Electronic Supplementary Information
DeepFrag: A Deep Convolutional Neural Network for Fragment-based Lead Optimization

Harrison Green[1], David R. Koes[2], Jacob D. Durrant[1,*]

[1] Department of Biological Sciences, University of Pittsburgh, Pittsburgh, Pennsylvania, 15260, United States
[2] Department of Computational and Systems Biology, University of Pittsburgh, Pittsburgh, Pennsylvania, 15260, United States

[*] To whom correspondence should be addressed. Email: durrantj@pitt.edu

1 Hyperparameter Tuning

We systematically explored a number of hyperparameter combinations to identify one well-suited for fragment prediction. We found that the choice of hyperparameters can substantially impact both the speed of grid generation and the quality of information ultimately provided to the machine-learning model.

Due to the number of parameters, testing all combinations was not practical. We therefore selected an initial parameter set and tested whether deviations from that set might improve performance. For the initial set, we considered only the atomic elements of receptor (N, O, C, S, other) and parent (N, O, C, other) heavy atoms (ignoring hydrogen atoms), without regard for atomic hybridizations (i.e., the *simple* grid-layer definition applied to both the receptor and parent; see Section 1.2.1).

We mapped the positions of these heavy atoms onto cubic voxel grids. The grid width, w, is defined as the number of grid points in each dimension, such that there are w^3 points total. The spacing in Ångstroms between the grid points, s, is called the grid resolution. The physical length of the grid is thus ws along each dimension, and the volume is w^3s^3. We initially considered grids with 24 x 24 x 24 points, spaced 1.0 Å apart along the x, y, and z axes.

Finally, to calculate each atom's contribution (density) at each grid point, we initially used the *SMOOTH* voxelation method. Voxelation methods determine the *shape* and falloff of the density at each atom. *SMOOTH* specifically uses a continuous, piecewise function, as in ref. [1]:

$$A(d,r) = \begin{cases} e^{-2d^2/r^2} & 0 \leq d < r \\ \frac{4}{e^2r^2}d^2 - \frac{12}{e^2r}d + \frac{9}{e^2} & r \leq d < 1.5r \\ 0 & d \geq 1.5r \end{cases}$$

where d is the distance from the grid point to the atom center and r is the atom-influence radius, initially set to 2.0 Å (see Section 1.2.2). The radius thus controls the *size* of each atom's density (i.e., how close an atom must be to a grid point to influence the density recorded at that point). Where more than one atom of the same type contributed to the same voxel point, the corresponding values were summed.

1.1 Early-Stage Tuning

Our early-stage efforts focused on identifying model and grid-density parameters well suited to the fragment-selection task. Because this phase of hyperparameter tuning required us to generate many models for comparison, we considered only the receptor/ligand complexes present in the PDBBind database [2, 3]. We note that the ligand SMILES strings in the PDBBind database are somewhat inconsistent (e.g., carboxylate moieties are inconsistently protonated), so that some fragment labels were duplicated in this initial dataset. This duplication led to overall lower accuracy. Fortunately, for the purpose of hyperparameter tuning we were concerned only with the relative performance between models. To further reduce the size of the dataset, we also considered only (*receptor/parent*, *fragment*) tuples with connection points that came within 3 Å of any receptor atom.

Table S1: Hyperparameters varied during our tuning protocol

Phase 1: Model parameters	
Learning rate	0.01, 0.001, **(0.0001)**
Batch size	**(16)**, 32, 64, 128
Blocks	[32,64], [16,32], [64,128], **([64,64])**
Fully Connected (fc)	[256], **([512])**, [1024], [2048], [256,256], [512,512], [1024,1024]
Grid width	16, **(24)**, 32
Grid resolution	0.5 Å, 1.0 Å, 2.0 Å
Phase 2: Grid-density parameters	
Grid resolution	0.25 Å to 3.00 Å **(0.75 Å)**
Atom-influence radius	0.25 Å to 4.00 Å
Phase 3: Grid-layer parameters	
Parent types	flat, flat-h, **(simple)**, simple-h
Receptor types	flat, flat-h, **(simple)**, simple-h, meta, meta-mix
Phase 4: Voxelation parameters	
Voxelation method	*SPHERE*, *CUBE*, *POINT*, *SMOOTH*, ***(SMOOTH-2)***, *LJ*
Atom-influence radius	1.0 Å, **(1.75 Å)**, 2.5 Å

Parentheses indicate optimal values that were fixed for later phases.

We divided these tuples into *TRAIN*, *TEST*, and *VAL* sets (60/20/20), using the approach described in the main text. We trained on examples in the *TRAIN* set and evaluated TOP-k accuracy on examples in the *VAL* set. We merged the fragment vectors from the *TRAIN* and *VAL* sets into a combined label set that we used for fragment selection. We trained each model variant for 8 hours on a GPU, achieving accuracy roughly 80-90% of the eventual maximum. During early experiments, we found that this shortened training cycle was sufficient to assess relative validation accuracy between model variants. That is, in general if a model performed better after 8 hours, it also performed better after full convergence.

1.1.1 Phase 1: Model Parameters

In the first phase of early-stage hyperparameter tuning, we randomly sampled 32 combinations of learning rates, batch sizes, model architecture parameters (blocks and fc), grid widths, and grid resolutions (Table S1), where "blocks" describes the number of filters in each 3-part convolution block, and "fc" describes the number and sizes of fully connected layers between the "Flatten" layer and the fragment output.

The learning rate, batch size, and model-architecture parameters impact training and information processing. Based on our hyperparameter search, we ultimately selected a learning rate of 0.0001, a batch size of 16, and a model architecture with two convolutional blocks of size 64 and a single fully connected layer of size 512.

The grid width and resolution parameters used to generate the voxel grids impact the speed of grid generation and the amount of information provided to the network. The grid resolution (i.e., the distance between adjacent grid points) is particularly impactful. Low-resolution grids can be generated quickly, but high-resolution grids provide more information. Based on our hyperparameter search, we selected a grid width of 24 (i.e., a cubic grid comprised of 24 x 24 x 24 points). However, this first-phase search did not unambiguously identify a best grid resolution.

1.1.2 Phase 2: Grid-Density Parameters

In the second phase of hyperparameter tuning, we fixed the best learning rate, batch size, model architecture, and grid width parameters found during the first phase. We then randomly sampled 32 different combinations of grid resolutions and atom-influence radii (Table S1). We continued to tune the grid resolution in the second phase because the first phase did not reveal an optimal value. Based on our hyperparameter search,

we selected a grid resolution of 0.75 Å (Table S1). The best atom-influence radius was less clear, though it seemed to lie between 1.0 and 2.0 Å. We tentatively fixed it at 1.0 Å.

1.2 Late-Stage Tuning

Our subsequent efforts to identify suitable voxelation and grid-layer parameters leveraged a dataset derived from the larger Binding MOAD database [4] rather than the PDBBind [2, 3]. The data was again split into *TRAIN*, *TEST*, and *VAL* sets (roughly 60/20/20), as described in the main text. We continued to train on examples in the *TRAIN* set and evaluated TOP-k accuracy on examples in the *VAL* set. We merged the fragment vectors from the *TRAIN* and *VAL* sets into a combined label set, which we used for fragment selection. We trained each model variant for 15 epochs (approximately 20 hours) on a GPU, again achieving accuracy roughly 80-90% of the eventual maximum.

1.2.1 Phase 3: Grid-Layer Parameters

In the third phase of hyperparameter tuning, we explored the effect of varying the grid-layer definitions of the parent and receptor atoms. In the same way a 2D image has three color channels (e.g., red, green, blue), our 3D grids have N atom (i.e., grid-layer) channels (e.g., carbon, nitrogen, oxygen, etc.). It is tempting to construct a large array of features (e.g., to create separate layers for aliphatic and aromatic carbon atoms) in order to maximize information. But using too many layers can slow training and make the model harder to deploy because each feature must be separately computed during inference. On the other hand, too few layers may result in information loss and worse performance.

We evaluated several "atom-channel" (i.e., grid-layer) definitions. In all cases, atoms from the receptor and parent always contributed to different grid layers, allowing our model to distinguish between the two. In the *flat* definition, we assigned all atoms to a single channel. This method serves as a baseline comparison and is analogous to converting an image to grayscale before training an image classifier. The *flat-h* definition includes hydrogen atoms, but *flat* does not.

In the *simple* definition (also described above), we assigned atoms to separate layers based on atomic number. We assigned the most common atoms to individual layers and aggregated all other atoms in a separate "other" layer. For ligands, the most common atoms are carbon, nitrogen, and oxygen. For receptors, we also include a separate sulfur layer. The *simple-h* definition further includes a hydrogen layer.

We also experimented with high-level descriptors for receptor atoms. In the *meta* definition, we assigned each atom to one or more channels based on the following properties: aromatic, hydrogen-bond donor, hydrogen-bond acceptor, partial positive charge, partial negative charge, and occupancy (any atom). For example, an aromatic carbon atom with a partial positive charge would be present in the aromatic, partial positive, and occupancy layers. In the *meta-mix* description we combine the *meta* layers and the *simple-h* layers.

We trained nine model variants in triplicate and report the average validation accuracy after 15 epochs (Table S3). When the parent-atom definition was varied, the receptor-atom definition was fixed as *simple*, and vice versa. We obtained the best performance when the *simple* grid-layer definition was applied to both the parent and receptor. It is interesting that this definition was optimal, given that it ignores both hydrogen atoms and atomic hybridizations. Other works have found that element-type channels are sufficient to achieve high performance on the related binding-affinity prediction task [1]. Similarly, including hydrogen atoms appears to have little impact on affinity-prediction performance [5].

1.2.2 Phase 4: Voxelation Parameters

Our grid representation requires that each atom contribute "density" to neighboring grid points. In the fourth phase of hyperparameter tuning (Table S1), we fixed the parent and receptor typing schemes as *simple* and used a comprehensive grid search to vary the voxelation method (Table S4). We also continued to tune the atom-influence radius because previous efforts had identified no optimal value. We considered the following voxelation methods (Figure S1):

1. SPHERE: Set all grid points within r to 1 (spherical boolean).
2. CUBE: Set all grid points within r along the x, y, or z axis to 1 (cubical boolean).

Table S2: Grid-layer (atom typing) descriptions

Parent Typing	Layers
flat	single channel, no hydrogen
flat-h	single channel, including hydrogen
simple	[C, N, O, (other)]
simple-h	[H, C, N, O, (other)]
Receptor Typing	**Layers**
flat	single channel, no hydrogen
flat-h	single channel, including hydrogen
simple	[C, N, O, S, (other)]
simple-h	[H, C, N, O, S, (other)]
meta	[aromatic, H-don, H-acc, pos, neg, occupancy]
meta-mix	*meta* + *simple-h*

Note that the *simple* and *simple-h* variants for the receptor include a separate sulfur layer. Otherwise, the parent and receptor typing schemes are identical. Additionally, the *meta* and *meta-mix* descriptions are used only for the receptor.

Table S3: Effect of varying grid layer descriptions on TOP-1 accuracy

Parent Typing	Receptor Typing	TOP-1 Accuracy
flat	simple	45.87 (± 0.44)
flat-h	simple	46.00 (± 0.55)
simple	simple	50.57 (± 0.25)
simple-h	simple	50.45 (± 0.06)
Parent Typing	**Receptor Typing**	**TOP-1 Accuracy**
simple	flat	46.20 (± 0.51)
simple	flat-h	47.27 (± 0.83)
simple	simple	50.57 (± 0.25)
simple	simple-h	50.10 (± 0.58)
simple	meta	49.22 (± 0.58)
simple	meta-mix	50.10 (± 0.87)

For each parent grid-layer variant, the receptor grid-layer definition was fixed at *simple*, and vice-versa. Each model variant was trained in triplicate on the *TRAIN* dataset for 15 epochs. We report TOP-1 accuracy (%) and standard error on the *VAL* set.

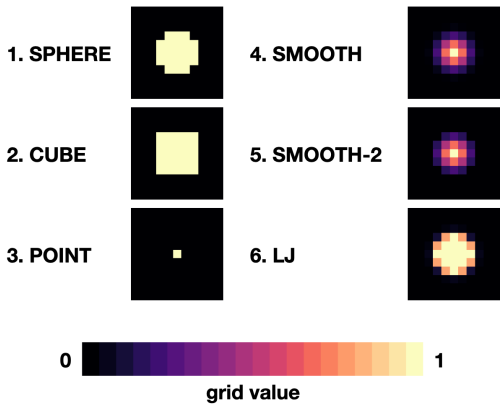

Figure S1: Illustration of different grid-voxelation methods in 2D.

3. POINT: Set only the nearest grid point to 1 (as in ref. [6]).

4. SMOOTH: Set nearby grid points per a continuous, piecewise function (as in ref. [1]), defined as:

$$A(d,r) = \begin{cases} e^{-2d^2/r^2} & 0 \le d < r \\ \frac{4}{e^2 r^2} d^2 - \frac{12}{e^2 r} d + \frac{9}{e^2} & r \le d < 1.5r \\ 0 & d \ge 1.5r \end{cases}$$

   where d is the distance from the grid point to the atom center. Note that in the present work, r is the same for all atom types and so is not equivalent to an atomic radius.

5. SMOOTH-2: Set nearby grid points per the exponential part of SMOOTH:

$$A(d,r) = \begin{cases} e^{-2d^2/r^2} & 0 \le d < r \\ 0 & d \ge r \end{cases}$$

6. LJ: Set grid points per the repulsive component of a Lennard-Jones potential (as in ref. [7]), defined as:

$$A(d,r) = \begin{cases} 1 - e^{-(r/d)^{12}} & 0 \le d < 5 \\ 0 & d \ge 5 \end{cases}$$

Each of these approaches has its advantages and disadvantages. For example, simple boolean voxelation is fast to compute but may oversimplify the input representation, while more complex functions can retain high-resolution distance information at the cost of slower grid generation. Surprisingly, the specific method had little effect on overall prediction accuracy (Table S4), with the exception of the *CUBE* and *POINT* methods, which apparently fail to provide sufficient information to the model. We hypothesize that the *CUBE* method does not preserve atomic symmetries (especially as the atom-influence radius increases), and the *POINT* method is too sparse. As others have noted [1], the hard boolean cutoff in *SPHERE* and the smooth Gaussian in *SMOOTH* are remarkably competitive, even though the former does not preserve atomic-distance information.

We ultimately settled on the *SMOOTH-2* voxelation method (1.75 Å radius) because it is the fastest to compute of the distance-preserving spherical methods. However, our hyperparameter search did not reveal a clearly optimal voxelation method and atom-influence radius; indeed, different methods/radii produce similar results.

Table S4: Effect of voxelation method on TOP-1 accuracy (%)

	1.0 Å	1.75 Å	2.5 Å	Overall
1. SPHERE	49.12 (± 0.48)	49.65 (± 0.38)	48.50 (± 0.43)	49.09 (± 0.30)
2. CUBE	49.27 (± 0.55)	42.12 (± 0.13)	39.18 (± 0.13)	44.06 (± 1.41)
3. POINT	44.13 (± 0.18)	43.27 (± 0.48)	43.40 (± 0.56)	43.64 (± 0.28)
4. SMOOTH	50.38 (± 0.08)	49.00 (± 0.06)	46.93 (± 0.70)	48.74 (± 0.56)
5. SMOOTH-2	51.13 (± 0.61)	49.40 (± 0.61)	46.23 (± 0.74)	48.92 (± 0.78)
6. LJ	50.03 (± 0.42)	48.68 (± 0.88)	47.32 (± 0.30)	48.68 (± 0.50)
Overall	49.01 (± 0.58)	47.13 (± 0.76)	45.62 (± 0.71)	

Each model variant was trained three times on the *TRAIN* dataset for 15 epochs. We report TOP-1 accuracy (%) and standard error on the *VAL* set.

Table S5: Effect of multiple-rotation sampling on accuracy

N	TOP-1	TOP-8	TOP-64
1	56.41	64.79	70.78
2	57.17	65.47	71.36
4	57.57	65.87	71.78
8	57.72	66.03	71.98
16	57.83	66.06	72.05
32	57.91	66.17	72.13

Each entry represents the TOP-k *VAL*-set accuracy obtained by averaging N-rotations per sample (fully converged model, roughly 50 epochs of training).

1.3 Final Hyperparameters Used to Train to Convergence

For the final DeepFrag model, we ultimately selected a learning rate of 0.0001, a batch size of 16, and a model architecture with two convolutional blocks of size 64 and a single fully connected layer of size 512 (Figure S2). To convert receptor and parent structures to voxel grids, we considered only the atomic elements of receptor (N, O, C, S, other) and parent (N, O, C, other) heavy atoms, without regard for atomic hybridizations (i.e., the *simple* definition for both the receptor and parent, see Section 1.2.1). We ignored hydrogen atoms.

Atomic positions were mapped onto cubic voxel grids with 24 x 24 x 24 points, spaced 0.75 Å apart along the x, y, and z axes. The influence of each atom at each point was calculated using the *SMOOTH-2* voxelation method (1.75 Å radius).

We found that averaging the fingerprint predictions of multiple randomly rotated grids further improved accuracy. In Table S5 we report the TOP-k test accuracy on the *VAL* set obtained by averaging N-rotations per sample for different values of N. By default, DeepFrag averages the output of 32 such rotations.

2 Converting DeepFrag Output to 3D Molecular Models

As output, DeepFrag produces an RDKFingerprint-like vector representing the candidate fragment, where each vector component is a continuous number in (0, 1). This fingerprint is then compared to a library (label set) of known fingerprints to identify corresponding SMILES strings. But SMILES strings do not describe 3D atomic coordinates, as required for subsequent visualization and *smina* rescoring (see below) [8].

To address this issue, we created 3D molecular models of DeepFrag-optimized compounds, comprised of the parent ligand and suggested fragment merged into one molecule. DeepFrag requires the 3D coordinates of parent-ligand atoms as input, so we retained parent coordinates. We then used RDKit [9] to (1) generate 3D coordinates of DeepFrag-suggested fragments, (2) position each 3D fragment at the appropriate location relative to the corresponding parent ligand, and (3) add a chemical bond between the atoms at the parent and fragment connection points.

DeepFrag assumes that the pose of the parent ligand is correct relative to the receptor, but RDKit assigns atomic coordinates to fragment atoms without regard for receptor atoms. Indeed, the fragment may extend into the receptor itself in some cases. To address this issue, we used RDKit to systematically rotate the

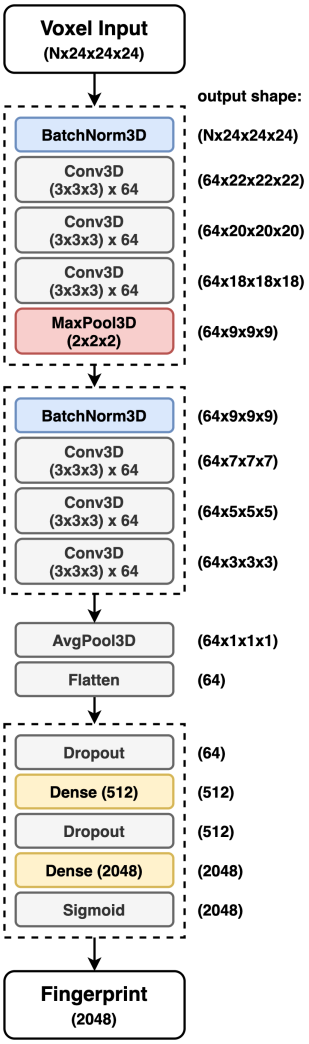

Figure S2: Final DeepFrag convolutional neural network architecture. The input tensor consists of concatenated atom-wise channels (grids) from the parent and receptor (N = total number of atom channels). The output tensor contains a raw fragment-fingerprint prediction. For this model, blocks = [64,64] and fc = [512].

modeled fragment around the dihedral angle of the new parent-fragment bond in 5° increments. At each angle, we calculated the energy relative to the respective crystallographic receptor using the UFF force field [10]. The positioned (posed) moiety conformation with the lowest calculated energy was retained.

3 Reevaluating DeepFrag-Optimized Compounds with *smina*

We used the docking program *smina* to provide independent validation of select DeepFrag-generated compounds. To further prepare the small-molecule models, we used manual inspection, Open Babel [11], and Avogadro [12] to (1) reassign the hydrogen atoms as appropriate for physiological pH (e.g., deprotonate the carboxylate moieties, protonate amines in some cases, etc.) and (2) correct occasional compounds with inappropriate geometries. To further prepare protein-receptor models, we used PDB2PQR [13, 14] to add hydrogen atoms as appropriate for physiological pH and to optimize the hydrogen-bond network. We converted the small-molecule and protein-receptor files to the *smina*-compatible PDBQT format using MGLTools [15].

We next used *smina* to score the 3D poses. The *--minimize* option allowed *smina* to optimize the geometry

of the crystallographic (original) and DeepFrag-modified molecules within the receptor binding site, without redocking. We used *--minimize* because, although our original efforts to position the suggested moieties using RDKit effectively prevented major steric clashes (Subsection 2), RDKit is not designed for fragment docking. Furthermore, in our RDKit protocol the positions of the parent atoms were fixed, but in reality a ligand must sometimes shift within a binding site to accommodate a new fragment addition.

4 DeepFrag Applied to a Crystallographic Fragment Screen

To build on the XChem crystallographic screen targeting the SARS-CoV-2 main protease (M^{Pro}) [16], we used DeepFrag (default settings) to identify 132 candidate moieties that could replace various hydrogen atoms belonging to any of 18 crystallographic ligands. We then followed the protocols described in Sections 2 and 3 to position and score 3D models of the crystallographic and DeepFrag-optimized compounds. For one of the 18 crystallographic ligands, the root-mean-square deviation (RMSD) was greater than 1.5 Å between the pre- and post-minimization poses. We discarded this compound and all its DeepFrag derivatives because our goal was to assess how DeepFrag additions alone might impact docking scores, and substantial shifts in ligand poses could introduce new confounding variables. Of the remaining 120 DeepFrag compounds, 23 similarly shifted more than 1.5 Å and so were also discarded.

We also further considered the positions of the heavy (non-hydrogen) fragment and receptor atoms. We realized that in some cases we had asked DeepFrag to replace hydrogen atoms that were pointed away from the protein receptor (e.g., towards bulk solvent), such that any added fragment was unlikely to interact with the receptor itself. Of the 97 remaining DeepFrag compounds, 24 had posed fragments that came within 3 Å of the receptor. These 24 candidate ligands were derived from 13 of the crystallographic ligands. The final compound set described in the main text is composed of these compounds.

References

(1) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. *Protein-Ligand Scoring with Convolutional Neural Networks*; tech. rep.; 2016.

(2) Wang, R.; Fang, X.; Lu, Y.; Yang, C. Y.; Wang, S. *J Med Chem* **2005**, *48*, 4111–4119.

(3) Wang, R.; Fang, X.; Lu, Y.; Wang, S. *J Med Chem* **2004**, *47*, 2977–2980.

(4) Hu, L.; Benson, M. L.; Smith, R. D.; Lerner, M. G.; Carlson, H. A. *Proteins* **2005**, *60*, 333–340.

(5) Ragoza, M.; Hochuli, J.; Idrobo, E.; Sunseri, J.; Koes, D. R. *J Chem Inf Model* **2017**, *57*, 942–957.

(6) Stepniewska-Dziubinska, M. M.; Zielenkiewicz, P.; Siedlecki, P. *Bioinformatics* **2018**, *34*, 3666–3674.

(7) Jiménez, J.; Doerr, S.; Martínez-Rosell, G.; Rose, A. S.; De Fabritiis, G. *Bioinformatics* **2017**, *33*, 3036–3042.

(8) Koes, D. R.; Baumgartner, M. P.; Camacho, C. J. *J Chem Inf Model* **2013**, *53*, 1893–904.

(9) Landrum, G. RDKit: open-source cheminformatics, Web Page.

(10) Rappé, A. K.; Casewit, C. J.; Colwell, K. S.; Goddard Iii, W. A.; Skiff, W. M. *Journal of the American chemical society* **1992**, *114*, 10024–10035.

(11) O'Boyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. *J. Cheminf.* **2011**, *3*, 33.

(12) Hanwell, M. D.; Curtis, D. E.; Lonie, D. C.; Vandermeersch, T.; Zurek, E.; Hutchison, G. R. *J Cheminform* **2012**, *4*, 17.

(13) Dolinsky, T. J.; Czodrowski, P.; Li, H.; Nielsen, J. E.; Jensen, J. H.; Klebe, G.; Baker, N. A. *Nucleic Acids Research* **2007**, *35*, W522–W525.

(14) Dolinsky, T. J.; Nielsen, J. E.; McCammon, J. A.; Baker, N. A. *Nucleic Acids Research* **2004**, *32*, W665–W667.

(15) Morris, G. M.; Ruth, H.; Lindstrom, W.; Sanner, M. F.; Belew, R. K.; Goodsell, D. S.; Olson, A. J. *Journal of Computational Chemistry* **2009**, DOI: 10.1002/jcc.21256.

(16) Douangamath, A. et al. *Nat Commun* **2020**, *11*, 5047.