ClassicalGSG: Prediction of logP Using Classical Molecular Force Fields and Geometric Scattering for Graphs

Nazanin Donyapour[*] Matthew J. Hirn[*†‡] Alex Dickson[*§¶]

November 18, 2020

Abstract

This work examines methods for predicting the partition coefficient ($\log P$) for a dataset of small molecules. Here, we use atomic attributes such as radius and partial charge, which are typically used as force field parameters in classical molecular dynamics simulations. These atomic features are transformed into index-invariant molecular features using a recently developed method called Geometric Scattering for Graphs (GSG). We call this approach "ClassicalGSG" and examine its performance under a broad range of conditions and hyperparameters. We train a ClassicalGSG $\log P$ predictor with neural networks using $10,722$ molecules from the ChEMBL21 dataset and apply it to predict the $\log P$ values from four independent test sets. The ClassicalGSG method's performance is compared to a baseline model that employs graph convolutional neural networks (GCNNs). Our results show that the best prediction accuracies are obtained using atomic attributes generated with the CHARMM generalized Force Field (CGenFF) and 2D molecular structures.

Keywords: $\log P$ prediction, Partition coefficients, Geometric scattering for graphs, Graph convolutional neural network ■

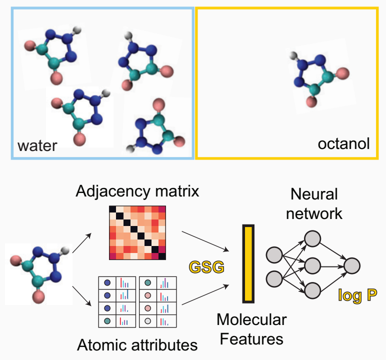
[*]Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, Michigan, USA

[†]Department of Mathematics, Michigan State University, East Lansing, Michigan, USA

[‡]Center for Quantum Computing, Science and Engineering, Michigan State University, East Lansing, Michigan, USA

[§]Department of Biochemistry and Molecular Biology, Michigan State University, East Lansing, Michigan, USA

[¶]Corresponding author: alexrd@msu.edu



The ClassicalGSG method utilizes neural networks (NNs) to predict the $\log P$ values from molecular features generated using a method called Geometric Scattering for Graphs (GSG). The GSG method creates molecular features from the graph representation of the molecule, where each atom has a set of atomic attributes. These atomic attributes are typically used as force field parameters in classical molecular dynamics simulations and include partial charges, Lennard-Jones well depth, Lennard-Jones radius, and atomic type.

1 INTRODUCTION

The partition coefficient (P) measures the relative solubility of a compound between two solvents. It is defined as the ratio of the concentration of an uncharged compound dissolved in an organic solvent (e.g. octanol) in comparison to water. The logarithm of this ratio ($\log P$) is considered to be one of the main factors in determining the drug-likeness of a chemical compound. The $\log P$ determines the lipophilicity, which affects bioavailability, solubility, and membrane permeability of a drug compound in the body. Moreover, an orally active drug should have a $\log P$ value of less than 5 according to one of the criteria of the famous Lipinski's Rule of Five.[1] Thus, predicting $\log P$ plays an essential role in the drug discovery process and is our main focus in this study. The prediction of $\log P$ is also used as a stepping stone to calculate other molecular properties such as the distribution coefficient ($\log D$),[2] drug solubility ($\log S$),[3,4] and Lipophilic efficiency (LiPE).[5] All of these properties have been used in drug discovery and design: $\log D$ is an effective descriptor for lipophilicity of an ionized compound,[6] LiPE combines the potency and lipophilicity of a drug compound to estimate its quality, and $\log S$ is important to model the solubility of a compound in the human body.

The partition coefficient is widely used in cheminformatics and generally there are diverse experimental methods to measure it.[7] However, these methods are time-consuming and expensive for large databases of compounds and not feasible for compounds that are not synthesized.[8] Therefore, a large number of computational methods have been developed to predict accurate values of $\log P$. These methods have a long history and can be classified by both their input features (atomic/fragment, molecular and hybrid) and their models or algorithms (parameteric models vs. machine learning methods) (Table 1).

Table 1: $\log P$ prediction models classifications

		Mathematical models	
		Parametric Models	Machine Learning Methods
Features	Atomic/Fragment	XlogP3,[9] AlogP,[10] ClogP,[11] KowWIN,[12] JPlogP[13]	László et al,[14] Huuskonen et al[15]
	Molecular	MlogP,[16] iLogP,[17] Manhold[18]	AlogPS,[19,20] S+logP,[21] CSLogP
	Hybrid	Silicos-IT LogP[22]	TopP-S,[23] OpenChem[24]

In atomic-based or fragment-based methods, which is based on atomic or fragment contributions, a set of atomic or fragment based descriptors is used as input to the model, while "molecular" methods use descriptors of the whole molecule, such as topology or molecular fingerprints.[16,17,19] "Hybrid" models combine atomic and molecular descriptors as input to the model.[23,24] In general, there are challenges with both atomic and molecular descriptors. In Atomic-based or fragment-based methods, the accuracy of the atomic contributions depends on the similarity of the atomic environments of the atoms in the group. Unfortunately, more training data is need as the number of groups grows larger. Fragment-based methods can struggle with defining the optimal size of fragments that participate in predictions. On the other hand, the accuracy of property-based methods heavily depends on the choice of molecular descriptors. Common descriptors include: 3D molecular structure,[17] molecular volume and surface area[25], solvation free energies,[25] number of carbon atoms and heteroatoms.[16] Furthermore, these molecular descriptors can be difficult or computationally costly to generate.

A second way of categorizing $\log P$ predictors is by the types of mathematical model used to process the input data. Parameteric models use methods such as least squares estimation or multiple linear regression to fit parameters that govern the relative contributions of the different input features. Machine learning based methods such as Support Vector Machines (SVM),[26–28] Neural Networks (NNs),[23,26,27,29] and Graph convolutional neural networks (GCNN)[24] have been used to predict logP values.

Recently, some methods have been described that create their own custom molecular features from atomic features, which go beyond simple additive models. The TopP-S[23] predictor was developed by Wu et al and uses the atomic positions to create topological descriptors called Betti barcodes. These Betti barcodes are used as molecular features that are input into deep neural networks, along with atomic features such as atom type. Results were shown to further improve upon the addition of 633 "molecular fingerprints" calculated from ChemoPy.[30–32] TopP-S has shown success in predicting $\log P$ over other methods like XlogP3, ClogP and KowWIN using a group of independent test sets.

Graph representations of molecules have also shown success in various applications including predicting molecular properties,[24,33–35] virtual screening[36] and molecular force field calculations.[37] In particular, OpenChem (`https://mariewelt.github.io/OpenChem/html/index.html`) uses a graph representation of the molecules. Each atom represents a node in a graph and has a vector of atomic features including element type, valence, charge, hybridization, and aromaticity; bonded atoms are connected by an edge in the graph. GCNNs are then trained on the graph representations created using these atomic features and the 2D structure – or, "graph structure" – of the molecules.

Graph representations are beneficial in that they are invariant to translation, rotation, and reflection symmetries. Another molecular symmetry that should be respected is invariance to the re-indexation of atoms: changing the order in which atoms are input to the model should not affect the computed molecular features. Summation operations respect re-indexation symmetry but it is not straightforward how to capture more detailed information about molecular structure while maintaining re-indexation symmetry. A recently-described method, Geometric Scattering for Graphs (GSG),[38] provides a solution to this problem. GSG, which is analogous to GCNNs, creates molecular features by scattering atomic features across a graph using lazy random walks. GSG is fast in creating re-indexation invariant features and also its feature vectors have the same length allowing us to easily measure the similarity of molecules, even those with different numbers of atoms. It has shown promising results in the classification of social network data and predicting Enzyme Commission (EC) numbers.[38]

Given this abundance of algorithms for creating molecular features, we seek to compare some different methods based on molecular graph representations and their ability to predict $\log P$. Here we use GSG in combination with a set of atomic descriptors that are generated for use with classical molecular dynamics force fields: partial charges, atom type, and Lennard-Jones interaction parameters. We call this method "ClassicalGSG" and examine its performance as a function of different atomic/molecular features. We compare the ClassicalGSG results with GCNNs trained on the same data and using a variety of atomic features, including those from previous work.[24] We then evaluate the performance of ClassicalGSG on several independent test sets and study the properties of features generated in the pipeline of GSGNN models. In addition, we investigate the properties of molecules with high $\log P$ prediction error. We conclude with a discussion about the GSG method generated features, the computational cost of generating atomic attributes with CGenFF and GAFF, and the relative performance of 2D versus 3D structure in predicting $\log P$ values.

2 METHODS

Datasets and generation of atomic attributes

The dataset used in this work is generated by Popova et al.[24] from the ChEMBL21 drug database (`https://www.ebi.ac.uk/chembl`). This dataset consists of 14176 molecules in SMILES format and their corresponding $\log P$ values.

The molecules are converted from SMILES format to mol2 format and their 3D structure is created by OpenBabel (`https://github.com/openbabel/openbabel`). Then CHARMM General Force Field (CGenFF)[39,40] CGenFF and General AMBER Force Field (GAFF)[41] parameter files are generated for each molecule. These force field parameter files are either created by CGenFF using the CGenFF tool of the SilcsBio software package (`http:`

`//silcsbio.com`) or by the Antichamber tool implemented in the Ambertools18[42] package. The process of generating CGenFF parameter files fails for 175 molecules, and GAFF for 681 molecules. These 774 molecules are removed from the ChEMBL21 dataset, resulting in a dataset of 13402 molecules. Then 80% of the molecules are used for training and the rest for testing. In addition, we evaluate our trained model on four independent test sets shown in Table 2. Star and NonStar[18] test sets are publicly available on `https://ochem.eu/article/17434` and the Husskonen[15] test set can be found on `https://ochem.eu/article/164`. The FDA dataset contains drug molecules that are approved by the Food and Drug Administration (FDA) of the United Sates and originally prepared by Chen et al.[9]

Table 2: Independent test sets used for evaluating ClassicalGSG models.

Test set name	Number of molecules
FDA[9]	406
Huuskonen[15]	348
Star[18]	223
NonStar[18]	43

As mentioned above, we use atomic attributes including partial charges, atom type, and Lennard-Jones interaction parameters. Below, we explain how we generate these atomic attributes.

Atomic partial charges for each atom are extracted from the parameter files generated by either the CGenFF[39,40] or GAFF[41] force field generator tools. To determine atom type classifications, we compared a number of different schemes. In one scheme, we classify atom types in one of five categories as shown in Table 3. This is referred to below as "AC5". Alternatively, we directly use the atom types as generated by either CGenFF or GAFF; referred to below as "ACall." In the third classification scheme, we manually sorted CGenFF atom types into 36 groups (AC36; Table S1) and GAFF atom types into 31 groups (AC31; Table S2) based on chemical knowledge. Specifically, efforts were made to make new groups for atom types with different elements and hybridization values and to separately identify atoms that are members of ring structures. Finally, a forth classification scheme simply uses a uniform atom type for all atoms, referred to as “AC1”.

Table 3: Classifying atoms in 5 categories (AC5).

Atom type	Category number
Hydrogen	1
Oxygen and Nitrogen	2
Carbon with hybridization value < 3	3
Carbon with hybridization value $= 3$	4
Others	5

The two Lennard-Jones parameters – radius (r) and well-depth (ϵ) – are extracted from either CHARMM or AMBER parameter files for each atom type. In summary the atomic attributes are defined by both the force field generation tool (CGENFF or GAFF) and the atom classification scheme AC1, AC5, AC36, AC31 or ACall and we generally refer to these as “FF” atomic attributes.

For comparison, we also examine a different set of atomic attributes, following previous work for the log P predictor from the OpenChem toolkit (`https://github.com/Mariewelt/OpenChem.git`).[24] These atomic attributes are defined as: atom element type, valence, charge, hybridization, and aromaticity. The 3D structure is generated from SMILES strings by RDkit (`https://www.rdkit.org`), then again using RDkit the atomic features are produced for each molecule and represented in one-hot encoding format as node features in OpenChem. These are referred to below as ”OPENCHEM” atomic attributes.

Log P predictions using GSG

Geometric Scattering for graphs

The Geometric scattering for graphs (GSG) method, which has been introduced in Ref.[38] is a feature extraction method for graph data types and uses a geometric transform defined on the graph. In the GSG method, molecules are represented by a graph of atoms where each atom has a vector of attributes; a single attribute evaluated at each vertex is also referred to as a “graph signal”. GSG combines the molecular structure (defined using an adjacency matrix) and atomic signal vectors to construct invariant, stable, and informative features as shown in Figure 1.

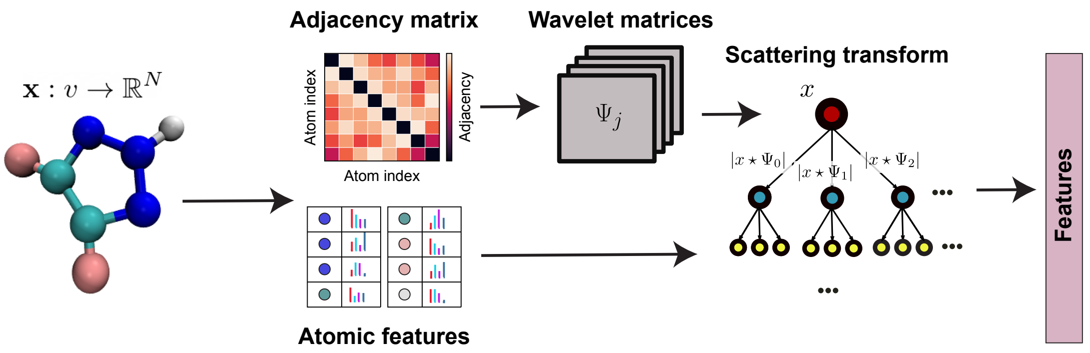


Figure 1: Architecture of the GSG method. The adjacency matrix describes the graph structure of the molecule. Each atom has a set of attributes that are shown as colored bars. Wavelet matrices Ψ are built at different logarithmic scales, j, using the adjacency matrix as described in the text. Finally, the scattering transform is applied to get the graph features using both the wavelet matrices and the signal vectors. Modified from figure made by Feng et al.[38]

Now we briefly explain the mathematics behind this method. Let $G = (V, E, W)$ be a weighted graph and $\mathbf{x}(v_l), 1 < l <= n$ be the signal function defined on each node where n is the number of nodes in the graph and v_l represents node l. A graph random walk operator is defined as $P = \frac{1}{2}(I + AD^{-1})$, where A is the adjacency matrix and D is the degree matrix. P^t describes the probability distribution of a lazy random walk after t steps and hence the term *lazy random walk* for P. In fact, graph random walks are low-pass filters that act like convolution filters to capture graph features at different scales. Graph wavelet operators – also known as "signal transform operators for graphs" – are defined at the scale 2^j as $\Psi_j = P^{2^{j-1}} - P^{2^j}$. Finally, a set of scattering transform operators $\mathbf{S}_0$, $\mathbf{S}_1$ and $\mathbf{S}_2$, are defined using the graph signals and wavelets to compute the molecular-level features. Unique $\mathbf{S}_0$, $\mathbf{S}_1$ and $\mathbf{S}_2$ vectors are created using different moments (q) of $\mathbf{x}$ and their wavelet coefficients Ψ, as well as different wavelet scales indexed by j:

$$\begin{aligned}
(\mathbf{S}_0)_q &= \sum_{l=1}^{n} \mathbf{x}(v_l)^q, \quad 1 \leq q \leq Q \\
(\mathbf{S}_1)_{j,q} &= \sum_{l=1}^{n} |\Psi_j \mathbf{x}(v_l)|^q, \quad 1 \leq j \leq J \quad 1 \leq q \leq Q \\
(\mathbf{S}_2)_{j,j',q} &= \sum_{l=1}^{n} |\Psi_{j'} |\Psi_j \mathbf{x}(v_l)||^q, \quad 1 \leq j < j' \leq J \quad 1 \leq q \leq Q
\end{aligned} \tag{1}$$

The scattering operators are named based on the number of times the wavelets Ψ are used to transform the signal $\mathbf{x}$. For example, $\mathbf{S}_0$ features do not use the wavelet operators, and are simply different moments of the atomic signals. These are called the "Zero order scattering moments." Accordingly, the $\mathbf{S}_1$ operators with one wavelet transformation are called "First order scattering moments" and $\mathbf{S}_2$ with two wavelet transformations are called the "Second order scattering moments." Note that these allow mixing between different wavelet scales as j and j' are set independently.

We define the set $\mathbf{S}$ as the concatenation of all $\mathbf{S}_0, \mathbf{S}_1$ and $\mathbf{S}_2$ vectors using all possible values of q, j and j' as specified by the ranges in Eq. 1. The set of graph features is generated deterministically from the atomic signals and adjacency matrix. The size of this set of features is equal to $N_s Q(1 + J + J(J-1)/2)$, where N_s is the number of attributes per vertex, although this is lower if not all zeroth, first and second order features are used.

As mentioned above, one of the inputs to GSG is the adjacency matrix of the graph and we consider two distinct ways to represent it. One is to use the 2D connectivity of the molecule, where $A_{ij} = 1$ indicates a bond between atoms i and j, and $A_{ij} = 0$ otherwise. Alternatively, the adjacency matrix can be calculated using the 3D structure, where $A_{ij} = f(R_{ij})$, where $f(R)$[43] is a smooth and differentiable function that is equal to 1 at low R and decreases to 0 as R exceeds some cutoff value R_c:

$$f(R_{ij}) = \begin{cases} 0.5(\cos(\frac{\pi R_{ij}}{R_c}) + 1) & \text{for } R_{ij} \leq R_c \\ 0.0 & \text{for } R_{ij} > R_c \end{cases} \tag{2}$$

where R_c is the radial cutoff and R_{ij} is the Euclidean distance between atoms i and j. If $f(R_{ij})$ is non-zero, nodes i and j are considered connected and disconnected otherwise. The wavelet operators in GSG can be defined using either discrete (2D) or continuous (3D) values in the adjacency matrix.

Neural networks architecture and training

To predict the $\log P$ values from the GSG features ($\mathbf{S}$), we develop a feedforward neural network using PyTorch[44] with a nonlinear activation function such as Rectified Linear Unit (ReLU). To determine the best performing network, we consider three important hyper-paramters including the number of hidden layers, the number of neurons in a hidden layer, and the dropout rate whose ranges are shown in Table 4. We perform cross validation using the PyTorch wrapper Skorch `https://skorch.readthedocs.io/en/stable/` to tune the hyperparameters. The loss function of our NN model is MSELOSS (Mean Squared Error) and we use the Adam (adaptive momentum estimation)[45] for optimizing the parameters. We use MultiStepLR, which has an adjustable learning rate set to the initial value of 0.005 and dynamically decreases during training every 15 steps by a factor of 0.5. In training our GSGNN models, we used two data regularization methods: L1 norm and standardization using the StandardScaler function from sci-kit learn. For GSG features with maximum wavelet scales of $J = 4$, 5 and 6 we mostly benefit from using the standardization method. Other settings can be found in Table 4.

Table 4: Neural network settings. Square brackets denote possible parameter values used in the grid search method.

Parameter	Values
Number of hidden layers	$[2, 3, 4, 5]$
Size of hidden layers	$[300, 400, 500]$
Dropout rate	$[0.2, 0.4]$
Initial learning rate	0.005
Learning coefficient	0.5
Batch size	256
Max epoch size	400

Log P predictions using GCNNs

Graph convolutional neural network

Graph convolutional neural networks (GCNNs) extend the application of convolutional neural networks to graph data. The goal of GCNNs is to take a graph and generate features according to node attributes and graph structure. The heart of the GCNN method is the convolution operator, which aggregates the features of neighboring nodes within the graph. Recently, various implementations of GCNNs[46–51] have been developed to increase the speed and accuracy of the GCNN models. In this paper, we employ the GCNN model that was developed in OpenChem based on the method introduced by Kipf and Welling .[49] Similar to the GSG model above, this model takes a graph $G = (V, E)$ as the input where each node has a vector of attributes $\mathbf{x}$. The model processes the graph by passing it through multiple hidden layers performing convolution operations (Figure 2).

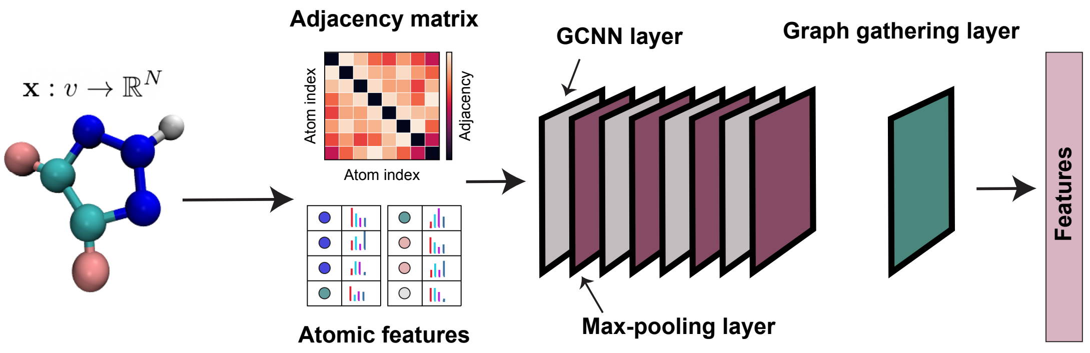


Figure 2: Architecture of the GCNN method. The adjacency matrix describes the graph structure of the molecule. Each atom has a set of attributes and are shown as colored bars. GCNN layers are shown by gray color and are followed a max-pooling layer which is shown in purple. The graph gathering layer is shown in green color adds features on all nodes to generate the molecular feature vector.

The GCNN method works by propagating a feature matrix H – originally set to the value of the attribute vector $\mathbf{x}$ for all nodes – through a set of convolution operators as follows:

$$\begin{aligned} H^{(0)} &= \mathbf{X} \\ H^{(l+1)} &= \sigma(\tilde{A}H^{(l)}W^{(l)}) \end{aligned} \tag{3}$$

where $\tilde{A} = A + I$, is the adjacency matrix of the input graph with self-connections, I is the identity matrix, $W^{(l)}$ is the trainable weight matrix of layer l and σ is a non-linearity function such as ReLU. $H^{(l)}$ denotes the value of the feature matrix on layer l.

Each convolution layer is followed by a graph max-pooling layer introduced in.[50] Following these convolution and max-polling operations , the final set of graph features is obtained by a graph gather layer, where the values of each feature are summed over nodes. This last layer gives GCNNs their index-invariance property.

GCNN architecture and training

We predict $\log P$ using GCNNs using the model implemented in the Openchem toolkit `https://mariewelt.github.io/OpenChem/html/index.html`. This model contains 5 layers of graph convolutions with a hidden layer size of 128. The GCNN layers are followed by max-pooling and a graph gather layer. A 2-layer neural network with ReLU as the activation function is added after the graph gather layer. The PyTorch Adam optimizer and the MSELOSS (Mean Squared Error) are used as the parameter optimizer and training loss function, respectively. We use the MultiStepLR learning scheduler, implemented in PyTorch, with an initial value of 0.001, a step size of 15 and a learning coefficient of 0.5 are used for training the model. Parameters used for the GCNN training are summarized in Table 5.

Table 5: Neural network settings

Parameter	Values
Number of GCNN hidden layers	5
Number of NN hidden layers	2
Size of hidden layers	128
Dropout rate	0
Initial learning rate	0.01
Learning coefficient	0.5
Batch size	128
Max epoch size	200

3 RESULTS

Evaluation of molecular representations

For our set of atomic attributes (ClassicalMD), we use parameters from classical MD force fields: the partial charge σ, and the Lennard-Jones radii and well-depth. We first compare the performance of atomic attributes generated using two different algorithms: GAFF and CGenFF. Both algorithms are used to automatically generate force field parameters for small molecule ligands, by matching atomic environments with atoms that are part of existing force fields; GAFF uses the Amber[41] force field, while CGenFF uses the CHARMM[39,40] force field. Here we generate molecular features from atomic features using GSG (see Section 2). We examine the four different atom type classification schemes discussed above: AC1, AC5, AC36/AC31 and ACall. The GSG parameters used to construct the molecular features are shown in Table 6.

Table 6: Parameters for the Geometric Scattering for Graphs algorithm. The square brackets show all of the values examined for each parameter. For “Scattering operators”, ‘z’ represents the zero order operator, ‘f’ is first order, and ‘s’ is second order.

Parameter	Values
Adjacency matrix (A)	[2D, 3D]
Wavelet maximum scale index (J)	$[4, 5, 6, 7, 8]$
Scattering operators (z, f, s)	Four combinations (z, f), (z, s), (f, s), (z, f, s)
Atom classification	[AC1, AC5, AC36/AC31, ACall]

We trained 160 models for each of the CGenFF and GAFF atomic feature sets, each trained using a 3-fold cross-validation method. After training, we ran tests on subsets of the full database using an 80:20 train:test split. We then calculated evaluation metrics such as the correlation coefficient (r^2), Root Mean squared errors (RMSE) and Mean Unsigned Errors (MUE) between the predicted and experimental $\log P$ values. In Figure 3, r^2 and RMSE values are shown for predictions using 2D and 3D molecular structures. Each r^2 and RMSE is averaged on 20 NN models that are created with all combinations of the 4 geometric scattering operator sets and the 5 wavelet step numbers as shown in Table 6.

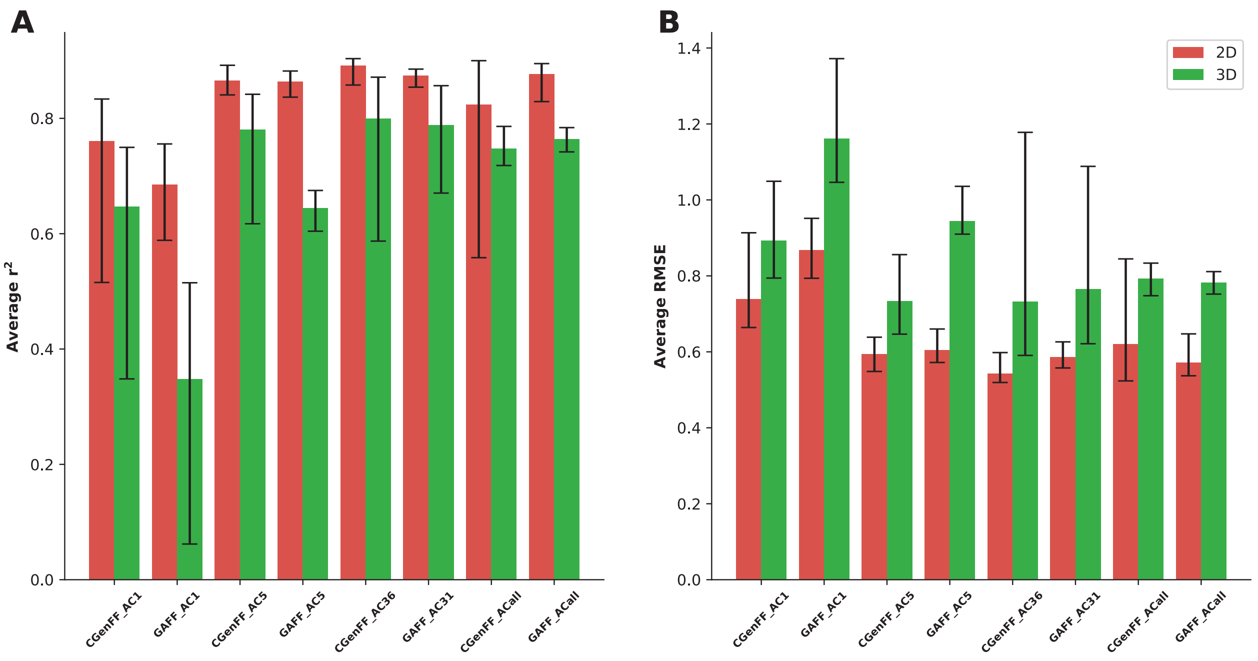


Figure 3: Average r^2 (A) and RMSE (B) for the ChEMBL21 test set using GSGNN models. Each average is calculated over 20 individual parameter values and the error bars show the best and worst performing models. The atomic features are generated with either CGenFF or GAFF force fields and using one of three atom type classification schemes (”AC1”, ”AC5”, ”AC36/AC31” or ”ACall”).

We find that models with 2D structure universally have higher accuracy in prediction of $\log P$ values. Additionally, it demonstrates that CGENFF atomic features on average are more accurate in predicting $\log P$ values compared to GAFF atomic features, although this is not true for ACall. We find that good results on average are obtained by classifying atom types using AC36 categories; the best performing individual models are also found in this category. We thus use CGenFF, 2D structure and AC36 for all models going forward.

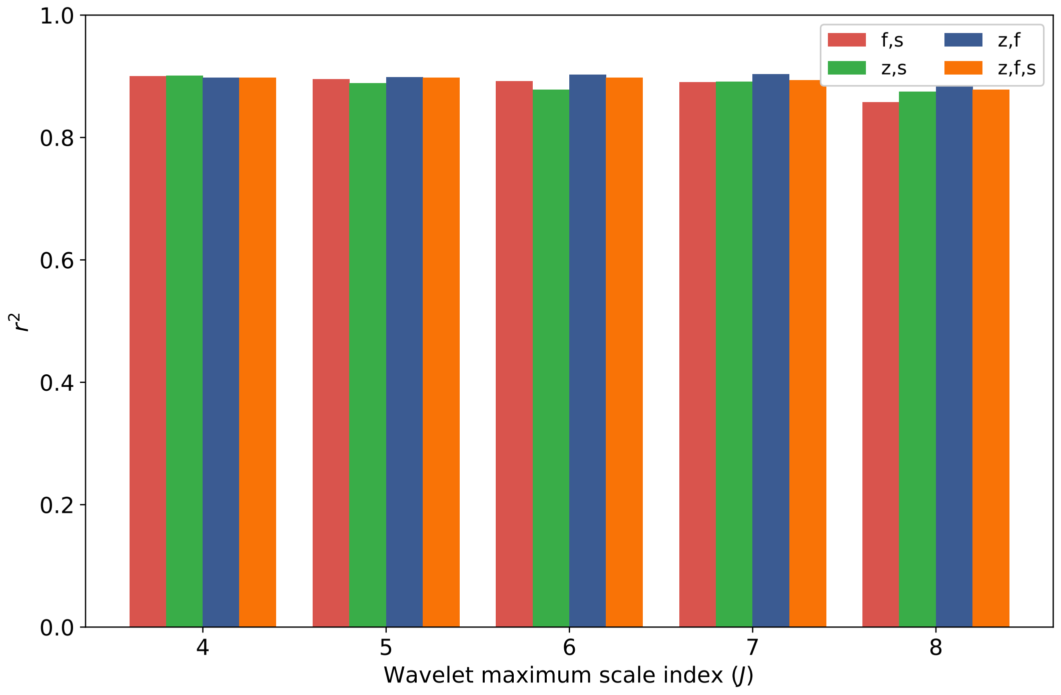


Figure 4: The r^2 for the OpenChem test set using GSGNN models. The atomic features are all generated with CGenFF force fields, AC36 atom type classification scheme, and 2D molecular structure.

We next investigate different scattering moment sets and wavelet scales in Figure 4. We find that for each examined wavelet number $(4, 5, 6, 7$ and $8)$ there is at least one model with an r^2 value of 0.9. The model also performs well for all combinations of scattering moments, making it difficult to determine an optimal set of parameters.

To further evaluate the accuracy of our method, we examine four different independent test sets: FDA, Huuskonen, Star and NonStar which are explained in Section 2. To avoid any accidental overlap, we identified shared molecules and removed them from our training dataset. The training dataset contains $10,722$ randomly selected molecules from the ChEMB21 dataset that successfully are processed by CGenFF. The trained models are evaluated using these test sets and the r^2 between the actual and predicted $\log P$ values are calculated. The results are shown in Figure 5. For the Huuskonen and FDA data sets we find that there is at least one GSGNN model with $r^2 = 0.92$ and $r^2 = 0.90$, respectively.

The best performing models for the Star test set is a maximum wavelet scale of $J = 4$ and zero- and second-order scattering operators, while the best model for NonStar is a maximum wavelet scale of $J = 7$ and zero-, first- and second-order scattering operators.

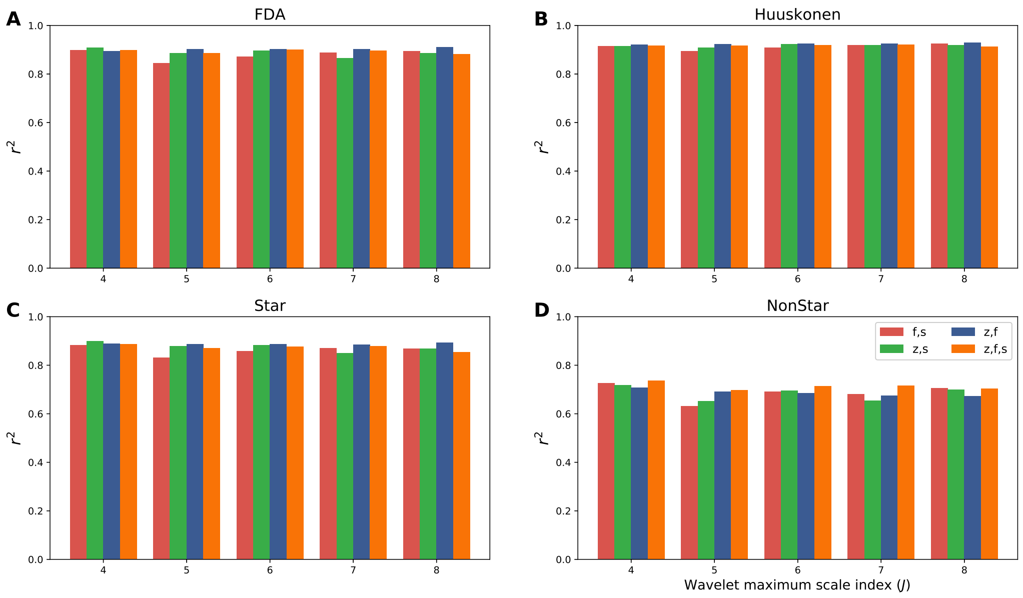


Figure 5: The r^2 for different test sets using GSGNN models. A) shows r^2 for the FDA test set. B) represent r^2 for the Huuskonen test set. C) and D) show r^2 for the Star and NonStar test sets, respectively. The horizontal axis indicates the maximum wavelet scale J. The atomic features are generated with 2D molecular structure, CGenFF force fields and using AC36 atom type classification scheme.

The r^2, RMSE and MUE of best-performing GSGNN model for each test set are shown in Table 7. In paper[23] the $\log P$ values for the FDA, Star, NonStar test sets are predicted using their topology-based models (TopP-S) and the different established methods such as ALOGPS, XLOGP3 and KowWIN. We find that our results do not outperform the TopP-S method but we do achieve higher accuracy compared to other known methods such as ALOGPS, KowWIN and XLOGP3 for FDA test set (Table S3), methods like XLOGP3 and ALOGP for Star test set (Table S4) and XLOGP3 method for NonStar test set (Table S5).

Table 7: The logP prediction performance results for four independent test sets.

Dataset	r^2	RMSE	MUE
FDA	0.91	0.56	0.35
Star	0.90	0.49	0.35
NonStar	0.72	1.02	0.81
Huuskonen	0.93	0.37	0.23

Geometric scattering vs graph convolution

To compare the performance of geometric scattering (GSGNN) models to the graph convolution (GCNN) models, we trained models using these two methods with two different sets of atomic attributes "CLASSICALMD" and "OPENCHEM" as described in Section 2. To rule out the influence of the training/test split, we trained 5 models for each method where the data is randomly divided into test and training sets with 80/20 ratio, respectively. Each of the five GSGNN models is trained using 5-fold cross validation. The RMSE and r^2 are determined for each model and are shown in Table 8. Note that the atomic attributes for these GSGNN models are generated by 2D molecular structure, CGenFF force fields, and AC36 atom type classification scheme. The GSG parameters that used to generate the molecular features are from one of the best performing models (a wavelet maximum scale of $J = 4$ and all three of the zero-, first- and second-order scattering operators) from Figure 4. Our results suggest that the ClassicalGSG method, which is a GSGNN model trained on FF atomic attributes, has the most accurate prediction of $\log P$ values.

Table 8: The logP prediction results using different set of features and models

Method name	Atomic attributes	Model	r^2 (STD)	RMSE (STD)
ClassicalGSG	FF	GSGNN	0.91 (0.003)	0.52 (0.009)
-	OPENCHEM	GSGNN	0.89 (0.003)	0.57 (0.005)
-	FF	GCNN	0.75 (0.091)	0.91 (0.18)
OpenChem	OPENCHEM	GCNN	0.79 (0.052)	0.83 (0.122)

Features visualization

To examine the molecular features that are generated in the pipeline of the ClassicalGSG method, we visualize the features generated by the GSG method and the last layer of NN model using t-distributed stochastic neighbor embedding (t-SNE)[52] plots. The t-SNE plots are intended for projecting high dimensional data into the low dimensional space so it can be visualized readily. Figure 6 shows the GSG and NN features for molecules in the ChEMBL21 test set visualized in a 2D space. Here each point shows a molecule in the ChEMBL21 test set and is colored by its $\log P$ value. The initial dimension of GSG features is 1716 while NN features have size of 400. These GSG features are generated from CGENFF atomic charges, 2D molecular structure, AC36 type classification scheme and all three scattering moment operators with wavelet step number of 4.

We can observe from Figure 6 that features extracted from NN are more discriminative with respect to $\log P$ values. More specifically, in the last layer of the NN model, molecules with similar $\log P$ value tend to be near each other and form distinct clusters. In contrast, GSG features are less discriminative where nearby molecules have different $\log P$ values. To further verify this, five nearest neighbors are determined for each point in the GSG and NN reduced feature space. Then, the difference between the actual $\log P$ value of each point and its five neighbors are calculated and averaged. This value averaged over all the molecules and is shown by $\langle \Delta \log P \rangle_N$ inside the figure. The $\langle \Delta \log P \rangle_N$ in the reduced GSG features space is 0.81 while this value for the reduced NN features space is only 0.17. The average $\log P$ difference between two random points is 2.00. This shows that molecules with similar $\log P$ values are closer in the NN features space.

This is noteworthy, as the GSG features are task independent and therefore are not adapted to any particular prediction task, including $\log P$ prediction. On the other hand, the NN model, which takes as input the GSG features, is a supervised model that is trained for the specific task of $\log P$ prediction. The results in Figure 6, in addition to the results in Tables 7 and 8, illustrate that the GSG features provide not only a translation, rotation, and permutation invariant representation of the molecules, but one that is also sufficiently rich so that, when combined with a downstream supervised NN, the resulting model provides accurate estimates of $\log P$ values for molecules.

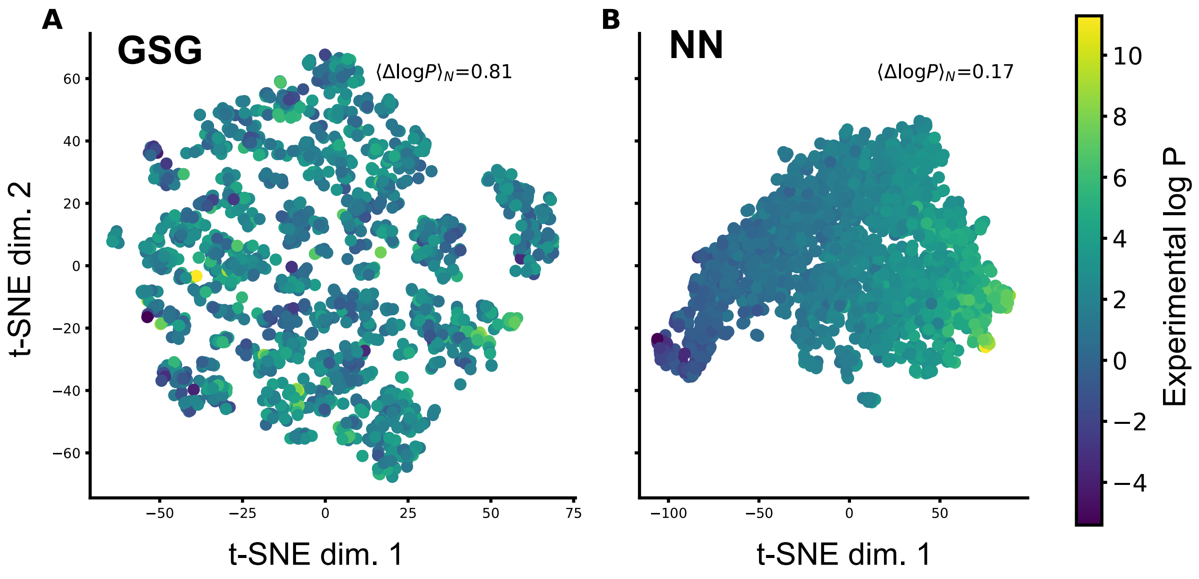


Figure 6: The t-SNE plots with GSG and NN features of the ChEMBL21 test set molecules. Each represents a molecule and is colored by its actual $\log P$ value. $\langle \Delta \log P \rangle_N$ shows the mean $\log P$ difference value calculated over the nearest neighbors in the t-SNE plot. A) The GSG features of size 1716 are projected into 2-dimensional space. B) The NN features from the last hidden layer with size of 400 are projected into 2-dimensional space.

Distinguishing features of failed molecules

To investigate the common characteristics of failed molecules during the prediction of their $\log P$ value, we created molecular fingerprints, which have been used extensively in cheminformatics, QSAR/QSPR predictions, and drug design .[23] Here we employ ChemoPy[30](`https://github.com/dlc62/pychem`) to create a set of constitutional fingerprints. To specify the failed molecules we define a failure cutoff value (here, 0.5) where if the difference between the actual and predicted $\log P$ values is larger than the cutoff, we consider the molecule as a failed prediction. We determine the failed molecules from ClassicalGSG models we described in Section 3. These models are trained on features constructed by CGenFF force fields, 2D structure and AC36 atom type classification method. The probability distribution of 30 constitutional fingerprints are calculated for all molecules in the ChEMBL21 dataset and for failed molecules in each of the 5 GSGNN models. KullbackLeibler divergence (KL-divergence)[53] values are determined by comparing probability distributions of all data with distributions of the failed molecules from each model. The KL-divergence values are averaged over five models and their standard error (STE) is shown in Table 9. We note that the PCX descriptors count the number of shortest paths of length X. The attributes with the highest KL-divergence values are: PC counts with lengths of $2, 1, 3$ and 4; molecular weight (Weight); the number of carbon atoms (ncarb); and the number of heavy atoms (nhev). In other words, the distributions of failed molecules are enriched in particular values of these attributes.

Table 9: The averaged KL-divergence between fingerprint distributions of all data versus failed molecules averaged over 5 GCGNN models.

Fingerprint	Average_KL	STE	Fingerprint	Average_KL	STE
PC2	0.048	0.005	nhet	0.028	0.003
PC1	0.047	0.006	noxy	0.026	0.003
PC3	0.046	0.007	nrot	0.026	0.004
Weight	0.045	0.004	nsulph	0.026	0.003
PC4	0.044	0.004	ndb	0.017	0.002
ncarb	0.043	0.005	ndonr	0.012	0.004
nhev	0.043	0.007	ncof	0.011	0.001
nta	0.043	0.007	AWeight	0.011	0.002
PC5	0.04	0.005	nnitro	0.01	0.001
naro	0.04	0.003	ncocl	0.009	0.002
PC6	0.039	0.005	nhal	0.008	0.001
nsb	0.038	0.004	nphos	0.007	0.003
naccr	0.036	0.002	ncobr	0.006	0
nring	0.035	0.004	ncoi	0.005	0.002
nhyd	0.031	0.003	ntb	0.001	0

The probability distributions of fingerprints with the highest KL-divergence values are shown in Figure 7. The distributions for the failed molecules largely follow the complete dataset distribution, but are enriched towards higher values, suggesting that the $\log P$ predictions are more likely to fail for larger molecules. Interestingly the distributions for attributes that count rare element types (e.g. number of Phosphorus (nphos), number of iodine atoms (ncoi) and number of bromine atoms (ncobr)) do not show large KL-divergence values, although this might have been expected given that they are poorly represented in the training set. The atom element types and their count in our dataset is shown in Figure S1.

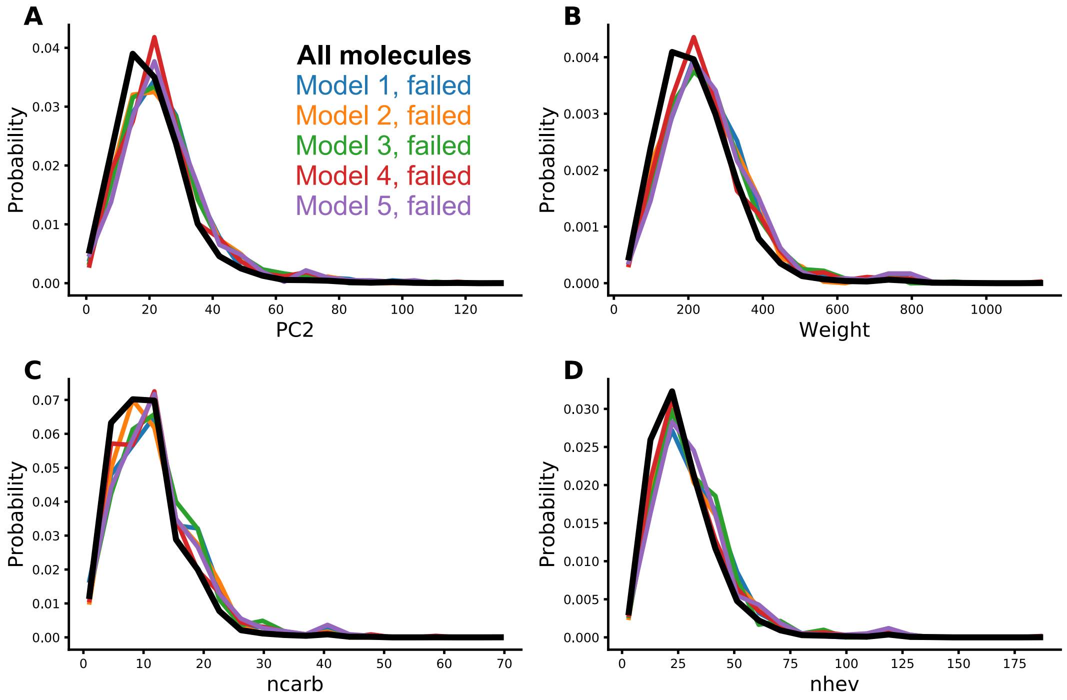


Figure 7: Probability distributions of molecular fingerprints. The histograms show the distribution of fingerprints of all data and failed molecules of 5 GCGNN models. The distribution of all data is shown in thick black line. A) The number of shortest paths of length 2, B) the atomic weight, C) the number of carbon atoms (ncarb) and D) the number of heavy atoms.

4 DISCUSSION AND CONCLUSIONS

In this paper, we introduced a method called ”ClassicalGSG” for predicting the partition coefficient. Our method uses atomic attributes that are usually utilized as parameters for classical MD simulations. These parameters include partial charges, Lennard-Jones well depth, Lennard-Jones radius and atomic type. The pipeline for generating these parameters includes: creating 3D structure from SMILES, creating PDB and Mol2 formatted files, and generating atomic parameters using either Antechamber (GAFF) or CGenFF tools. We note that, in our implementation, the Antechamber tool is about 200 times slower than CGenFF, requiring a couple of days to process 10,000 molecules.

We employ the geometric scattering for graphs (GSG) method to transform the atomic features into molecular features that satisfy re-indexation symmetry. Our results indicate that GSG is powerful enough to capture the universal molecular features, as the average $\log P$ difference for all pairs of adjacent molecules in a t-SNE plot ($\langle \Delta \log P \rangle_N$) is 0.81, which is a low value compared to the random pair $\log P$ difference (2.00). As these features are general to the molecule and not specific to $\log P$, this suggests that they can be used in multi-task NNs to predict other molecular properties, such as solubility ($\log S$), melting point, pKa, and intestinal permeability (e.g. Caco2).[54]

Our results show that employing a 2D molecular structure in ClassicalGSG yields accurate $\log P$ predictions compared to 3D structures, and this confirms the same conclusion achieved previously.[55,56] This could be due to difficulties in generating appropriate 3D structures, or that a single 3D structure is insufficient to capture the high probability conformations of a given molecule. Additionally, our 3D adjacency matrix did not explicitly distinguish between bonded and non-bonded interactions, where the former are much more important to determine molecular properties.

The results reported here for four independent external test sets show that our $\log P$ ClassicalGSG method is generalizable to new molecules. However, we do not expect this model to perform well for molecules with new elements or functional groups that are not covered in the training set. Like other empirical methods, we expect the accuracy will improve as the availability of training data grows.

To facilitate the use of the ClassicalGSG method we made code available on GitHub `https://github.com/ADicksonLab/ClassicalGSG`. This repository contains modules for training and testing NN models using the ClassicalGSG method explained in this paper as well as a trained model that was used to make predictions presented here. A command-line executable is also included for predicting $\log P$ values that takes the mol2 and the CGenFF parameter files of the molecule as input.

5 Acknowledgements

AD acknowledges support from the National Institutes of Health [NIGMS grant R01GM130794] and the National Science Foundation [DMS grant 1761320]. MJH acknowledges support from the the National Institutes of Health [NIGMS grant R01GM135929] and the National Science Foundation [CAREER award 1845856 and DMS grant 1912906].

6 Supporting Information

Additional supporting information is available for this article. Table S1: Classification of CGenFF atom types for the AC36 scheme. Table S2: Classification of GAFF atom types for the AC31 scheme. Table S3: Results from ClassicalGSG and different methods on the FDA test set. Table S4: Results from ClassicalGSG and different methods on the Star test set. Table S5: Results from ClassicalGSG and different methods on the NonStar test set. Figure S1: Counts of different atom types in the ChEMBL21 dataset.

References

1. C. A. Lipinski, F. Lombardo, B. W. Dominy, and P. J. Feeney, Advanced drug delivery reviews **23**, 3 (1997).

2. Y. Kwon, *Handbook of essential pharmacokinetics, pharmacodynamics and drug metabolism for industrial scientists* (Springer Science & Business Media, 2001).

3. Y. Ran and S. H. Yalkowsky, Journal of chemical information and computer sciences **41**, 354 (2001).

4. S. H. Yalkowsky and S. C. Valvani, Journal of pharmaceutical sciences **69**, 912 (1980).

5. T. Ryckmans, M. P. Edwards, V. A. Horne, A. M. Correia, D. R. Owen, L. R. Thompson, I. Tran, M. F. Tutt, and T. Young, Bioorganic & medicinal chemistry letters **19**, 4406 (2009).

6. S. K. Bhal, K. Kassam, I. G. Peirson, and G. M. Pearl, Molecular pharmaceutics **4**, 556 (2007).

7. Z. Chen and S. G. Weber, Analytical chemistry **79**, 1043 (2007).

8. J. Sangster, *Octanol-water partition coefficients: fundamentals and physical chemistry*, vol. 1 (John Wiley & Sons, 1997).

9. T. Cheng, Y. Zhao, X. Li, F. Lin, Y. Xu, X. Zhang, Y. Li, R. Wang, and L. Lai, Journal of chemical information and modeling **47**, 2140 (2007).

10. A. K. Ghose and G. M. Crippen, Journal of computational chemistry **7**, 565 (1986).

11. A. J. Leo, Chemical Reviews **93**, 1281 (1993).

12. W. M. Meylan and P. H. Howard, Journal of pharmaceutical sciences **84**, 83 (1995).

13. J. Plante and S. Werner, Journal of Cheminformatics **10**, 61 (2018).

14. L. Molnár, G. M. Keserű, Á. Papp, Z. Gulyás, and F. Darvas, Bioorganic & medicinal chemistry letters **14**, 851 (2004).

15. J. J. Huuskonen, D. J. Livingstone, and I. V. Tetko, Journal of Chemical Information and Computer Sciences **40**, 947 (2000).

16. I. Moriguchi, S. HIRONO, Q. LIU, I. NAKAGOME, and Y. MATSUSHITA, Chemical and pharmaceutical bulletin **40**, 127 (1992).

17. D. Chen, Q. Wang, Y. Li, Y. Li, H. Zhou, and Y. Fan, Chemosphere **247**, 125869 (2020).

18. R. Mannhold, G. I. Poda, C. Ostermann, and I. V. Tetko, Journal of pharmaceutical sciences **98**, 861 (2009).

19. I. V. Tetko, V. Y. Tanchuk, and A. E. Villa, Journal of chemical information and computer sciences **41**, 1407 (2001).

20. I. V. Tetko and V. Y. Tanchuk, Journal of chemical information and computer sciences **42**, 1136 (2002).

21. A. Predictor, Inc.: Lancaster, CA (2009).

22. Silicos-it, *Filter-it software*, URL `http://silicos-it.be.s3-website-eu-west-1.amazonaws.com/software/filter-it/1.0.2/filter-it.html`.

23. K. Wu, Z. Zhao, R. Wang, and G.-W. Wei, Journal of computational chemistry **39**, 1444 (2018).

24. M. Popova, O. Isayev, and A. Tropsha, Science advances **4**, eaap7885 (2018).

25. J.-W. Zou, W.-N. Zhao, Z.-C. Shang, M.-L. Huang, M. Guo, and Q.-S. Yu, The Journal of Physical Chemistry A **106**, 11550 (2002).

26. H.-F. Chen, Chemical biology & drug design **74**, 142 (2009).

27. E. W. Lowe, M. Butkiewicz, M. Spellings, A. Omlor, and J. Meiler, in *2011 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)* (IEEE, 2011), pp. 1–6.

28. Q. Liao, J. Yao, and S. Yuan, Molecular diversity **10**, 301 (2006).

29. A. Breindl, B. Beck, T. Clark, and R. C. Glen, Molecular modeling annual **3**, 142 (1997).

30. D.-S. Cao, Q.-S. Xu, Q.-N. Hu, and Y.-Z. Liang, Bioinformatics **29**, 1092 (2013).

31. K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, et al., Journal of chemical information and modeling **59**, 3370 (2019).

32. S. Korkmaz, Journal of Chemical Information and Modeling **60**, 4180 (2020).

33. J. S. Delaney, Journal of chemical information and computer sciences **44**, 1000 (2004).

34. A. Lusci, G. Pollastri, and P. Baldi, Journal of chemical information and modeling **53**, 1563 (2013).

35. D. L. Mobley, K. L. Wymer, N. M. Lim, and J. P. Guthrie, Journal of computer-aided molecular design **28**, 135 (2014).

36. J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, Journal of chemical information and modeling **55**, 263 (2015).

37. J. S. Smith, O. Isayev, and A. E. Roitberg, Chemical science **8**, 3192 (2017).

38. F. Gao, G. Wolf, and M. Hirn, in *International Conference on Machine Learning* (2019), pp. 2122–2131.

39. K. Vanommeslaeghe and A. D. MacKerell Jr, Journal of chemical information and modeling **52**, 3144 (2012).

40. K. Vanommeslaeghe, E. P. Raman, and A. D. MacKerell Jr, Journal of chemical information and modeling **52**, 3155 (2012).

41. J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, Journal of computational chemistry **25**, 1157 (2004).

42. D. Case, I. Ben-Shalom, S. Brozell, D. Cerutti, T. Cheatham III, V. Cruzeiro, T. Darden, R. Duke, D. Ghoreishi, M. Gilson, et al., University of California, San Francisco (2018).

43. J. Behler and M. Parrinello, Physical review letters **98**, 146401 (2007).

44. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., in *Advances in Neural Information Processing Systems* (2019), pp. 8024–8035.

45. D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).

46. J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, arXiv preprint arXiv:1312.6203 (2013).

47. M. Henaff, J. Bruna, and Y. LeCun, arXiv preprint arXiv:1506.05163 (2015).

48. W. Hamilton, Z. Ying, and J. Leskovec, in *Advances in neural information processing systems* (2017), pp. 1024–1034.

49. T. N. Kipf and M. Welling, arXiv preprint arXiv:1609.02907 (2016).

50. H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, ACS central science **3**, 283 (2017).

51. D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, in *Advances in neural information processing systems* (2015), pp. 2224–2232.

52. G. E. Hinton and S. T. Roweis, in *Advances in neural information processing systems* (2003), pp. 857–864.

53. S. Kullback and R. A. Leibler, The annals of mathematical statistics **22**, 79 (1951).

54. I. J. Hidalgo, T. J. Raub, and R. T. Borchardt, Gastroenterology **96**, 736 (1989).

55. R. D. Brown and Y. C. Martin, Journal of Chemical Information and Computer Sciences **37**, 1 (1997).

56. G. B. McGaughey, R. P. Sheridan, C. I. Bayly, J. C. Culberson, C. Kreatsoulas, S. Lindsley, V. Maiorov, J.-F. Truchon, and W. D. Cornell, Journal of chemical information and modeling **47**, 1504 (2007).