678, 460, and 460, respectively.

For the ENN, we use a reduced dataset without hydrogens and only the most abundant heavy elements in the full dataset (C, N, O, S, Zn, Cl, F, P, Mg). From the binding pocket, we only use atoms within a distance of 6 Å from the ligand and only so many atoms as to not have more than 600 atoms in total (ligand + protein). This limitation of atom numbers is purely technical. The Kronecker products involved in the covariant neurons are memory intensive in the Cormorant implementation we used, and training on larger structures was limited by the memory of the GPUs available to us.

D.6 Ligand Efficacy Prediction (LEP)

Each input consists of a ligand bound to both the active and inactive conformation of a specific protein. The goal is to predict the label for this drug/ligand, either an "activator" or "inactivator" of the protein function. Why include these protein conformations in the input? From a biochemical perspective, if the drug binds much more favorably to the active protein conformation, it will act as an activator of the protein function. The model may then learn this differential binding strength to improve predictions of ligand function.

Pairs of structures for 27 proteins are obtained through manual curation of the Protein Data Bank structures where "active" and "inactive" conformational states are both available. For example, for ion channels, this means a channel in an open vs. closed state. 527 ligands with known protein binding and labeled function are selected from the IUPHAR database. We label ligands as activators if they are listed as "agonists" or "activators" and label ligands as inactivators if they are listed as "inhibitors" or "antagonists". We select up to 15 of both activating and inactivating ligands for each protein.

We model the drugs bound to the relevant protein. To prepare protein structures for use in docking, we first prepare structures using the Schrödinger suite. All waters are removed, the tautomeric state of the ligand present in the experimentally determined structure is assigned using Epik at pH 7.0 +/–2.0, hydrogen bonds are optimized, and energy minimization is performed with non-hydrogen atoms constrained to an RMSD of less than 0.3 Å from the initial structure. For ligands to be docked, the tautomeric state is assigned using Epik tool at target pH 7.0. Ligands are docked using default Glide SP. This results in 527 pairs of complexes. These are split into training, validation, and tests sets by protein target to ensure generalizability across proteins.

For ENN training, we reduce the structures to a size that is tractable for the Cormorant implementation we used. To this end, we only use the regions within a radius of 5.5 Å around the ligand. For 3D-CNNs, we use a radius of 25 Å. GNNs were trained on complete structures.

We require that efficacy predictions for a given ligand at a given protein not use information about efficacy of other ligands at that protein, to model a case when no such information is available. When efficacy measurements are available for other ligands at the same protein—as is the case for many well-studied drug targets—methods that take advantage of these (e.g., quantitative structure-activity relationship methods) may produce more accurate efficacy predictions.

D.7 Protein Structure Ranking (PSR)

The Critical Assessment of Structure Prediction (CASP) [Kryshtafovych et al., 2019] is a long-running international competition held biennially, of which CASP13 is the most recent, that addresses the protein structure prediction problem by withholding newly solved experimental structures (called *targets*) and allowing computational groups to make predictions (called *decoys*), which are then evaluated for their closeness to their targets after submission. Those submissions are then carefully curated and released as decoy sets in two stages (20 decoys per target for Stage 1, 150 decoys per target for Stage 2) for the Model Quality Assessment (MQA), one of the categories in CASP which aims to score a set of decoys of a target based on how closely they are to the target. For the PSR dataset, we compiled those decoys sets released in CASP5-13, then relaxed those structures with the SCWRL4 software [Krivov et al., 2009] to improve side-chain conformations. For all decoys in the dataset, we computed the RMSD, TM-score, GDT_TS, and GDT_HA scores using the TM-score software [Zhang and Skolnick, 2007].

Mirroring the setup of the competition, we split the decoy sets based on target and released year. More specifically, we randomly split the targets in CASP5-10 and randomly sample 50 decoys for each target to generate the training and validation sets (508 targets for training, 56 targets for validation), and use the whole CASP11 Stage 2 as test set (85 targets total, with 150 decoys for each target). We chose CASP11 as test set, as the targets in CASP12-13 are not fully released yet.

D.8 RNA Structure Ranking (RSR)

The RNA Puzzles competition [Cruz et al., 2012] is a rolling international competition dealing with the RNA structure prediction problem. Similarly to CASP, newly solved experimental structures, referred to as natives, are withheld until computational groups make prediction, referred to as candidates. These candidates are then evaluated by their RMSD from the native. For this task, we use candidate structures created by the state-of-the-art structure generation method, FARFAR2 [Watkins et al., 2020], for each of the 21 first RNA Puzzles. These are made available as part of the FARFAR2 publication. There are an average of 21303 (standard deviation of 13973) candidates generated per puzzles, with a large range of RMSDs. For the RSR dataset we randomly sample 1000 candidates per puzzle. We split temporally, by puzzle, using RNA Puzzles 1-13 for training, 14-17 (excluding 16) for validation, and 18-21 for testing.

E Task-Specific Experimental Details

Below we describe the architectures and hyperparameters used for benchmarking. In general, these are intended to be robust but simple benchmarks for each task, so we did not undertake full tuning of every hyperparameter for every task, which would be very expensive. However, we did tune specific crucial hyperparameters such as learning rate, number of epochs, and 3DCNN grid size/resolution using a grid search methodology. Final models and hyperparameter settings were selected using performance on the validation set, with the held-out test sets only used to report final performance.

E.1 3DCNNs

Our base 3DCNN architecture consists of four 3D-convolutional layers with increasing filter size (32, 64, 128, and 256) — each followed by ReLU activation, max-pooling (for every other convolution layer), and dropout — and one fully-connected layer of size 512, followed by ReLU activation and dropout. For single model task (PSR, RSR, LBA, SMP), we add an additional fully-connected layer to transform to the required output dimension size. For paired tasks (PIP, LEP, MSP), we adapt this base architecture into a twin network, add an additional fully-connected layer followed by ReLU activation and dropout to combine the output of each member of the pair, and finally add a final fully-connected layer to transform to the required output dimension size, as in [Townshend et al., 2019].

For input to the 3DCNNs, we represent our data as cube in 3D space of certain radius (40 Å for PSR, RSR; 17 Å for PIP; 20 Å for LBA; 7.5 Å for SMP; 25 Å for LEP, MSP; 10 Å for RES) that are discretized into voxels with resolution of 1 Å to form a grid (for PSR and RSR, we need to decrease the resolution to 1.3 Å in order to fit them in the GPU memory). For paired tasks (PSR, RSR, and PIP), we form a separate voxel grid for each member of the pair. For most tasks, we use the centroid of each input structure as center of the grid, excluding LBA where we use the centroid of the ligand as center and MSP where we use the mutation point as center. Each grid voxel is associated with a binary feature vector which encodes the presence or absence of each specified atom type in that voxel. For PSR, RSR, PIP, and RES, we encode the presence of heavy atoms C, O, N, and S (P for RSR since S does not exist in RNA structures). For other tasks where hydrogen bonds might play an important role, we encode the hydrogen atom (H) in addition to C, O, N, and few other abundant atoms (F for LBA and SMP; S, Cl, F for LEP; S for MSP). To encode rotational symmetries, we apply a data augmentation strategy in which we apply 20 random rotations to the input grid, as in [Townshend et al., 2019], except for RES, where we instead apply the canonicalization procedure described in [Anand et al., 2020].

For binary classification tasks, we use binary cross-entropy weighted by the class distribution (i.e. rarer class is weighted more heavily on mistakes). To address issues with imbalanced datasets, we randomly oversample/undersample the less/more frequent class respectively during training. For regression tasks, we use mean-squared error loss for training. All models were trained with Adam optimizer with default beta parameters and learning rate 0.0005 for SMP; 0.0001 for PSR, RSR, PIP, RES; 0.001 for LBA; and 0.00001 for LEP, MSP. We monitor the loss on the validation set at every epoch. The weights of the best-performing are then used to evaluate on the held-out test set. The models were all trained on 1 Titan X(p) GPU for 4–24 hours depending on the task.

E.2 GNNs

Our base GNN architecture consists of five layers of graph convolutions as defined by Kipf and Welling [Kipf and Welling, 2016], with increasing hidden dimension (64, 128, 128, 256, 256) each followed by batch normalization and ReLU activation. For tasks with paired input structures (PIP, LEP, MSP), we apply this convolutional architecture to each input structure separately in a twin network architecture with tied weights, and then concatenate the outputs before passing through two fully-connected layers of size 256 to transform to an output dimension of one neuron for binary classification. We regularize using dropout with a probability of 0.25 after the first fully-connected layer. Some tasks require classification of an entire structure, and thus are well-suited to graph-level outputs (PSR, RSR, LBA). Here, we apply global mean or addition pooling across all nodes before applying the final two layers. For PIP, RES, and MSP, instead of pooling we instead extract the embedding of the node corresponding to the Cα atom of the residue in question (interacting residue, deleted residue, and mutated residue, respectively) after the final convolutional layer. For SMP, we use the previously-developed architecture presented in [Gilmer et al., 2017], which is publicly available.

We use a very simple featurization scheme for atomic systems. We define edges between all atoms separated by less than 4.5 Å. Edges are weighted by the distance between the atoms, and nodes are featurized by one-hot-encoding all heavy (non-hydrogen) atoms. The only exception is SMP, where we adopt the established featurization scheme used in MoleculeNet [Wu et al., 2018]. For tasks involving protein-ligand binding (LBA and LEP), we distinguish the protein and the ligand by using separate channels in the node features for each. All GNNs were implemented in PyTorch Geometric [Fey and Lenssen, 2019].

For binary tasks, we use a binary cross-entropy loss criterion weighted by the class distribution (e.g. a 1:4 positive:negative ratio would result in positive examples being up-weighted four-fold). For regression tasks, we use a mean-squared error criterion. For all models, we train with the Adam optimizer with learning rate 0.0001 (except for PIP, which uses a learning rate of 0.001) and monitor the relevant metrics (see Table 8) on the validation set after every epoch. The weights of the best-performing are then used to evaluate on the held-out test set. Models were all trained using 1 Tesla V100 GPU for 4–48 hours depending on the task.

Certain tasks involve making a prediction on a specific amino acid (PIP, RES, and MSP; see Table 2), yet GNNs typically rely on summing over all node embeddings to compute a final graph embedding, making it difficult to isolate this amino acid. To remedy this, after our convolutional layers we extract the embedding of only the Cα atom of the amino acid in question, thereby allowing our GNNs to isolate it.

E.3 ENNs

For the core of all Cormorant architectures in this work, we use a network of four layers of covariant neurons that use the Clebsch–Gordan transform as nonlinearity, with $L = 3$ as the largest index in the $SO(3)$ representation and 16 channels, followed by a single $SO(3)$-vector layer with $L = 0$. An input featurization network encodes the atom types as one-hot vectors. For SMP, input and output are passed through multi-layer perceptrons (MLP) as in [Anderson et al., 2019]. For the input, a weighted adjacency matrix with a learnable cut-off radius is constructed. This mask is passed alongside the input vector through a MLP with a single hidden layer with 256 neurons and ReLU activation. The output network is constructed from a set of scalar invariants that are passed through a network of two MLPs. Each of these MLPs has a single hidden layer of size 256, and the intermediate representation has 96 neurons. For LBA and LEP, input and output layers are a single learnable mixing matrix, as used in the original Cormorant implementation for MD-17[Anderson et al., 2019]. The twin networks required for LEP and MSP was constructed by training two ENNs that are then connected by concatenating the single-network outputs which are then passed to a MLP analogous to the one described above for SMP. For MSP, the two structures corresponded to the wild-type structure and the mutated one; for LEP to the active and inactive one. We extend the original Cormorant implementation to handle classification problems (binary and multi-class) and the twin network architecture. Our implementation[6] also allows to set a boundary on the Clebsch-Gordan product to eliminate training instabilities from a divergent loss that would otherwise arise occasionally for some of the tasks.

We use MSE loss for regression tasks and cross-entropy loss for classification tasks. For all tasks, we used the AMSgrad optimizer with an initial learning rate of 0.001 and a final learning rate of 0.00001, decaying in a cosine function over the training process. We trained SMP and LBA for 150 epochs, LEP and MSP for 50 epochs, and RES for 30 epochs. We monitor the loss for the validation set after every epoch. The weights of the best-performing are then used to evaluate on the held-out test set. The models were all trained on 1 Titan X(p) GPU for 1–5 days depending on the task.

F 1D and 2D Baselines

For each task, we select a baseline that fulfills the following criteria: (1) represents the current state-of-the-art (SOTA) for that task—or as close as possible—using only 1D (sequence only) or 2D (sequence and/or bond connectivity) molecular representations, and (2) either has a publicly available implementation or has reported results for the same task and splitting criteria. For PSR and RSR, which are inherently 3D tasks and have no appropriate 1D or 2D representation, we compare to the state-of-the-art 3D methods instead. Below we describe the choice and implementation of baseline models for each task.

F.1 SMP

As a 2D method for predicting molecular properties, we choose molecular GNNs [Tsubaki et al., 2019] which are based on learning representations of subgraphs in molecules. We use an implementation that only uses the SMILES representation of the molecular graph.[7] As an additional 2D baseline, we compare to N-Gram Graph XGB [Liu et al., 2019]. This method is based on an unsupervised representation called N-gram graph which first embeds the vertices in the molecule graph and then

---

[6] `https://github.com/drorlab/cormorant`

[7] `https://github.com/masashitsubaki/molecularGNN_smiles`

assembles the vertex embeddings in short walks in the graph. This representation is combined with the XGBoost learning method [Chen and Guestrin, 2016].[8]

F.2 PIP

For the PIP task, our non-3D method is the BIPSPI [Sanchez-Garcia et al., 2018] model, a gradient-boosted decision tree. We compare to their model that uses only sequence and sequence conservation features and is evaluated on DB5.

F.3 RES

As a 1D sequence-based model for predicting residue identity, we choose the transformer architecture TAPE, introduced by [Rao et al., 2019]. We use their reported accuracy on heldout families for language modeling, as that corresponds to a sequence-only version of our RES tasks, with similar stringency in terms of splitting criteria.

F.4 MSP

We use the publicly provided implementation of TAPE [Rao et al., 2019][9]. We modify their sequence-to-sequence head to predict the effect of mutations at specific positions, using the original unmutated protein as the input sequence and writing the output sequence as a one-hot-encoded 20-dimensional vector, indicating if a given mutation would be beneficial or detrimental. Note that the vast majority of positions would be unlabeled and therefore not included in the learning task.

F.5 LBA

As a 1D method for predicting ligand binding affinity, we choose DeepDTA [Öztürk et al., 2018][10], a 1DCNN based model that takes in pairs of ligand SMILES string and protein sequence as input. We use the same hyperparameters as in the original paper for the baseline.

We also compare our results against DeepAffinity [Karimi et al., 2019][11]. We compare to their unified RNN/RNN-CNN model that takes in pairs of ligand SMILES string and their novel representations of structurally-annotated protein sequences (SPS/Structure Property-annotated Sequence) as input. Per the authors' recommendation, we use the DSSP software [Joosten et al., 2010, Kabsch and Sander, 1983] to generate the protein secondary structure and the protein relative solvent accessibility used in the SPS representation directly from the protein 3D structure, rather than the predicted ones by the SSpro/ACCpro software [Magnan and Baldi, 2014, Cheng et al., 2005] as done in the DeepAffinity paper. We use the same hyperparameters as in the original paper for the baseline, except for the maximum SMILES string and SPS lengths which we increase from 100 and 152 in the paper to 160 and 168, respectively, to accommodate for larger ligands/proteins in the PDBBind dataset. We used the pre-trained seq2seq encoders for proteins and ligands to initialize the joint supervised training of the encoders and CNN, and trained the DeepAffinity models for 1000 epochs. The pre-trained DeepAffinity seq2seq encoders were trained with maximum SMILES string and SPS lengths of 100 and 152, however, there are only 4% of the ligands in the PDBBind dataset with SMILES string length larger than 100, and even much smaller percentage of the proteins (around 0.2%) with SPS length larger than 152, so the input data distribution for PDBBind should still be in the range of that of DeepAffinity.

F.6 LEP

We train DeepDTA [Öztürk et al., 2018][10] (with the same hyperparameters as in the original paper) on the LEP dataset as baseline. As the inherent protein sequences and ligand SMILES strings are the same for both the inactive and active structures, the problem is reduced to binary classification task given a pair of protein sequence and the ligand SMILES string, and does not require modifying the DeepDTA architecture to make the twin network as in the GNN, ENN, or 3DCNN case.

---

[8] `https://github.com/chao1224/n_gram_graph`

[9] `https://github.com/songlab-cal/tape`

[10] `https://github.com/hkmztrk/DeepDTA`

[11] `https://github.com/Shen-Lab/DeepAffinity`

F.7 PSR

We compare our results against the state-of-the-art single-model methods as reported in [Pagès et al., 2019]. These include 3DCNN [Hou et al., 2019] and Ornate [Pagès et al., 2019], 3DCNN voxel-based methods trained on structural information, and Proq3D [Uziela et al., 2017], a deep-learning based method which employs structural information, Rosetta energy terms [Leaver-Fay et al., 2011], and evolutionary information derived from the amino acid sequence. We exclude ProteinGCN [Sanyal et al., 2020], a recent GNN-based method, from comparison as they do not provide results on CASP11 dataset.

F.8 RSR

For RNA structure ranking, we compare our results against the Rosetta scoring function [Alford et al., 2017]. In past RNA Puzzles competitions, methods using the Rosetta scoring function have been found to most consistently produce the lowest RMSD candidates. This is a physical- and knowledge-based potential specifically tuned for biomolecular structure.

G State-Of-The-Art Methods

When possible, for tasks in Table 7, we choose 3D methods that fulfill the following criteria: (1) they represent the current state-of-the-art for that task, or as close as possible, and (2) they either have a publicly available implementation or have reported results for the same task and splitting criteria. Here our choice of methods is described in more detail if not already discussed in the section above.

G.1 SMP

We compare to the state of the art, i.e., the best achieved prediction on each task, as reported in Anderson et al. [2019]. Many methods have been tested on QM9 and have reached excellent performance which makes them comparably hard to beat for new methods. In general, the best methods for QM9 so far are message passing neural networks Gilmer et al. [2017], continuous-filter convolutional neural networks Schütt et al. [2017], and Cormorant Anderson et al. [2019]. Differences in performance between earlier Cormorant studies and this work can be attributed to the different (random) split.

G.2 PIP

We compare our results against the BIPSPI Sanchez-Garcia et al. [2018] model, a gradient-boosted decision tree. In contrast to the 1D/2D baseline comparison to BIPSPI, in this case we compare against their model that employs both structural- and sequence-based amino acid features.

G.3 RES

Since there have been no standardized datasets for this task to date, it is difficult to perform a direct comparison of methods. The closest comparison for a CNN trained on a balanced dataset of residue environments is 0.425, as reported in Torng and Altman [2017]. While higher performance was reported by Anand et al. [2020] (accuracy 0.572), this model was trained on an unbalanced dataset comprising every standard residue environment in all training set PDBs. Similar performance has also been reported with other deep learning architectures Weiler et al. [2018], Boomsma and Frellsen [2017], but these do not describe their training/evaluation data or splitting criteria. In contrast, we restrict our training and evaluation to a balanced subset, downsampled to the frequency of the rarest class, which limits performance slightly. Additionally, to enable fair comparison over three replicates between 3DCNN and GNN, we then trained on only half of these down-sampled environments. The discrepancy in performance we observe on this subset is indicative of the fact that the differences in residue environment are subtle and complex, so simply increasing training data can result in higher performance. This is especially true for common classes such as leucine and glycine, which are over five times as frequent than the least common class, cysteine. Within these common classes, accuracy exceeds 80%, which increases the average accuracy when classes are imbalanced.

G.4 LEP

Because this was a novel dataset, we computed initial results a non-deep learning method, Schrödinger's Glide, to score each protein-ligand complex. Glide is state-of-the-art for scoring protein-ligand complexes and determining how "good" a pose is. This resulted in 2 scores per ligand; the score to the inactive protein structure and the score to the active protein structure. We then performed a binary classification by training an SVM on these two features to predict the ligand activity class. This approach is reasonable from a physical basis: if the ligand binds much better to the active protein structure than the inactive protein structure, then it will be an activator of the protein's function.

G.5 LBA

Many methods have been developed for the prediction of ligand binding affinity using the PDBBind dataset. However, the standard has been to evaluate performance on the so-called "core set", as described in Section 3, after training and validating on the remainder of the refined set. The state-of-the-art reported on this core set has been achieved by the 3DCNN-based KDEEP Jiménez et al. [2018], followed closely by the popular random forest–based method RF-score Ballester and Mitchell [2010]. However, because the core set contains only proteins that are also present in the training set, this only measures *in-distribution* performance, not generalizable scoring ability. Thus, the most comparable baseline for our dataset, which was split at 30% sequence identity, is the performance of the empirical linear regression–based scoring function X-score fitted to complexes with less than 30% identity to the core set, as reported in Li and Yang [2017]. We note that this is not a perfect comparison, since the procedure used in Li and Yang [2017] reduces the size of the training set significantly; however, as an empirical scoring function the performance of X-score is not very sensitive to training set size, compared to RF-score, which was significantly affected.

H Supplementary Tables

Table 7: Comparison of performance against state-of-the-art methods, where available. The 3DCNN, GNN, and ENN networks achieve state-of-the-art in several tasks; for those where they do not (SMP, PIP, LBA), we note that the competing methods also use the 3D geometry of molecules. Asterisks (*) indicate that the exact training data differed (though splitting criteria were the same).

Task	Metric	3D			SOTA	
		3DCNN	GNN	ENN		
SMP	μ [D]	0.754	0.501	0.052	***0.030**	[Gilmer et al., 2017]
	$\varepsilon_{\mathrm{gap}}$ [eV]	0.580	0.137	0.095	***0.061**	[Anderson et al., 2019]
	U_0^{at} [eV]	3.862	1.424	0.025	***0.014**	[Schütt et al., 2017]
PIP	AUROC	0.844	*0.669	—	**0.919**	[Sanchez-Garcia et al., 2018]
RES	accuracy	**0.451**	0.082	*0.072	*0.425	[Torng and Altman, 2017]
MSP	AUROC	0.574	**0.609**	0.574	—	
LBA	RMSE	**1.416**	1.601	1.568	*1.838	[Li and Yang, 2017]
	glob. R_P	0.550	0.545	0.389	***0.645**	[Li and Yang, 2017]
	glob. R_S	0.553	0.533	0.408	***0.697**	[Li and Yang, 2017]
LEP	AUROC	0.589	0.681	0.663	**0.770**	[Friesner et al., 2004]
PSR	mean R_S	**0.431**	0.411	—	**0.432**	[Pagès et al., 2019]
	glob. R_S	**0.789**	0.750	—	**0.796**	[Pagès et al., 2019]
RSR	mean R_S	**0.264**	**0.234**	—	0.173	[Alford et al., 2017]
	glob. R_S	0.372	**0.512**	—	0.304	[Alford et al., 2017]

Table 8: Complete benchmarking results from Tables 3–6, with additional metrics and standard deviations reported over three replicates. R_K is Kendall correlation and AUPRC is area under the precision-recall curve. SMP metrics are all mean absolute error (MAE). Asterisks (*) indicate that the exact training data differed (though splitting criteria were the same).

Task	Metric	3DCNN	GNN	ENN	SOTA Baseline [Pagès et al., 2019]
PSR	mean R_P	**0.557 ± 0.011**	0.500 ± 0.012	—	0.444
	mean R_K	**0.308 ± 0.010**	0.289 ± 0.005	—	0.304
	mean R_S	**0.431 ± 0.013**	0.411 ± 0.006	—	**0.432**
	global R_P	**0.780 ± 0.016**	0.747 ± 0.018	—	0.772
	global R_K	**0.592 ± 0.016**	0.547 ± 0.016	—	**0.594**
	global R_S	**0.789 ± 0.017**	0.750 ± 0.018	—	**0.796**
					SOTA Baseline [Alford et al., 2017]
RSR	mean R_P	**0.286 ± 0.038**	**0.275 ± 0.007**	—	0.129
	mean R_K	**0.181 ± 0.032**	**0.157 ± 0.004**	—	0.119
	mean R_S	**0.264 ± 0.046**	**0.234 ± 0.006**	—	0.173
	global R_P	0.360 ± 0.030	**0.519 ± 0.051**	—	0.161
	global R_K	0.247 ± 0.017	**0.348 ± 0.038**	—	0.206
	global R_S	0.372 ± 0.027	**0.512 ± 0.049**	—	0.304
					Non-3D Baseline [Sanchez-Garcia et al., 2018]
PIP	AUROC	**0.844 ± 0.002**	*0.669 ± 0.001	—	0.841
					Non-3D Baseline [Rao et al., 2019]
RES	accuracy	**0.451 ± 0.002**	0.082 ± 0.002	*0.072 ± 0.005	*0.30
MSP	AUROC	0.574 ± 0.005	**0.609 ± 0.011**	0.574 ± 0.040	0.554
	AUPRC	0.187 ± 0.007	0.176 ± 0.003	**0.196 ± 0.010**	—
					Non-3D Baseline [Tsubaki et al., 2019]
SMP	μ [D]	0.754 ± 0.009	0.501 ± 0.002	**0.052 ± 0.007**	0.496 ± 0.002
	α [bohr^3]	3.045 ± 1.128	1.562 ± 0.038	**0.127 ± 0.026**	0.392 ± 0.004
	$\varepsilon_{\mathrm{HOMO}}$ [eV]	0.303 ± 0.000	0.092 ± 0.002	**0.044 ± 0.010**	0.107 ± 0.001
	$\varepsilon_{\mathrm{LUMO}}$ [eV]	0.517 ± 0.011	0.096 ± 0.001	**0.035 ± 0.003**	0.115 ± 0.001
	$\varepsilon_{\mathrm{gap}}$ [eV]	0.580 ± 0.004	0.137 ± 0.002	**0.095 ± 0.021**	0.154 ± 0.001
	R^2 [bohr^2]	64.514 ± 1.524	89.912 ± 32.591	**1.045 ± 0.065**	27.976 ± 0.212
	ZPVE [meV]	88.219 ± 16.287	33.504 ± 5.548	**1.705 ± 0.044**	10.614 ± 0.270
	U_0^{at} [eV]	3.862 ± 0.594	1.424 ± 0.211	**0.025 ± 0.001**	0.182 ± 0.004
	U_{298}^{at} [eV]	4.356 ± 0.498	1.227 ± 0.610	**0.025 ± 0.001**	0.181 ± 0.001
	H_{298}^{at} [eV]	4.088 ± 0.229	0.927 ± 0.177	**0.024 ± 0.001**	0.180 ± 0.004
	G_{298}^{at} [eV]	4.369 ± 0.805	1.171 ± 0.497	**0.024 ± 0.001**	0.173 ± 0.000
	C_v $[\frac{\mathrm{cal}}{\mathrm{molK}}]$	1.418 ± 0.200	0.350 ± 0.078	**0.034 ± 0.001**	0.187 ± 0.004
					Non-3D Baseline [Öztürk et al., 2018]
LBA (30%)	RMSE	**1.416 ± 0.021**	1.601 ± 0.048	1.568 ± 0.012	1.565 ± 0.018
	global R_P	0.550 ± 0.021	0.545 ± 0.027	0.389 ± 0.024	**0.573 ± 0.022**
	global R_S	0.553 ± 0.009	0.533 ± 0.033	0.408 ± 0.021	**0.574 ± 0.010**
LBA (60%)	RMSE	1.621 ± 0.025	**1.408 ± 0.069**	1.620 ± 0.049	1.760 ± 0.415
	global R_P	0.608 ± 0.020	**0.743 ± 0.022**	0.623 ± 0.015	0.713 ± 0.013
	global R_S	0.615 ± 0.028	**0.743 ± 0.027**	0.633 ± 0.021	0.702 ± 0.013
LEP	AUROC	0.589 ± 0.020	**0.681 ± 0.062**	**0.663 ± 0.100**	**0.696 ± 0.021**
	AUPRC	0.483 ± 0.037	**0.598 ± 0.135**	**0.551 ± 0.121**	**0.550 ± 0.024**

Table 9: Number of samples for each dataset presented in the benchmark.

Task	Number of Samples		
	Train	Val	Test
SMP	103547	12943	12943
PIP	87303	31050	15268
RES	3820837	192371	648372
MSP	2864	937	347
LBA	3563	448	452
LEP	304	110	104
PSR	25400	2800	16099
RSR	12479	4000	4000