Large-scale comparison of machine learning methods for drug target prediction on ChEMBL — *Supplementary* —

Andreas Mayr[1]*, Günter Klambauer[1]*, Thomas Unterthiner[1]*, Marvin Steijaert[2], Jörg K. Wegner[3], Hugo Ceulemans[3], Djork-Arné Clevert[4], and Sepp Hochreiter[1]

[1]LIT AI Lab and Institute of Bioinformatics, Johannes Kepler University Linz, Austria
[2]Open Analytics NV, Belgium
[3]Janssen Pharmaceutica NV, Belgium
[4]Bayer AG, Germany
*These authors contributed equally to this work

Contents

List of Figures

List of Tables

S1 Overview

This supplemental report comprises the following parts: The first part (Section S2) details how we obtained datasets and folds for comparisons. The second part (Section S3) describes the target prediction methods we compared. The third part provides additional details on the results shown in the main text (Section S4). The appendix presents descriptions of external tables.

S2 Data Collection and Clustering

In this section, we first provide details on how we generated the benchmark dataset for the target prediction methods (Section S2.1). In particular, we present criteria for considering a measurement to be active or inactive. Further, we describe how compounds were represented (Section S2.2) and how we derived compound clusters and cross-validation folds from these compound representations (Section S2.3). Finally, we list some criteria for identifying assay pairs for the prediction performance comparison between in-vitro assays and in-silico assays and we give further details on the comparison procedure (Section S2.4).

S2.1 Extraction of a Benchmark Dataset from ChEMBL

Overall, the ChEMBL 20 database contains 1,456,020 compounds, 13,520,737 bioactivity measurements, and 1,148,942 assays assigned to 10,774 targets. The distribution of measurements across assays is highly unbalanced. There are assays with a large number of measurements (tens of thousands), while for more than half of the assays there is only a single measurement available. Similarly, the distribution of assays across targets is unbalanced. For many targets there is only a single assay in ChEMBL, while for others there are more than a hundred different assays. ChEMBL also contains repeated measurements (i.e., compounds were measured multiple times by the same assay). Because of this heterogeneity of assays and their outcomes, it is often unclear what measurements can be considered active, inactive or unknown. Therefore, we defined a protocol for generating a benchmark dataset from ChEMBL in which binary labels (active/inactive) are assigned to compound-measurement pairs:

1. **Standard Activity comment**: The ChEMBL table `activities` includes an `activity_comment` field. If the `activity_comment` was either `"Active"` or `"active"`, we considered a measurement active; if the comment was either `"inactive"`, `"Not Active"` or `"Not Active (inhibition < 50% 10 uM and thus dose-response curve not measured)"` we considered the measurement to be inactive. In these cases, we did not consider any further details about the measurements.
2. **Activity filter**: Of the measurements that had no standard activity comments as described in Step 1., we discarded those for which either `standard_value` was empty, `standard_units` was unequal to `"nM"` or `standard_relation` was not in {`">"`, `">="`, `"<"`, `"<="`, `"="`, `"~"`}
3. **Assignment of labels**: We considered density plots of active and inactive measurements with given standard activity comments and determined thresholds for activity and inactivity. These thresholds are given in Table S1; we considered a measurement to be indeterminate

if it was neither clearly active nor clearly inactive according to our activity threshold. Additionally, for comparing the performance of Deep Learning with that of in-vitro assays (not for method comparison), we allowed weak labels for the comparison of DNN predictions and surrogate in-vitro assay predictions to their ground truth (not for training DNN models). To all measurements that passed the activity filter described in Step 2. and that had no standard comment in Step 1., we assigned labels according to this table, depending on `standard_value` and `standard_relation` (note that the negative logarithm changes the direction of the inequality sign for the threshold).

Label	**Threshold in -log10(M)**	**Relation Type**
active	≥ 5.5	`"<", "<=", "=", "~"`
weak active	> 5.0 and < 5.5	`"<", "<=", "=", "~"`
inactive	≤ 4.5	`">", ">=", "=", "~"`
weak inactive	> 4.5 and < 5.0	`">", ">=", "=", "~"`
indeterminate	> 4.5 or < 5.5	any

Table S1: Activity thresholds for measurements without standard activity comments

4. **Removing contradicting measurements**: We merged all measurements obtained in steps 1. and 3., and removed all measurements for which compounds were reported to be both active and inactive in the same assays.

Applying this protocol created a cleaned-up ChEMBL dataset comprising 6,882,639 measurements from 290,041 assays and 1,057,015 compounds. We further removed assays with fewer than ~ 100 measurements to avoid the problem of an insufficient number of data points being available either for training or for evaluation. Additionally, we had to ensure that each cross-validation fold contained at least one active and one inactive compound to be able to calculate an ROC-AUC evaluation value. Finally, we obtained a benchmark dataset with 1,310 assays and 4,743,712 assay measurements of 456,331 compounds. The exact list of assays used is given in Table S10.

S2.2 Representation of Chemical Compounds

A very commonly used description of a chemical compound is the graph representation, which is often given by an adjacency matrix. A further representation is given by strings, that describe the molecule graph. Examples are SMILES [Weininger, 1988] or InChI [Heller et al., 2013] representations.

However, a graph or string representation can only be processed by some algorithms, while others expect a representation of the inputs as feature vectors. Therefore, we computed a number of chemical descriptors using standard software [e.g., Cao et al., 2013, Hinselmann et al., 2011, RDKit]. As in [Mayr et al., 2016] and as mentioned in the main text, we grouped types of chemical descriptors into static features, which are typically identified by experts as promising properties for predicting biological activity, and dynamic features, which are extracted on the fly from the chemical structure of a compound in a prespecified way. The most obvious difference between static and dynamic features is that the number of static features is fixed and equal for each compound, while the number of dynamic features that describe a compound is compound-specific. We defined “semi-sparse” features as an additional category. We consider a feature to be semisparse if it is predefined or fixed in size, as is typical of static features, but sparse like most dynamic features. An example of semisparse features are MACCS descriptors. Further, we generated 2,200 toxicophore descriptors [Mayr et al., 2016] as a separate feature category.

In detail we used the following features categories:

- Common static features (**StaticF**, static) as provided by [Cao et al., 2013]:
  - Basak information indices
  - Burden descriptors: eigenvalues of Burden matrices [Burden, 1989]
  - Charge descriptors
  - Charged partial surface area (CPSA) descriptors [Stanton and Jurs, 1990]
  - Constitutional descriptors: number of atoms of type “C”, “N”, molecular weight, number of charges, number of aromatic rings, etc.
  - Electrotopological state indices [Kier and Hall, 1990]
  - Geary autocorrelation descriptors [Geary, 1954]
  - Geometric descriptors
  - Kappa shape descriptors [Hall and Kier, 1991]
  - Molecular connectivity indices [Hall and Kier, 1991]
  - Moran autocorrelation descriptors [Moran, 1950]
  - Moreau-Broto autocorrelation [Broto et al., 1984]
  - MoRSE descriptors [Schuur et al., 1996]
  - Radial Distribution Function (RDF) descriptors [Hemmer et al., 1999]
  - Topological descriptors
  - MOE-type descriptors for surface area
  - Molecular properties: molar refractivity, lipophilic contributions of atoms, etc.
  - Weighted Holistic Invariant Molecular (WHIM) descriptors [Todeschini et al., 1994, Todeschini and Gramatica, 1997]
- Common semisparse features (**SemiF**, semispare):
  - Chemically Advanced Template Search [CATS2D, Schneider et al., 1999, Renner et al., 2006] as implemented in [Hinselmann et al., 2011]
  - MACCS fingerprints: 166 common substructure features
  - PubChem Substructure Fingerprints (PCFP): 881 binary substructure features
  - Shannon Entropy Descriptors [SHED, Gregori-Puigjané and Mestres, 2006] as implemented in [Hinselmann et al., 2011]
  - Daylight-like fingerprint features as implemented in [RDKit]
- Toxicophore features (**ToxF**, semispare): about 2,276 in-house toxicophore features which comprise substructures previously reported as toxicophores

- ECFP features [dynamic, Rogers and Hahn, 2010] with counts and radii 2 (**ECFP4**) and 3 (**ECFP6**) and DFS features [dynamic, Swamidass et al., 2005] with counts and diameter 8 (**DFS8**) as implemented in [Hinselmann et al., 2011] using the option `ELEMENT_SYMBOL` for encoding atom types for ECFP6 and DFS8 and `DAYLIGHT_INVARIANT_RING` for encoding atom types for ECFP4.

S2.3 Clustering the ChEMBL Database and Definition of Folds

As mentioned in the main text, we clustered all 1,456,020 compounds of the ChEMBL database using single-linkage clustering, which is an algorithm that is able to find a clustering with guaranteed minimum distances minD between any two different clusters. This is an important property for cluster-cross-validation, as it avoids that a compound in the training set is closer than minD to a compound in the test set. In single-linkage clustering, the actual distance D between two clusters or sets of points $\mathrm{Cluster}_1$ and $\mathrm{Cluster}_2$ is defined by the minimum distance d between any two points from the two sets:

$$D(\mathrm{Cluster}_1, \mathrm{Cluster}_2) = \min_{p_1 \in \mathrm{Cluster}_1, p_2 \in \mathrm{Cluster}_2} d(p_1, p_2).$$

For the distance d between compounds, we used the Jaccard distances on binarized ECFP4 compound representations. Any two identified different clusters of compounds $\mathrm{Cluster}_1$ and $\mathrm{Cluster}_2$ finally fulfil the property $D(\mathrm{Cluster}_1, \mathrm{Cluster}_2) \geq \mathrm{minD}$, where minD can be considered as a hyperparameter for the clustering, as it is not only a bound for the distances between any two clusters identified, but also determines whether sets of compounds should be merged in the process of identifying clusters.

Note that for high values of minD, all compounds fall into one or a few large clusters, while for small values very small clusters with only a few compounds emerge. As the number of clusters can become very high, and an imbalanced clustering with some very large and many small clusters may be found, we decided to finally merge the clusters to form three folds of approximately equal size. Merging all compounds of a cluster to the same fold ensures that we have minimum distances minD between compounds of different folds.

We computed clusterings for various minD values and checked the resulting cluster size distribution. We chose $\mathrm{minD} = 0.3$, as around this threshold the large clusters seemed to fall apart. Thus, we obtained 425,252 clusters of compounds with 103,287 and 9,134 compounds for the largest and the second-largest clusters, respectively (see Table S2). We considered this to be a good trade-off between an excessive number of clusters and a very degenerate clustering with one or a few extremely large clusters. Since we merged clusters to form 3 different folds with about 500,000 compounds each, one cluster with about 100,000 compounds seemed acceptable. Table S3 shows the actual sizes of the 3 folds, produced by merging clusters.

Cluster Nr.	1	2	3	4	5	6	7	8	9	10
Size	103287	9134	3701	3535	3425	3155	3045	2854	2761	2595

Table S2: Ten largest chemical clusters identified in ChEMBL, ordered by size

Fold Nr.	1	2	3
Size	551173	455840	449007

Table S3: Sizes of the three folds

S2.4 Comparison of in-vitro Assays to in-silico Assays

S2.4.1 Search for Assay Pairs

In order to find assays that rely on different biotechnological principles, but are designed to measure the same effect, we considered assay pairs with the following properties: both share the same target, and one of the two is described as a high-throughput assay, and the second is not explicitly described as a high-throughput assay. Note that this does not necessarily mean that the second assay is not a high-throughput assay. Further, we aimed at a minimal overlap between assays in terms of compounds.

The exact criteria were:

- identical biomolecular targets assigned to both assays;
- at least 15 compounds measured in both assays;
- one assay described as a high-throughput assay;
- the other assay not described as a high-throughput assay.

Further, we checked the assay descriptions manually to determine whether the assays were intended to measure the same biological effect. The assay pairs found in this manual check are given in the main text and Table S14. The assay pairs, which were removed by the manual check are listed in Table S15.

S2.4.2 Performance Comparison

We trained three multitask FNNs on all assays from our benchmark dataset that correspond to the three folds of the benchmark dataset. Each FNN was trained on two folds of compounds and the values for the compounds from the remaining fold were predicted with the trained FNNs. The predictions from the three folds were aligned to one prediction matrix for all compounds and assays of the benchmark dataset. Note that for an entry of this prediction matrix, the cluster of the corresponding compound was excluded from the training data (but not the whole two assays).

To compare FNN predictions to a surrogate in-vitro assays in terms of prediction performance for a selected in-vitro assay, that had to be predicted, we applied a proportion test that considered FNN predictions against the in-vitro assay that serves as the ground truth assay as well as the surrogate in-vitro assay with the same target as the ground truth assay against the ground truth assay itself. For FNN predictions we had to find a threshold, to differentiate inactive compounds from active compounds. Therefore, we randomly split the compounds of FNN predictions into a set for optimising the threshold and a set to evaluate predictions against the ground truth assay. Note, that we used the same threshold for all folds. Although in general it is not guaranteed, that the same threshold can be used for each fold, it seemed to work here.

S3 Methods

We considered several methods for predicting the outcomes of assays. A basic description can be found in the main text. Here we provide further details for reproducibility reasons.

S3.1 Deep Learning

S3.1.1 Overview

Formal Description of Deep Neural Networks A deep neural network can be considered a function that maps the input vector $\mathbf{x}$, which is the representation of a compound by its features, to an output vector $\mathbf{y}$ that represents the activity prediction. This function has parameters, so-called weights, that are organized in layers. Figure S2 shows a general deep neural network architecture. A layer implements a linear mapping of its inputs to its outputs followed by a non-linear activation function f (see Figure S1 for a visualization of the rectified linear unit [ReLU, Nair and Hinton, 2010, Glorot et al., 2011], the sigmoid, and the scaled exponential linear unit [SELU, Klambauer et al., 2017] activation functions mentioned in the main text). Let w_{ji}^{l} be the weight connecting neuron i in layer $l-1$ to neuron j in layer l, and b_j^l be the intercept of neuron j, then the following formula computes the output $\mathbf{y}$ of a neural net with m layers:

$$\begin{aligned}
\mathbf{h}^0 &= \mathbf{x}\,; \\
h_j^l &= f\Big(\sum_i w_{ji}^l\, h_i^{l-1} + b_j^l\Big)\,; \\
\mathbf{y} &= \mathbf{h}^m\,.
\end{aligned}$$

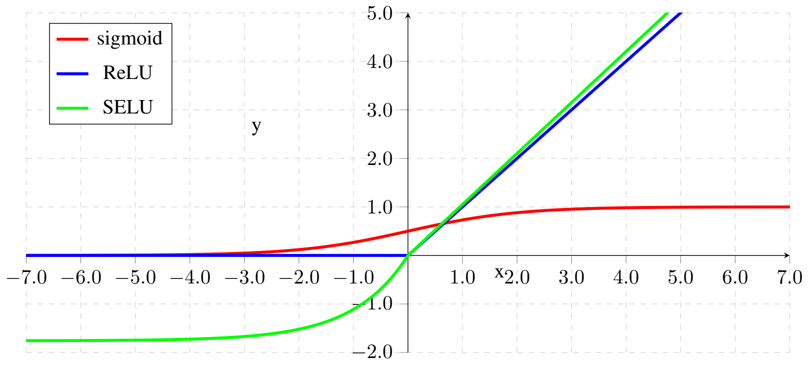

Figure S1: Comparison of SELU, ReLU and sigmoid activation functions

Objective Function In order to produce accurate predictions, a neural network aims to minimize the errors of the predictions y_k for the given training data t_k of a task k as determined by an appropriate error function. A typical error function for deep networks that perform a binary prediction task is the cross entropy error function, which is to be minimized with respect to the

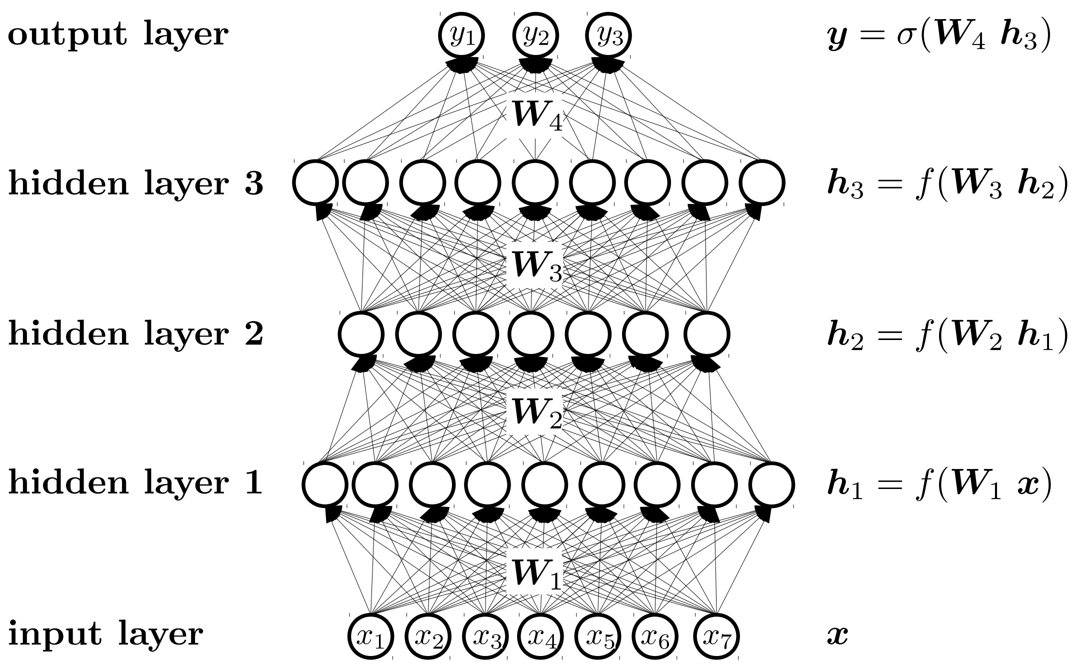
output layer
hidden layer 3
hidden layer 2
hidden layer 1
input layer
y_1 y_2 y_3
$\boldsymbol{W}_4$
$\boldsymbol{W}_3$
$\boldsymbol{W}_2$
$\boldsymbol{W}_1$
x_1 x_2 x_3 x_4 x_5 x_6 x_7
$\boldsymbol{y} = \sigma(\boldsymbol{W}_4\ \boldsymbol{h}_3)$
$\boldsymbol{h}_3 = f(\boldsymbol{W}_3\ \boldsymbol{h}_2)$
$\boldsymbol{h}_2 = f(\boldsymbol{W}_2\ \boldsymbol{h}_1)$
$\boldsymbol{h}_1 = f(\boldsymbol{W}_1\ \boldsymbol{x})$
$\boldsymbol{x}$

Figure S2: Architecture of DNNs

neural network weights. In chemoinformatics, the activity information is usually sparse, which means that compounds are measured only for a certain number of targets. For these targets k we know whether the compound was active ($t_k = 1$) or inactive ($t_k = 0$). For all other compounds we do not have such information. We therefore used a modified cross entropy function that considers missing label information by using a binary mask m_k, which is one if a sample has a label for task k and zero otherwise. Our objective function to be minimized is therefore:

$$- \sum_{k=1}^{K} m_k \left(t_k \ \log(y_k) \ + \ (1 - t_k) \ \log(1 - y_k) \right) \ .$$

S3.1.2 Standard Feed-forward Neural Networks (FNNs)

Training of FNNs and software implementation We used Minibatch Stochastic Gradient Descent (SGD) with a batchsize of 128 to train the FNNs. The input features to the networks were normalized to mean 0 and standard deviation 1. We had to apply feature selection for the dynamic features, since the whole set of features from a dynamic feature category (i.e. ECFP, DFS, ECFP+ToxF) is extremely large (several 100,000s). Because of the sizes of the emerging matrices and the number of networks to train, training networks with all features from a category would be computationally infeasible. We therefore removed features that were sparse, i.e. which were only present in a few compounds. In detail, for ECFP and ECFP+ToxF we removed features, that occured in less than 0.25% of all compounds in the respective training sets, for DFS (which was less sparse than ECFP) we removed features, that were present in less than 2%.

Hyperparameter search We wanted to keep the hyperparameter search space small in order to deal with the large computational effort per experiment. Therefore, we performed a number of smaller experiments on a subset of the data to determine sensible ranges for some hyperparameters. Finally, we performed a hyperparameter search for the activation function and the corresponding network architecture, the number of hidden units per layer, the number of layers, the learning rate and to decide whether Input Dropout should be used. Table S4 shows a list of these hyperparameters and architecture design parameters that were used for the FNNs, together with their search ranges. We considered all combinations of these hyperparameters, except 2 and 4 hidden layer networks for 1024 and 4096 hidden units (in order to keep the search space smaller and with the assumption that these combinations may be close to other combinations in the grid).

Hyperparameter	Values considered
Architecture/Activation Function	{ReLU, SELU}
Number of Hidden Units	{1024, 2048, 4096}
Number of Hidden Layers	{2, 3, 4}
Learning Rate	{0.01, 0.1}
Input Dropout	{0.2, 0.0}
Dropout	{0.5}

Table S4: Hyperparameters considered for FNNs

S3.1.3 Graph Convolution Networks

Training of graph convolutional networks and software implementation We used a basic graph convolutional network implementation [GC: GraphConvTensorGraph, Duvenaud et al., 2015, Wu et al., 2018] and the Weave convolutional networks [Weave: WeaveTensorGraph, Kearnes et al., 2016] from the DeepChem package [2016].

While the GC architectures use a sequence of Convolution and Pooling Layers (that are more or less sequences of matrix multiplications and subsequent maximum operations), the core of the Weave architectures is the usage of a sequence of Weave modules, that allow for pair features and construct atom and pair output features from atom and pair input features. Therefore, several (complex) operations, that preserve invariances are defined.

In the version, we used (DeepChem 1.3.1), it seemed, that some functionalities needed to be modified (Batch Normalization), to allow networks to be trained efficiently. To train GC and Weave Tensorgraph models, we used a batchsize of 128 and used the standard optimizer of Tensorgraph models (Adam [Kingma and Ba, 2014]).

Hyperparameter search We basically used the suggested DeepChem default architectures, but extended them a little, to allow more hyperparameters to define the final architecture. Table S5 shows a list of these hyperparameters and the values we considered.

Hyperparameter	Values considered
Size of Dense Layer	{1024, 2048}
Size of Graph Layers	{128}
Number of Graph Layers	{2, 3, 4}
Learning Rate	{0.001, 0.0001}
Dropout	{0.0, 0.5}

Table S5: GC and Weave Hyperparameters considered for Graph Convolutional Networks

S3.1.4 SmilesLSTM

Training of SmilesLSTM and software implementation We used long short-term memory (LSTM) recurrent neural networks [Hochreiter and Schmidhuber, 1997] based on the 1-hot encoded SMILES representation [Weininger, 1988] of chemical compounds. The full architecture of this model "SmilesLSTM" consists of one or several 1D-convolutional layers with SELU activations [Klambauer et al., 2017] followed by mean- or max-pooling layers, then one or multiple stacked LSTMs and a final fully-connected output layer. The architecture is depicted in Figure S3. We implemented the SmilesLSTM in the Keras (version 2.1.2) [Chollet, 2015] library with Tensorflow backend. All parameters of the network layers are left to default except the ones given in Table S6. We used different optimizers, such as SGD and Adam [Kingma and Ba, 2014], with a batchsize of 128 together with a momentum term to train the SmilesLSTM.

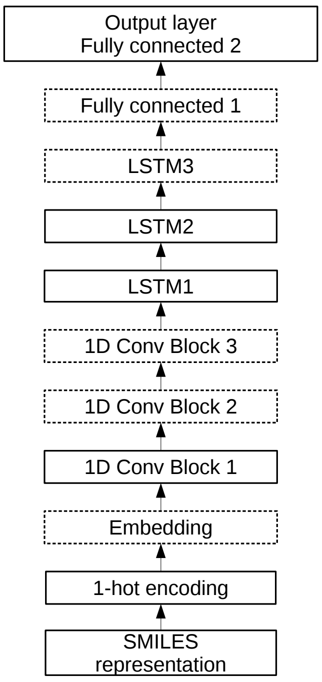

Figure S3: The principle architecture of SmilesLSTM: The one-hot encoded input sequence is passed to 1D-convolutional layers with pooling layers. Then stacked LSTMs are applied and a final fully-connected output layer is used.

Hyperparameter Selection The hyperparameter space of this model is large, which makes grid-search prohibitively expensive with respect to computational costs. The list of hyperparameters with considered values is given in Table S6. Instead of grid-search we use manual hyperparameter search [Goodfellow et al., 2016] on the inner fold of our nested cross-validation procedure.

S3.2 Support Vector Machines (SVMs)

Formal Description of SVMs Basically, SVMs classify samples using a prediction function of the following form:

$$y = \operatorname{sign}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right).$$

Here, b is the bias weight and $\mathbf{w}$ are the feature weights, which are determined by solving a convex minimization problem. In this optimization problem the strength of the weights $\mathbf{w}$ is minimized while ensuring correct classifications by constraints. The weights can be expressed as a linear combination of the training feature vectors $\mathbf{x_s}$ of sample s, where the labels t_s (0 or 1) and the dual variables α_s resulting from convex optimization, serve as scaling factors for the corresponding sample feature vectors:

$$\mathbf{w} = \sum_s \alpha_s t_s \mathbf{x_s}.$$

Using this formula allows us to express the classification function without explicitly computing the feature weights $\mathbf{w}$. Therefore, the classification function can be rewritten as:

$$y = \operatorname{sign}\left(\sum_s \alpha_s t_s \langle \mathbf{x_s}, \mathbf{x} \rangle + b\right).$$

The basic SVM classification function can be considered a linear classifier. To allow nonlinear classification, the kernel trick can be applied, that is replacing the scalar product $\langle .,. \rangle$ with a more general kernel function $k(.,.)$ that should be positive definite. Mathematically, this can be interpreted as projecting the samples to a higher dimensional space in which the scalar product is computed. From an application point of view, we can consider the kernel function as a similarity measure between samples.

Training and software implementation We used the LIBSVM implementation [Chang and Lin, 2011] for computing the dual variables. Since run time complexity is quadratic in the number of samples, we had to integrate the LIBSVM library into an OpenMP application that was run on a supercomputer.

Kernel function As mentioned in the main text, we used the Minmax kernel. For compounds $\mathbf{x}$ and $\mathbf{z}$ [Mayr et al., 2016] it is:

$$k_{\text{Minmax\_new}}(\mathbf{x}, \mathbf{z}) = \frac{\sum_{P \in \mathcal{P}:\ N(P,\mathbf{x})+N(P,\mathbf{z})>0} \frac{\min(N(P,\mathbf{x}),N(P,\mathbf{z}))}{\max(N(P,\mathbf{x}),N(P,\mathbf{z}))}}{\sum_{P \in \mathcal{P}:\ N(P,\mathbf{x})+N(P,\mathbf{z})>0} 1}.$$

Hyperparameter	**Considered values**	**Selected hyperp.**
Number of FC layers	$\{1, 2\}$	1
Number of units in second FC layer	$2^{[9,11]}$	–
L1 Weight Decay in FC layers	$\{10^{-9}, 10^{-8}, 10^{-7}, 0\}$	0
L2 Weight Decay in FC layers	$\{10^{-9}, 10^{-8}, 10^{-7}, 0\}$	10^{-9}
Number of LSTM layers	{1,2,3}	2
Number of units in LSTM3	$2^{[2,10]}$	–
Dropout rate in LSTM3	$[0, 0.5]$	–
Number of units in LSTM2	$2^{[2,9]}$	2^9
Dropout rate in LSTM2	$[0, 0.5]$	0.1
Number of units in LSTM1	$2^{[2,9]}$	2^7
Dropout rate in LSTM1	$[0, 0.5]$	0.1
Convolutional block 3	{yes, no}	no
Pooling type after convolutional block 3	{no, mean, max}	–
Number of kernels in convolutional block 3	$2^{[2,10]}$	–
Stride in convolutional block 3	{1–3}	–
Kernel size of convolutional block 3	{1–6}	–
Dropout rate in convolutional block 3	$[0, 0.75]$	–
Convolutional block 2	{yes, no}	no
Pooling type after convolutional block 2	{no, mean, max}	–
Number of kernels in convolutional block 2	$2^{[2,10]}$	–
Stride in convolutional block 2	{1–3}	–
Kernel size of convolutional block 2	{1–6}	–
Dropout rate in convolutional block 2	$[0, 0.75]$	–
Convolutional block 1	{yes, no}	yes
Pooling type after convolutional block 1	{no, mean, max}	max
Number of kernels in convolutional block 1	$2^{[2,10]}$	2^5
Stride in convolutional block 1	{1–3}	2
Kernel size of convolutional block 1	{1–6}	4
Dropout rate in convolutional block 1	$[0, 0.75]$	0.5
Embedding	{yes, no}	no
Embedding dimension	$2^{[2,10]}$	–
Input dropout rate	$[0, 0.2]$	0.1
Learning Rate	$10^{[-5,-1]}$	10^{-4}
Optimizer	{SGD, Momentum, Adam}	Adam
Activation function for convolutional layers	{ReLU, SELU}	SELU
Early-stopping parameter (epochs)	{1–300}	300

Table S6: Hyperparameters considered for SmilesLSTM. Manual hyperparameter selection was performed on the inner fold of a nested cross-validation procedure to avoid the hyperparameter selection bias.

Here, $N(P, \mathbf{x})$ quantifies feature P for compound $\mathbf{x}$. In total, $|\mathcal{P}|$ features are considered. Other definitions of Minmax kernels exist. The advantage of our version is, that it is scale invariant with respect to the features. Additionally, we scale the kernel value with a parameter γ and apply the $\exp$ function to it.

Hyperparameter search We applied hyperparameter selection over the cost parameter for misclassified samples as well as over γ for our kernel. Table S7 gives an overview of the hyperparameters considered.

Hyperparameter	Values considered
Cost	{0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0}
γ	{no scaling and exponentiation, 0.1, 1.0, 10.0, 100.0}

Table S7: Hyperparameters considered for SVMs

S3.3 Random Forests (RFs)

For RF hyperparameters, we used a high number of trees to obtain a stable model with good performance [Oshiro et al., 2012]. The critical parameter is the number of features considered at each split [Louppe, 2014], which we adjusted in a nested cross-validation setting. We trained and assessed models in our established framework using various hyperparameters (see Table S8) and used the RF implementation “ranger” [Wright and Ziegler, 2015].

Hyperparameter	Values considered
number of trees	{1000}
number of features considered at each split	$\{1, 2, 3\} \times \sqrt{\text{total number of features}}$

Table S8: Hyperparameters considered for RFs

S3.4 K-Nearest Neighbour

Distance function For comparability with the SVM results, we simply defined the distance to be $1 - k_{\text{Minmax\_new}}(\mathbf{x}, \mathbf{z})$ to determine the nearest neighbour. Further, we used the same similarities as in the SVM case.

Hyperparameter search The natural hyperparameter is the number K of nearest neighbours. It was selected from the set {1, 3, 5, 10, 50, 100}.

S3.5 Naive Bayes (NB) Statistics

Formal Description This approach computes Laplacian-adjusted probability estimates for the features, which leads to individual feature weights that are finally summed to give the predictions.

The method defines the probability of a compound $\mathbf{x}$ to be active by the ratio between the number of active samples S_k^+ in the training set and the training set size S_k:

$$p\left(y_k = 1\right) \; = \frac{S_k^+}{S_k}.$$

It is assumed that a compound is described by a set of features $P \in \mathcal{P}$. A Laplace-corrected probability for a compound $\mathbf{x}$ to be active if a certain feature P is available is computed from the number of compounds S_k^P containing the feature and the number $S_k^{P,+}$ of active compounds from those which contain P:

$$p_{corr}\left(y_k = 1|P\right) \; = \frac{S_k^{P,+} \; + \; p\left(y_k = 1\right) \; \alpha}{S_k^P \; + \; \alpha},$$

where α is defined as $\frac{1}{p(y_k=1)}$. Finally, a weight for feature P is computed as:

$$w_{k,P} = \log\left(\frac{p_{corr}\left(y_k = 1|P\right)}{p\left(y_k = 1\right)}\right).$$

A prediction score for compound $\mathbf{x}$, consisting of a set of features, is then obtained by summing the corresponding feature weights.

Hyperparameter search and software implementation We reimplemented the approach as described in [Xia et al., 2004], which does not introduce hyperparameters. A hyperparameter search was therefore not necessary.

S3.6 Similarity Ensemble Approach (SEA)

Similarity measure For comparability with other results, we used the same Tanimoto kernel function as for SVMs and K-nearest neighbour.

Hyperparameter search and Software implementation The approach introduces a correction model for which the parameters are optimized on a background set. The procedure for determining these parameters was described by the authors. In particular, a threshold for cutting off the Tanimoto similarity had to be determined. This hyperparameter selection as well as the SEA prediction procedure and the correction model were reimplemented to be used on a multi-core supercomputer as described in [Keiser and Hert, 2009], since this description seemed to be more precise to us than the descriptions in [Keiser et al., 2007].

S4 Results

S4.1 Method Performance Comparison

We performed Wilcoxon signed rank tests between all pairs of algorithms for target prediction to determine, whether some methods significantly outperformed others. In Table S9 we exemplarily show the results (p-values) of the tests, where we fixed the input feature category for algorithms, that rely on compound descriptors, either to ECFP6 or a combination of ECFP6 and ToxF (ECFP6+ToxF).

ECFP6

>	FNN	SVM	RF	KNN	NB	SEA	GC	Weave	LSTM
FNN	1.000E+000	1.985E-007	8.491E-088	4.814E-128	3.882E-122	4.538E-167	5.519E-083	4.613E-130	3.154E-069
SVM	1.000E+000	1.000E+000	1.549E-090	1.002E-118	4.433E-102	4.376E-170	2.034E-048	4.057E-094	6.591E-035
RF	1.000E+000	1.000E+000	1.000E+000	2.109E-011	6.830E-017	1.278E-115	1.000E+000	1.154E-005	1.000E+000
KNN	1.000E+000	1.000E+000	1.000E+000	1.000E+000	5.437E-006	2.084E-109	1.000E+000	6.510E-001	1.000E+000
NB	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.356E-109	1.000E+000	1.000E+000	1.000E+000
SEA	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.000E+000	1.000E+000
GC	1.000E+000	1.000E+000	1.524E-008	2.190E-029	7.580E-039	7.017E-142	1.000E+000	7.243E-034	9.999E-001
Weave	1.000E+000	1.000E+000	1.000E+000	3.490E-001	1.634E-005	3.423E-109	1.000E+000	1.000E+000	1.000E+000
LSTM	1.000E+000	1.000E+000	1.087E-016	2.143E-045	1.476E-044	8.271E-146	1.294E-004	1.310E-056	1.000E+000

ECFP6+ToxF

>	FNN	SVM	RF	KNN	NB	GC	Weave	LSTM
FNN	1.000E+000	8.777E-011	2.759E-033	5.151E-140	2.892E-114	5.125E-101	3.286E-142	2.504E-097
SVM	1.000E+000	1.000E+000	3.249E-021	2.716E-132	3.214E-093	9.519E-073	1.887E-109	5.453E-062
RF	1.000E+000	1.000E+000	1.000E+000	1.066E-094	1.092E-066	2.606E-032	1.755E-074	4.278E-021
KNN	1.000E+000	1.000E+000	1.000E+000	1.000E+000	2.189E-003	1.000E+000	1.231E-003	1.000E+000
NB	1.000E+000	1.000E+000	1.000E+000	9.978E-001	1.000E+000	1.000E+000	8.926E-001	1.000E+000
GC	1.000E+000	1.000E+000	1.000E+000	8.724E-017	9.296E-021	1.000E+000	7.243E-034	9.999E-001
Weave	1.000E+000	1.000E+000	1.000E+000	9.988E-001	1.074E-001	1.000E+000	1.000E+000	1.000E+000
LSTM	1.000E+000	1.000E+000	1.000E+000	1.300E-030	1.015E-027	1.294E-004	1.310E-056	1.000E+000

Table S9: p-values from a Wilcoxon signed rank test with the alternate hypothesis that algorithms in the row have a higher AUC than the algorithms in the column, using either ECFP6 feature encoding or a combination of ECFP6 and ToxF features for algorithms, that rely on compound descriptors

S4.2 AUC vs. Training and Test Set Sizes

We examined a matrix of scatter plots (Figure S4) to investigate the dependencies between training set size, test set size and the AUC values obtained by FNNs on ECFP6+ToxF features. Since training and test set sizes seem to correlate well, we conclude that the most likely cause for larger (smaller) AUC values is a larger (smaller) training set size.

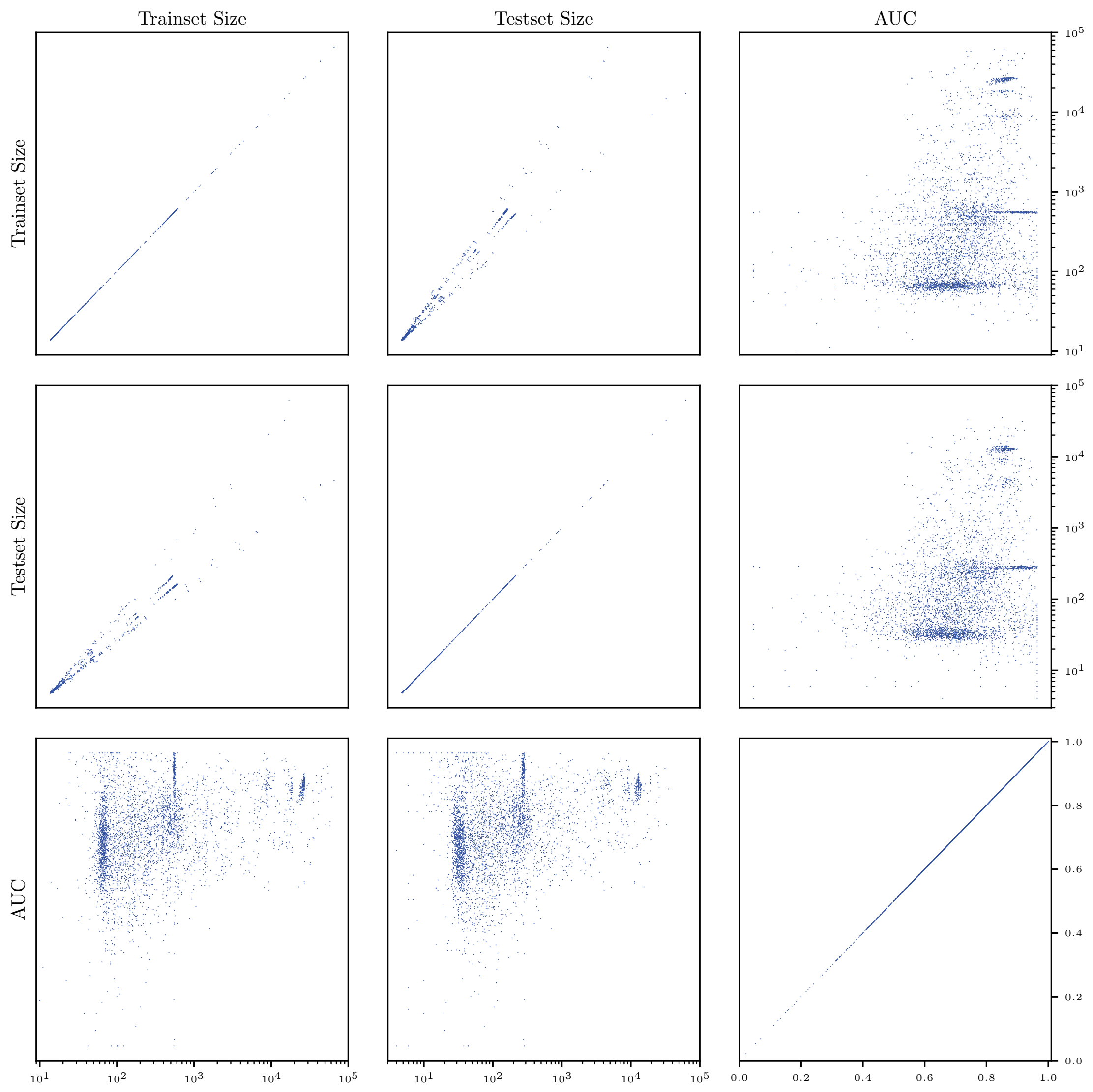

Figure S4: Correlation of training set size with test set size and correlation with AUCs from FNNs applied to ECFP6+ToxF features

S5 Appendix

S5.1 External Tables

Table S10: ChEMBL targets used and fold sizes

The table is attached as an external file: `Supplementary_Table_S10.csv`
Each row represents a ChEMBL target with the ChEMBL identifier in the first column and information on the corresponding training and test set sizes of the folds for each class (active/inactive) in the columns.

Table S11: FNN assay-AUC Performance for ChEMBL Assays and Feature Categories

The table is attached as an external file: `Supplementary_Table_S11.csv`
Each row represents a ChEMBL target with the ChEMBL identifier in the first column and the prediction performance (assay-AUC: mean AUC over folds) per feature category in the corresponding columns.

Table S12: FNN ROC-AUC Performance for ChEMBL Assays and Feature Categories

The table is attached as an external file: `Supplementary_Table_S12.csv`
Each row represents a ChEMBL target with the ChEMBL identifier in the first column and the prediction performance (ROC-AUC) per feature category and fold in the corresponding columns.

Table S13: FNN PR-AUC Performance for ChEMBL Assays and Feature Categories

The table is attached as an external file: `Supplementary_Table_S13.csv`
Each row represents a ChEMBL target with the ChEMBL identifier in the first column and the prediction performance (PR-AUC) per feature category and fold in the corresponding columns.

Table S14: Selected Assay Pairs, where one assay was described to be a high-throughput assay and the other was not explicitly described to be a high-throughput assay

The table is attached as an external file: `Supplementary_Table_S14.csv`

Table S15: Manually removed Assay Pairs, where one assay was described to be a high-throughput assay and the other was not explicitly described to be a high-throughput assay

The table is attached as an external file: `Supplementary_Table_S15.csv`

References

Democratizing Deep-Learning for Drug Discovery, Quantum Chemistry, Materials Science and Biology. `https://github.com/deepchem/deepchem`, 2016.

Pierre Broto, Gilles Moreau, and Corinne Vandycke. Molecular structures: perception, autocorrelation descriptor and SAR studies. Autocorrelation descriptor. *European Journal of Medicinal Chemistry*, 19(1):66–70, 1984.

Frank R Burden. Molecular Identification Number for Substructure Searches. *Journal of Chemical Information and Computer Sciences*, 29(3):225–227, 1989.

Dong-Sheng Cao, Qing-Song Xu, Qian-Nan Hu, and Yi-Zeng Liang. ChemoPy: freely available python package for computational biology and chemoinformatics. *Bioinformatics*, 29(8):1092–1094, 2013.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Advances in Neural Information Processing Systems 28*, pages 2224–2232, 2015.

Robert C Geary. The Contiguity Ratio and Statistical Mapping. *The Incorporated Statistician*, 5 (3):115–146, 1954.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *AISTATS*, pages 315–323, 2011.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep Learning*, volume 1. MIT press Cambridge, 2016.

Elisabet Gregori-Puigjané and Jordi Mestres. SHED: Shannon Entropy Descriptors from Topological Feature Distributions. *Journal of Chemical Information and Modeling*, 46(4):1615–1622, 2006.

Lowell H Hall and Lemont B Kier. The Molecular Connectivity Chi Indexes and Kappa Shape Indexes in Structure-Property Modeling. In Kenny B. Lipkowitz and Donald B. Boyd, editors, *Reviews in Computational Chemistry*, pages 367–422. VCH Publishers, 1991.

Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. InChI - the worldwide chemical structure identifier standard. *Journal of Cheminformatics*, 5(1):7, 2013.

Markus C Hemmer, Valentin Steinhauer, and Johann Gasteiger. Deriving the 3D structure of organic molecules from their infrared spectra. *Vibrational Spectroscopy*, 19(1):151–164, 1999.

Georg Hinselmann, Lars Rosenbaum, Andreas Jahn, Nikolas Fechner, and Andreas Zell. jCompoundMapper: An open source Java library and command-line tool for chemical fingerprints. *Journal of Cheminformatics*, 3(1):1–14, 2011.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design*, 30 (8):595–608, 2016.

Michael J. Keiser and Jérôme Hert. Off-Target Networks Derived from Ligand Set Similarity. In Edgar Jacoby, editor, *Chemogenomics*, pages 195–205. Humana Press, 2009.

Michael J. Keiser, Bryan L. Roth, Blaine N. Armbruster, Paul Ernsberger, John J. Irwin, and Brian K. Shoichet. Relating protein pharmacology by ligand chemistry. *Nature Biotechnology*, 25(2):197–206, 2007.

Lemont B Kier and Lowell H Hall. An Electrotopological-State Index for Atoms in Molecules. *Pharmaceutical Research*, 7(8):801–807, 1990.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 972–981, 2017.

Gilles Louppe. Understanding Random Forests: From Theory to Practice. *arXiv preprint arXiv:1407.7502*, 2014.

Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. DeepTox: Toxicity Prediction using Deep Learning. *Frontiers in Environmental Science*, 3(80), 2016.

Patrick AP Moran. Notes on Continuous Stochastic Phenomena. *Biometrika*, 37(1–2):17–23, 1950.

Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. How Many Trees in a Random Forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer, 2012.

RDKit. RDKit: Open-source cheminformatics and machine learning. `http://www.rdkit.org`, 2016. [Online; accessed 18-August-2016].

Steffen Renner, Uli Fechner, and Gisbert Schneider. Alignment-free Pharmacophore Patterns – A Correlation-vector Approach. In *Pharmacophores and Pharmacophore Searches*, pages 49—-79. Wiley, 2006.

David Rogers and Mathew Hahn. Extended-Connectivity Fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010.

Gisbert Schneider, Werner Neidhart, Thomas Giller, and Gerard Schmid. "Scaffold-Hopping" by Topological Pharmacophore Search: A Contribution to Virtual Screening. *Angewandte Chemie International Edition*, 38(19):2894–2896, 1999.

Jan H Schuur, Paul Selzer, and Johann Gasteiger. The Coding of the Three-Dimensional Structure of Molecules by Molecular Transforms and Its Application to Structure-Spectra Correlations and Studies of Biological Activity. *Journal of Chemical Information and Computer Sciences*, 36(2):334–344, 1996.

David T Stanton and Peter C Jurs. Development and Use of Charged Partial Surface Area Structural Descriptors in Computer-Assisted Quantitative Structure-Property Relationship Studies. *Analytical Chemistry*, 62(21):2323–2329, 1990.

S. Joshua Swamidass, Jonathan Chen, Jocelyne Bruand, Peter Phung, Liva Ralaivola, and Pierre Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21(Suppl 1):i359–i368, 2005.

Roberto Todeschini and Paola Gramatica. 3D-modelling and prediction by WHIM descriptors. Part 5. Theory development and chemical meaning of WHIM descriptors. *Quantitative Structure–Activity Relationship*, 16(2):113–119, 1997.

Roberto Todeschini, Marina Lasagni, and Emilio Marengo. New molecular descriptors for 2D and 3D structures. Theory. *Journal of Chemometrics*, 8(4):263–272, 1994.

David Weininger. SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.

Marvin N Wright and Andreas Ziegler. ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *arXiv preprint arXiv:1508.04409*, 2015.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science*, 9(2):513–530, 2018.

Xiaoyang Xia, Edward G. Maliski, Paul Gallant, and David Rogers. Classification of Kinase Inhibitors Using a Bayesian Model. *Journal of Medicinal Chemistry*, 47(18):4463–4470, 2004.