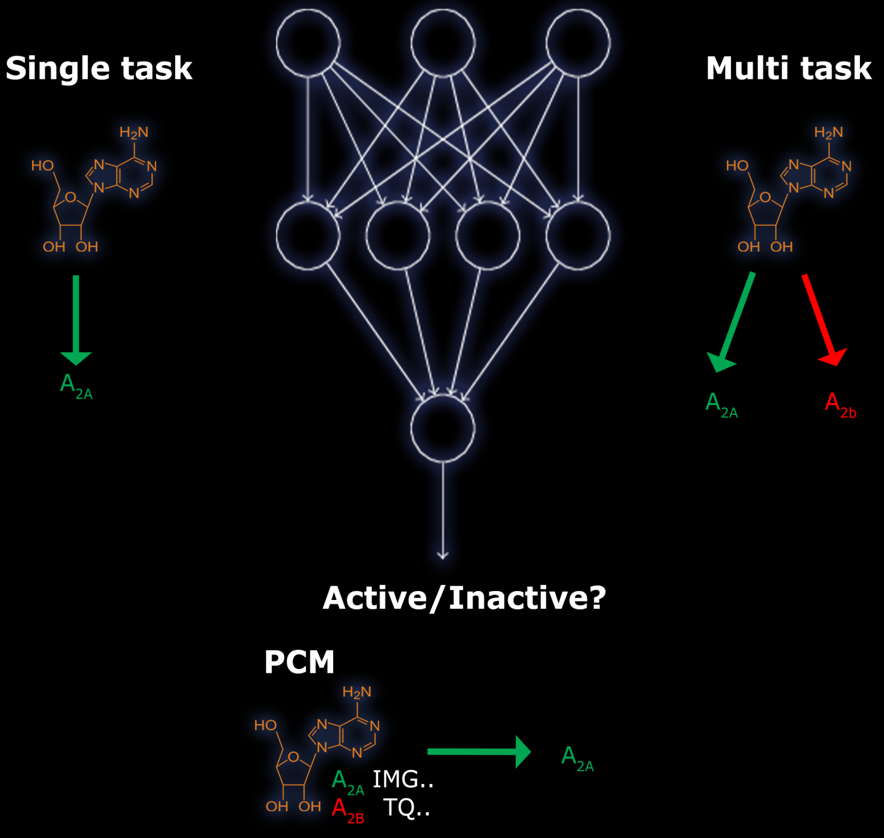
Chapter 6
Beyond the Hype: Deep Neural Networks Outperform Established Methods Using A ChEMBL Bioactivity Benchmark Set



Adapted from:
"Beyond the Hype: Deep Neural Networks Outperform Established Methods Using A ChEMBL Bioactivity Benchmark Set." Eelke B. Lenselink, Niels ten Dijke, Brandon Bongers, George Papadatos, Herman W.T. van Vlijmen, Wojtek Kowalczyk, Adriaan P. IJzerman, and Gerard J.P. van Westen.
Manuscript in preperation

Chapter 6

Docking and virtual screening strategies for GPCR drug discovery

Contents

About this chapter

The increase of publicly available bioactivity data in recent years has fueled and catalyzed research in chemogenomics data mining and modeling approaches. As a direct result, over the past few years a multitude of different methods has been reported and evaluated, such as target fishing, nearest neighbor similarity-based methods, and QSAR-based protocols. However, such studies are typically conducted on different datasets, using different validation strategies, and different metrics. Moreover, multiple algorithms have been published leading to slightly different results. Hence we aimed to compare different methods, specifically QSAR and PCM methods, using one single standardized dataset obtained from ChEMBL. We validated different methods using both a random and a temporal validation with the latter being a more realistic benchmark of expected prospective execution. The performance of Random Forests (RF), Deep Neural Networks (DNN), and Naive Bayes (NB) models using binary class, multiclass, and proteochemometric methods was assessed. Overall, we observed that DNN are the top performing algorithm, as it was best in terms of prediction quality and runtime. Further tuning and using an ensemble of DNN enhanced the performance by another 27%, highlighting the future applicability of DNN in addition to other more conventional methods. By providing both the data and the protocols we offer a standardized set to test different machine learning algorithms in the context of multitask learning.

Introduction

The amount of chemical and biological data in the public domain has exploded over the last decades. With the advent of ChEMBL computational drug discovery in an academic setting has been revolutionized;[1,2] indeed, the amount of data available in ChEMBL is also growing rapidly. Yet data availability and data quality still pose limitations.[3] Public data is sparse (on average 1 compound is tested on 2 proteins) and prone to error (on average 0.5 log units for IC_{50} data)[3,4] To make full use of the potential of this sparse data and study ligand-protein interactions on a proteome wide scale, computational methods are indispensable as they can be used to predict bioactivity values of compound-target combinations that have not been tested experimentally.[5–7]

Activity threshold and data quality

The modelling in a classification based approach requires setting of activity thresholds wherein a preset cutoff decides what is 'active' and what 'inactive'. Data gathered here contains pChEMBL values, which represent comparable measures of concentrations to reach half-maximal response / effect / potency / affinity transformed to a negative logarithmic scale. The threshold for 'actives' determines the fraction of data points belonging to the 'active' class. If the threshold is set at 10 µM (pChEMBL = 5) as is done frequently in literature, almost 90% of the extracted ChEMBL data is 'active' making it the default state.[7,9] Such a high fraction of actives is not in accordance with what is observed experimentally. Moreover, in an experimental context, model output should ideally lead to identification of compounds with affinity higher than 10 µM. Based on these considerations we chose to set the activity threshold at 6.5 log units (approximately 300 nM), defining interactions with a log affinity value larger than 6.5 as 'active'. At this threshold the distribution between 'actives' and 'inactives' is roughly 50%, and the modelling aim is more in line with experimental goals.

However, as was touched upon above, public data can have relatively large measurement errors, mostly caused by the data being generated in separate laboratories by different scientists at different points in time with different assay protocols. To make sure that bioactivity models are as reliable as possible, we chose to limit ourselves to the highest data quality available in ChEMBL (see methods for details).

Bioactivity modeling is a multi-label problem

As a consequence of the 'active' and 'inactive' definition, compounds can be active on multiple proteins.[10] This can be modeled using (ensembles of) binary class estimators, for instance by combining binary class RF models (Figure 6.1). Another strategy is to assemble one model with all targets, which can be done in various ways. With multiclass QSAR we can predict if a compound is active based on the probability of belonging to the active target class versus the inactive target class; each compound-target combination is assigned as active or inactive (Figure 6.1). Another approach is to apply machine learning algorithms with added protein descriptors which is commonly known as PCM.[11,12] In essence, PCM transforms the multiclass approach into a binary class approach by addition of protein descriptors (Figure 6.1). Explicitly quantifying this protein similarity allows models to make predictions for targets with no data or very little data but for which a sequence is known. Moreover, explicit protein features allow interpretation of both the important protein and ligand features from the validated model. The relationships between structure and biological activity in these large pooled datasets are non-linear and best modelled using non-linear methods such as RF or Support Vector Machines (SVM). Alternatively, when for instance PLS is used, cross-terms are required that account for the non-linearity.[13] Another non-linear method, deep neural networks (DNN) have recently gained traction, being successfully applied to a variety of artificial intelligence tasks such as image recognition, autonomously driving cars and the GO-playing program AlphaGO.[14,15] Multiclass DNN explicitly model every target by assigning one output node to a target (Figure 6.1). Given the relative novelty of DNN we will further introduce the method below as opposed to the more established RF and NB methods.

Deep neural nets in bioactivity modeling

DNN have been applied to model bioactivity data previously; in 2012 Merck launched a challenge on Kaggle to build QSAR models for 15 different tasks.[16] The winning solution contained an ensemble of single-task deep neural networks (DNN), multi-task DNN, and Gaussian process regression models. The multi-task neural networks modeled all 15 tasks simultaneously, which were later discussed in the corresponding paper.[16] Later (multi-task) DNN have also been applied on a larger scale to 200 different targets,[17] tested in virtual screening,[18] and was one of the winning algorithms of the Tox21 competition.[19] Hence, DNN have shown to be an algorithm with great potential in bioactivity modeling and has been included in the current work as this technique may become the algorithm of choice in both PCM and QSAR.

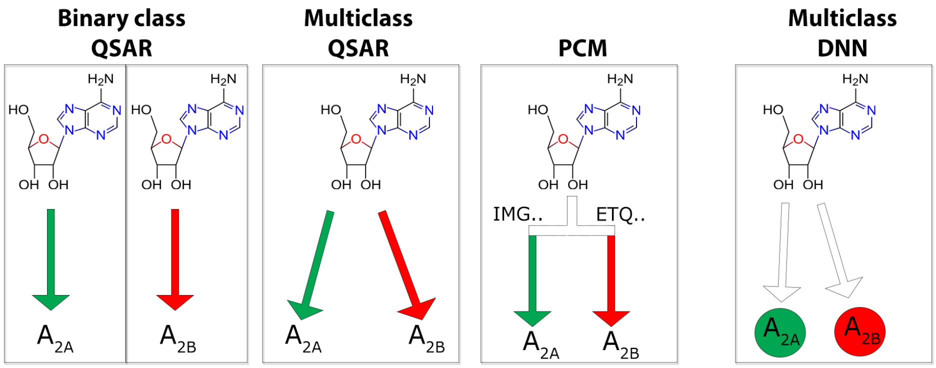


Figure 6.1: Differences between the methods in modeling bioactivity data, exemplified by the ligand adenosine which is more active (designated as 'active') on the adenosine A2A receptor, than on the A2B receptor ('inactive', using PChEMBL > 6.5 as a cutoff). With binary class QSAR, individual models are constructed for every target. With multiclass QSAR one model is constructed based on the different target labels (A2A_active, A2B_inactive). With PCM one model is constructed where the differences between proteins are considered in the descriptors (i.e. based on the amino acid sequence). With multiclass DNN one output node are explicitly assigned for every target.

Established algorithms in bioactivity modeling

DNN algorithms come within a large variety of existing open-source and closed-source packages. A commonly used toolkit in pharma companies is Pipeline Pilot (PP) with R-statistics.[20,21] However, software libraries such as RDKit for cheminformatics and scikit-learn for machine learning in Python (PY) are increasingly gaining ground.[22,23] Publication of these (open source) algorithms have heralded open source drug discovery. In addition to this increase in popularity, they have also been included in the myChEMBL virtual machine released by the ChEMBL group at EBI, and as such are universally available on a standalone, easy-to-use platform, t

hus making their inclusion in our benchmark a requirement.[24] Here we choose to apply RF and NB as these methods are arguably some of the most commonly used methods, RF for its superb accuracy, NB for its simplicity.

Aims

As touched upon above, many computational methods and studies rely on data from ChEMBL. Yet very few of these studies have used the highest quality of ChEMBL data, a common misconception being that confidence class is interpreted as quality *quantification* rather than *classification*. Furthermore, most studies evaluate various machine learning algorithms but do not test more fundamental differences, such as addition of protein information. Finally, in some cases validation methods differ between different techniques.

Here we aim to address these caveats by performing a systematic study on a high quality ChEMBL dataset. We compare QSAR and PCM methods, multiple algorithms (including DNN), the differences between binary class and multiclass models, and usage of temporal validation (effectively validating true prospective use) (Table 6.1). For this we used both open- and closed-source software and provide the dataset, descriptors and scripts in the Supplementary Information and on FTP. We make direct recommendations on which technique to use for public data modeling. In this way, the current work contributes to the literature by providing not only a standardized dataset, but also a realistic estimate of the performance that published methods can currently achieve in preclinical drug discovery.

Methods

Hardware

Experiments were performed on a Linux server running Centos 6.7. The server was equipped with dual Xeon E5-2620v2 processors and 128 GB RAM. GPUs installed are a single NVIDIA K40 and 2 NVIDIA K20s.

Software used

In this study, multiple, different software libraries and packages were compared. As closed source we used the commonly used software; Pipeline Pilot (version 9.2.0),[20] including the chemistry collection for calculation of descriptors, and R-statistics (R version 3.1.2) collection for the machine learning.[21]

The open-source software Python (version 2.7) was used with the following libraries: RDKit (version 2014.09.2)[22] for the calculation of the fingerprints and descriptors, scikit-learn version 0.16 for the NB and RF.[23] For the neural networks we used Theano,[31] and nolearn, together with Lasagne which were pulled from GitHub.[32–34]

Table 6.1: Overview of the tested methods

Abbreviation	Software	Classes	Machine Learning
PP NB MC-QSAR	Pipeline Pilot	Multiclass	NB
PP RF QSAR	Pipeline Pilot / R	Binary class	RF
PP RF MC-QSAR	Pipeline Pilot / R	Multiclass	RF
PP RF PCM	Pipeline Pilot / R	Binary class	RF
PY NB MC-QSAR	RDkit / Python	Multiclass	NB
PY RF QSAR	RDkit / Python	Binary class	RF
PY RF MC-QSAR	RDkit / Python	Multiclass	RF
PY RF PCM	RDkit / Python	Binary class	RF
PY DNN QSAR	RDkit / Lasagne	Binary class	DNN
PY DNN MC-QSAR	RDkit / Lasagne	Multiclass	DNN
PY DNN PCM	RDkit / Lasagne	Binary class	DNN

Abbreviations: NB - Naive Bayes, RF - Random Forest, DNN - Deep Neural Nets, QSAR - Quantitative Structure Activity Relationship, PCM - Proteochemometric modeling, MC- Multiclass

Dataset

Data was obtained from the ChEMBL database (version 20), [1] containing 13,488,513 data points. Activities were selected that met the following criteria: at least 10 compounds tested per protein, assay confidence score of 9, 'single protein' target type, assigned pCHEMBL value, no flags on potential duplicate or data validity comment, originating from scientific literature.

For multiple ligand-receptor data points the median value was chosen and duplicates were removed. This reduced the total amount of data points to 341,517 data points, or approximately 2.5% of the total data in ChEMBL 20.

Typically studies have used thresholds for activity between 5 and 6.[7,9,25,26] We assigned data points to the 'active' class if the pCHEMBL value was above 6.5 (corresponding to approximately 300 nM) and to the 'inactive' class if the pCHEMBL value was equal to or below 6.5 for two reasons. First of all this threshold gave a good ratio between actives and inactives. Around 90% of the data points is active when a threshold of 10 μM is used, while a roughly equal partition occurs at a threshold of 300 nM. Secondly it represents an activity threshold that is more relevant for biological activity.

The final set consisted of 1,227 targets, 204,085 compounds, and 314,767 data points (0.13% complete). In total the set contained 70,167 Murcko scaffolds.[35] The set was divided in 15 L1 target classes, 34 L2 target classes, and 91 L3 targets classes.

Compound Descriptors

For every compound the following physicochemical descriptors were calculated: Partition Coefficient (AlogP),[36] Molecular Weight (MW), Hydrogen Bond Acceptors and Donors (HBA/HBD), Fractional Polar Surface Area (Fractional PSA),[37,38] Rotatable Bonds (RTB). In addition, circular fingerprints were also included, with slight differences between PP and RDKit (Python) implementation. Previous work showed this descriptor selection to be high performing across diverse target classes.[39] The latter uses RDKit Morgan fingerprints, with a radius of 3 bonds and a length of 256 bits, while in the former Extended Circular FingerPrints with the same radius and length (ECFP_6) were chosen.[22,40] According to Landrum the RDkit implementation of circular fingerprints is very similar but not identical as was demonstrated in the RDkit user group meeting.[41]

Protein Descriptors

For the Proteochemometric models, protein descriptors were calculated, which are based on physicochemical properties of amino acids as has been described previously.[42,43] Lacking the ability to align all proteins, descriptors were made alignment independent and were thus generalized to all targets in ChEMBL in the following way. The protein was split into 20 equal parts (where part length differed based on protein length). Per part, for every amino acid the following descriptors were calculated: Amount of stereo atoms, LogD,[36] charge, HBA and HBD, rigidity, aromatic bonds, MW. Subsequently per part the mean value for each descriptor of the contained amino acids was calculated. The method was repeated for the whole protein, calculating the mean value for the full sequence length. Finally, sequence length was included as separate descriptor.

Machine Learning – Naive Bayes classifier

In PP, a NB categorical model was trained using 10 bins, both the active (pChEMBL >6.5) and inactive data points were used by relabeling targets to active and inactive target classes. For every target class NB scores were transformed into z-scores (compared to the mean score of actives and the mean score of inactives for a given target).[44,45] Subsequently z-scores of the inactive class were subtracted from the active class yielding one z-score for every target. Finally, data points with z-scores higher than 0 were assigned to the active class of a target and lower than 0 to the inactive class. In scikit-learn, the Naive Bayes models were trained using the same procedure and MultinomialNB.[46]

Machine Learning – Random Forest

RF trained in PP used R package randomForest, as has been done before.[47] The following settings were used: 1000 trees, 30% of the features were randomly selected to compute the best split point, with no limit on the maximum depth of the tree. For scikit-learn the same settings were used except for the multiclass models where the depth of the tree was set to 10 due to memory limitations (>120 GB memory usage). For the multiclass RF, a probability of each class (both active/inactive) was calculated for each entry. The highest probability was chosen as the predicted label and compared to actual experimental label (e.g. for adenosine: A_{2A}_active, A_{2B}_inactive).

Machine Learning – Neural Networks

Given the novelty of DNN, they are elaborately described in this section below. Neural networks were accelerated on GPUs using nolearn/Lasagne and Theano packages.[31,33,34] Feedforward neural networks were constructed using techniques similar to previous studies.[16] Models were trained using batch gradient descent with Nesterov momentum using a batch size of 128.[48] For both the learning rate and Nesterov momentum we used an adaptive version: the starting Nesterov momentum was set to 0.8 and 0.999 for the last epoch. The learning rate for the first epoch was set to 0.005 and 0.0001 for the last epoch. The networks consisted of the input layer (e.g. 256 fingerprints) connected to 3 layers of 4000, 2000, 1000 rectified linear units (ReLu) and a linear output layer. The linear output layer consisted of the number of targets modelled (e.g. 1227 for the multi-task network). Because a linear output was used, pChEMBL values were predicted, which were subsequently converted to classes (pChEMBL > 6.5 = active, pChEMBL ≤ 6.5 = inactive). The target protein features were scaled to zero mean and unit variance. To prevent overfitting of the networks, we used 25% dropout on the hidden layers together with early stopping.[49] The early stopping validates the loss on an evaluation set (20% of the data) and stops training if the network does not improve on the evaluation set after 200 epochs. The maximum number of iterations was set to 2000 epochs. For the multi-task QSAR models the output layer consists of one output node for each target. The output for a particular compound is going to be sparse, i.e. for most targets there will be no known activity. During training, only targets for which we have data were taken into account when computing the error function to update the weights. We chose to weigh each target, for which we had data, in the error function equally.

For the DNN optimization, we used 500 epochs of training to speed up the training. In total, 63 individual models were trained to validate the influence of different descriptors, architecture and type of neural network. Due to problems with the stochastic gradient descent, for the PCM models with 4096 fingerprints plus physicochemical chemical properties, architectures 2, 4, 6; a batch size of 256 instead of 128 was used. In all cases where physicochemical chemical properties were used, they were scaled to zero mean and unit variance.

Data formatting – Binary and Multiclass

For binary class implementations simple 'active' and 'inactive' classes were used as described above. The multiclass implementation consisted of a concatenation of the binary class and protein modeled as described by Uniprot accession (e.g. 'active_P29274' and 'inactive_P29274'). In the MC DNN the active and inactive data points were modeled by one output node for every target simultaneously.

Validation Metrics

We used the Mathews Correlation coefficient (MCC) as a primary validation metric, because this metric is considered to be a good metric for unbalanced classes.[39,50] Two separate MCCs were calculated. One value for the pooled predictions (pooling true positives, false positives, true negatives, and false negatives) was calculated, and secondly an average MCC was calculated per protein. Of these two values the mean is visualized in figures 6.1 and 6.2.When no predictions were made for a given target-compound combination a random number was generated for this pair between 0 and 1 . The reason for this is that we aimed to simulate a true use case and not cherry pick good or bad predictions. To be able to compare prediction quality across the different methods used random values were used, leading to MCC scores close to 0 for these cases. For values > 0.5 this score was deemed 'active' and anything below 0.5 was deemed 'inactive'.Finally, for a selected number of methods (DNN PCM, NB MC-QSAR, and RF MC-QSAR) we also plotted an ROC curve and calculated the AUC for a proper comparison with other published work.

Validation Partitioning

Two different methods were applied to partition the data in training/validation sets. The first method that was used was a 'random split', where data was partitioned using semi-stratified random partitioning with a fixed seed as implemented in PP.[20] For the second method was a separate set was constructed wherein the year of the publication was the split criterion. All data points originating from publications that appeared prior to 2013 were used in the training set, while newer data points went into the validation set.

To further explore the differences between random split validation and year split validation we explored the amount of compounds, scaffolds, and targets shared in the training and test set. In the random split validation 29,368 compounds (22,945 scaffolds) were shared in training and test set; 44,048 compounds (11,166 scaffolds) were unique in the test set; and 117,790 compounds (33,005 scaffolds) unique in the training set. In the year split, 1,216 compounds (2,157 scaffolds) were shared in training and test set; 36,693 compounds (12,646 scaffolds) were unique in the test set; and 165,037 compounds (55,364 scaffolds) unique in the training set. Likewise in the random split validation set 1,223 targets were shared in both, 3 targets were unique to the training set, and there were no unique targets in the test set. In the year split 902 targets were shared in training and test set, 22 were unique to the test set, and 303 were unique to the training set.

Results and discussion

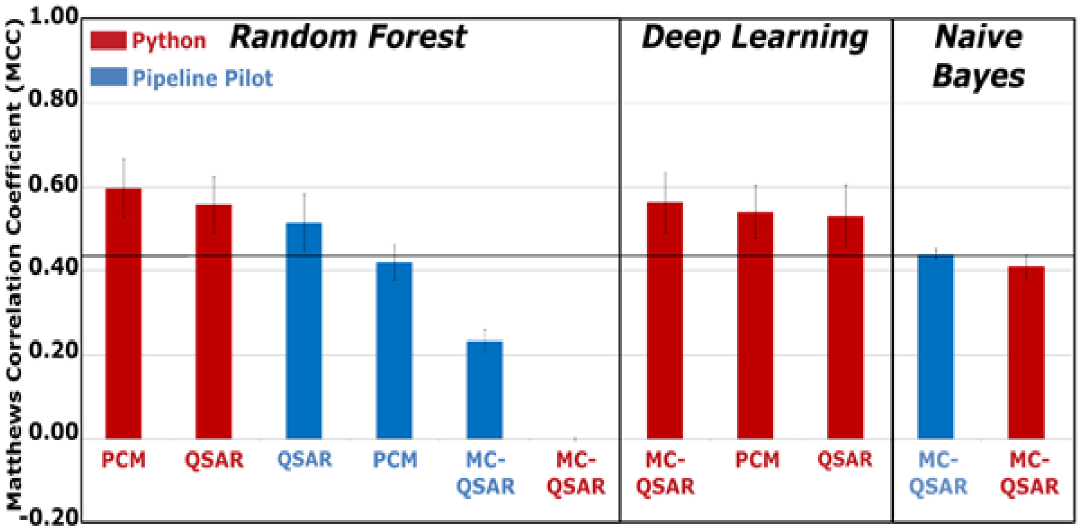


Figure 6.2: Performance of the different methods in the random split validation, grouped by method and colored by software used. Shown are the mean and standard error of the mean of the Matthews Correlation Coefficient (MCC) calculated over all external validation predictions and combined with the MCC calculated per protein. The black line represents the average MCC of all methods used here (0.44). PCM stands for Proteochemometric modeling, MC-QSAR for multiclass QSAR, and QSAR for Quantitative Structure Activity Relationship.

Random split partition

Firstly we tested all algorithms using random split partitioning given that it is one of the most common methods to perform validation of machine learning (Figure 6.2). We trained the models on 70% of the data and validated them on the remaining 30%. This validation of the classifier predictions on a multi-target data set such as this one can be done in several ways, the most common methods being based on all predictions in a single confusion matrix or by calculating a confusion matrix per target and subsequently using the mean value. Both methods provide relevant information, thus we followed both and show the mean and standard error of the mean obtained from these two calculations (with original data for both in The average Matthews Correlation Coefficient (MCC) of all algorithms is 0.44, underlining the predictive power of most methods. Although in theory results between the same algorithms should be the same, small differences in PY and PP implementations of RF and NB exist, explaining the observed differences in MCC. Focusing on our reference method RF, which we applied in previous work, we observe that the regular QSAR (binary class, BC) models outperform the MC-QSAR (multiclass, MC) RF models. The most predictive model is the PCM model in Python with an MCC of 0.60. Conversely, both RF MC-QSAR models show poor performance with the worst case being the MC-QSAR model in Python (MCC of 0.00). Hence, based on the MCC this method actually performs on par with random guessing, yet the Area under curve (AUC) for the Receiver Operator Curve (ROC) is still 0.60. Poor performance of the RF MC-QSAR models is likely caused by the very small class sizes resulting from both the sparseness of the data and the multiclass setup combined with active/inactive distinction. This hypothesis is confirmed by the much better performance of the PCM models that combine the benefits of a binary class setup with the use of multiple proteins in the form of descriptors. Indeed, due to the sparseness only a minority of predictions is made by the RF MC-QSAR models (7-21% of total), demonstrating that these models only retrieved the minority of measured targets. For these predictions it might very well be that a target for which the activity is predicted (either active or inactive), is a correctly predicted target. However typically if this prediction is not present in ChEMBL we cannot validate if this compound-target combination is an active or inactive from our data. We wanted to compare predictions across all techniques and not suffer from large differences in total number of predictions and resulting statistical bias. For instance, it could be that only the 'easy extrapolations' (e.g. nearest neighbors) are retrieved by the different methods, leading to an overly optimistic score.

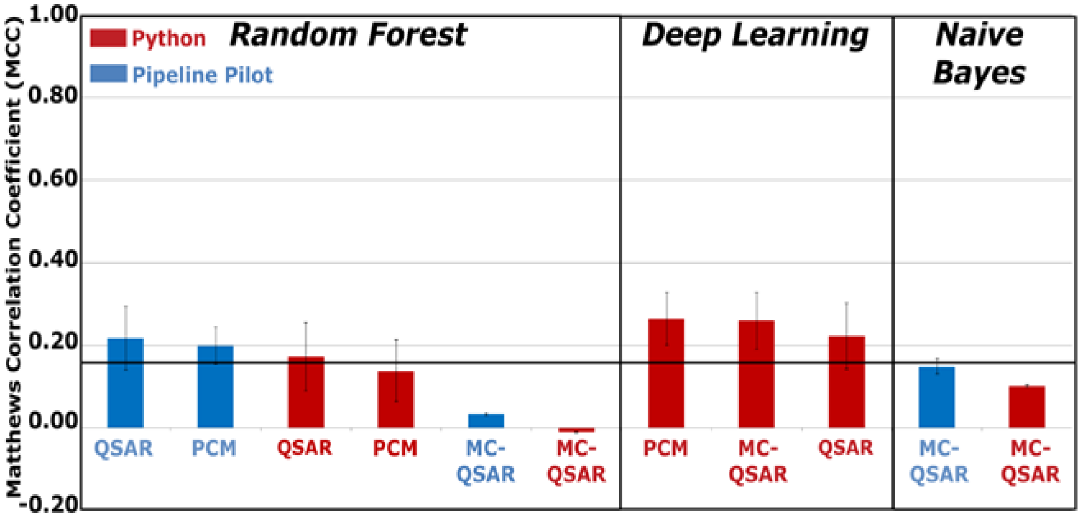


Figure 6.3: Performance of the different methods in the temporal split validation, grouped by method and colored by software used. Overall performance is worse than the random splits with an average MCC of 0.16 (black line). In this validation DNN outperform the RF, with the PCM model being the best.

Hence, for missing predictions we have chosen to use a randomly generated value between 0 and 1, as in our use case failure to predict activity on the relevant target is not usable either. These random predictions would correspond to a mean MCC of 0, with negligible variance, as can be observed in the results.The performance of all DNN are well above the average performance, the best methods being both the MC-QSAR and QSAR DNN with a MCC of 0.56 (AUC 0.85). PCM is performing marginally worse than the QSAR based methods (MCC of 0.54). Finally, for the NB methods, performance is just above average. The PP method with a MCC of 0.44 (AUC 0.83) slightly outperformed the method in Python (MCC 0.41). In the random splitting used here all data points (measured activities of protein-compound combinations) are considered as separate entities. Hence, members of a compound series from a given publication can be part of the test set while the remaining are part of the training set (see methods - validation partitioning). Therefore this method is expected to give an unrealistic estimate of model performance; for a more representative performance estimate, a more challenging validation is exemplified below.

Temporal split partition

In the temporal split, training data was grouped by publication year rather than random partitioning (Figure 6.3). Again the mean MCC and standard error of the mean are shown. All methods performed worse than on the random split benchmark (average MCC 0.16

versus 0.44), confirming that this is indeed a more challenging form of validation.

Methods performing better than average include all three DNN methods, PP RF PCM, and PP/PY RF QSAR. Interestingly the DNN PCM were now the best performing method (MCC 0.26, AUC 0.73) together with the DNN MC-QSAR (MCC 0.26). Moreover, we found that for targets with few data points in the training set, the PCM models were able to extrapolate predictions, underlining the superior performance of DNN PCM.The RF models in Python performed slightly worse than the ones implemented in PP (MCC approximately 0.16 versus approximately 0.20). Likewise, as observed in the random split validation, all RF MC-QSAR models were not predictive with MCCs close to 0 (MCC for the PP RF MC-QSAR is 0.03, AUC 0.52). Indeed, significant differences were observed between the multiclass RF and DNN. For the NB models, performance was close to the average performance of all methods (MCC for the PP MC NB QSAR is 0.15, AUC 0.70).

The lower performance observed here is more in line with the performance that can be expected from a true prospective application of these types of models. However, in addition to raw performance, training time is also of great importance. Quick training models allow for easy retraining when new data becomes available, models that require a long time are not readily updated, making their maintenance a trade off.

Run time

To compare the performance of the different methods, we measured total run time (wall clock hours, Figure 6.4**)**. Much larger differences were observed between the methods than we noticed in the performance plots. Hence a logarithmic scale was used in figure 6.4. The fastest is the NB classifier in Python with a training time of only 20 minutes. Subsequently eight methods follow within one order of magnitude (time between 2 hours and 10 hours) including the different RFs with the exception of PP PCM (14 hours) and PP MC-QSAR (142 hours).The DNN models were also trained within 10 hours, with the DNN QSAR being the fastest (5 hours), followed by both the DNN MC-QSAR and DNN PCM (around 9 hours). The NB in PP was considerable slower than the NB trained in Python (20 minutes compared with 31 hours). Perhaps the bottleneck of the PP NB is the calculation of the background scores for the z-score calculation (see methods for details).

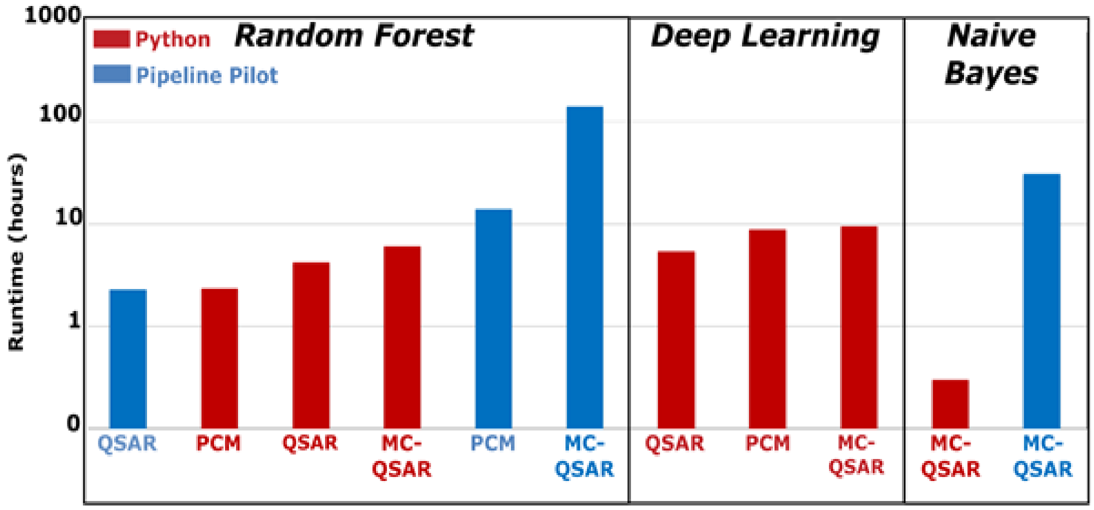


Figure 6.4. Comparison of the run times of the different methods. The Python implementation of NB classification is by far the fastest and the bulk of the methods took between 2 and 10 hours. The MC implementations of RF in Pipeline Pilot are trailing at over 140 hours.

Comparing these results with the random and year split validation, we observed that longer training times do not necessarily imply better model performance (MC models). Likewise, shorter training times do not always provide worse models (BC RF models). Finally, it should be noted that the GPU-implementation of the DNN speeds up the calculation about ~150 times when compared with the CPU-implementation (observed in the MC and benchmarked on a single core); this makes GPUs a definite requirement for the training of DNN.

Ranking the various methods

From the above sections we learned that there is a trade-off between model performance and training times. We calculated a z-score for each method within the experiments (random splits, year splits, and training time, Table 6.2). Herein DNN models are the best algorithm, and have the most consistent performance. For instance, for DNN the best model (PY MC-QSAR) has an average z-score of 0.70, compared to -0.10 for the best NB model and 0.51 for the best RF. Moreover, the three best methods based on the average z-score are all DNN, which are subsequently followed by RFs. Indeed this is because the performance of DNN in training time and in both of the validations is well above average.

When comparing QSAR with PCM there is no clear winner; for RF the performance of QSAR seems to be slightly higher especially when comparing both methods trained in PP (i.e. 0.51 versus 0.17). For DNN on the other hand PCM outperforms regular QSAR (binary class) but not MC-QSAR. NB methods perform just below the average performance, with the one implemented in PY being faster but worse in both the random and temporal validation.

The worst results are observed in the RF MC-QSAR models, both the one implemented in PY and in PP are the worst two methods tested here.Given the observed differences in performance, we wanted to investigate whether certain target classes (e.g. GPCRs) or scaffolds (e.g. xanthine) are modeled better using a certain method. This will be explored below.

Table 6.2: Overview of the performance of the benchmarked methods.

Algorithm	Method	Z-score random split	Z-score temporal split	Z-score training time	Average Z-score
RF	PY PCM	**0.90**	-0.23	**0.44**	0.37
	PY QSAR	0.68	0.15	0.40	0.41
	PP QSAR	0.44	**0.66**	0.44	**0.51**
	PP PCM	-0.09	0.46	0.16	0.17
	PP MC-QSAR	-1.15	-1.41	-2.95	-1.84
	PY MC-QSAR	-2.46	-1.91	0.35	-1.34
DNN	PY MC-QSAR	**0.71**	1.14	0.27	**0.70**
	PY PCM	0.60	**1.19**	0.28	0.69
	PY QSAR	0.52	0.71	**0.37**	0.53
NB	PP MC-QSAR	**0.02**	**-0.11**	-0.26	-0.12
	PY MC-QSAR	-0.15	-0.64	**0.49**	**-0.10**

Z-scores were calculated, Z-scores in bold indicate the best performance for a given machine learning algorithm column wise.

Per target class performance

The data in ChEMBL is organized in many ways, one of which is the target class hierarchy. Herein protein targets are grouped on different levels (e.g. level 1 (L1) being 'membrane receptor' or 'enzyme' and level 2 (L2) being 'family A GPCR' or 'kinase'). We explored the differences between the DNN based, RF based, and NB methods from a protein class point of view. Hence, we selected the best performing implementation for each on the temporal split (i.e. three methods with the highest Z-score of the temporal split, in bold, table 2). This led to the selection of the PCM implementation of the DNN models, the PP RF QSAR, and the PP NB MC-QSAR method. Performance estimates were assessed by recalculating the MCC grouped by L1 or L2 target class and using the temporal split. We limited the grouping to protein target classes with more than 100 proteins and the L1 and L2 proteins classes.

For the L1 target class-based groups DNN show the best performance which is most apparent in the 'membrane receptors' and 'transporter' classes (Figure 6.5A). Other points of interest are the poor performance of the NB model on the 'transporter' class, the MCC is worse than a random guess for this class. The cause for this is not immediately clear, but could possibly be explained by either the data sparseness or statistical dependence of the descriptors on each other. The ligands do not appear to be more diverse, to contain fewer actives, nor are different in any other way when compared with the other protein target classes. Finally, a performance slightly below average of the DNN in the 'ion channel' class was observed, yet still having an MCC of approximately 0.3.

For the L2-based grouping we see a similar trend (Figure 6.5B). The higher resolution afforded by the finer grained groups taught us that the ligand-gated ion channels are the causative factor for the performance drop of the DNN in the ion channels class. As this effect is also present in the QSAR based DNN models, the relatively simple protein descriptors used here cannot be the cause. Interestingly, performance in the voltage gated ion-channels is roughly equal. Similarly, we observed that the NB poor performance in L1 'transporters' is caused by the 'electrochemical transporters' L2 class.In conclusion, the performance gains of the DNN appear to be relatively consistent, however the 'ligand-gated ion channels' performance warrants caution. Perhaps these proteins are better modeled with target dedicated QSAR based methods.

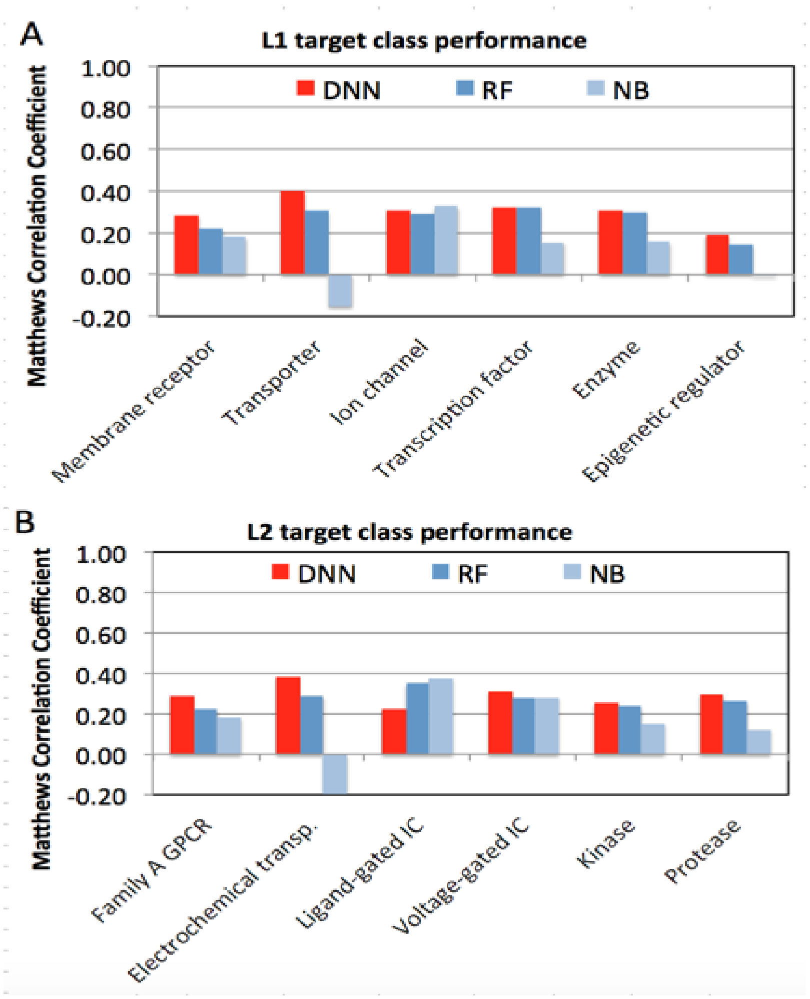


Figure 6.5: Performance of the different methods for the individual target classes in ChEMBL 20. In the L1 target classes (A) DNN outperform the other methods in all but one case, which is also true for the L2 target classes in (B). Only the 'ligand-gated ion channels' seem to be challenging for the DNN. Noteworthy is the poor performance of the NB models for the 'transporter' class. This is caused by the poor performance on the electrochemical transporters (a relatively large target class) as observed on the L2 level. Finally, the RF have a reasonable performance on all classes (but never the absolute best).

Exploring the potential of DNN

Given the same data set and descriptors we have shown that DNN methods generally outperform other algorithms. Therefore, we further explored the potential of this technique through a grid search based exploration of model parameter space (defined here by width and depth of the network) to obtain the best prediction quality. In addition, we explored the usage of more extensive compound descriptors (up to 4096 bits with additional physicochemical descriptors) which was not possible with the RF and NB models.

Moreover, we investigated the differences between PCM, MC-QSAR, and QSAR. To test all these different combinations we decreased the maximum number of epochs from 2000 to 500 (see methods - machine learning methods - neural networks). These settings were validated on the temporal split because it represents a more realistic validation. Moreover, out of the models trained with 256 bits descriptors for ligands, the PCM DNN consistently outperformed the others, likely due to the fact that PCM profits from the added protein features containing information. This is consistent with the observation that more features in general perform better.Of the three different DNN, PCM slightly outperforms the other methods (average improvement 8%), although examples of both single and multi-task models are also found in the top performing methods (average increase 2% and 3% respectively).

With regard to the architecture, deep and wide networks seem to perform best (e.g. architecture 3 with an average increase of 12%) , although some of the shallow, multiclass and binary class networks (architecture 7) are also found in the top performing methods. Overall it seems that increasing dropout leads to a poorer performance. Dropout is a technique to prevent overfitting, by randomly discarding neural network units (and their connections). Hence if the dropout is too high the DNN is underfitted. This is confirmed by our results as higher dropout rates and dropout on the visible layer (the fingerprint/feature layer) result in a drop in performance (1 versus 2, and 3 versus 4). Therefore an option to be considered is a less aggressive dropout. Alternatively lowering the dropout percentage adaptively (during the training), just like with the learning rate would be an option too.Finally, the best performance is observed by pooling the predictions from all models in an average prediction or majority vote (improvement 25-26%, black bars Figure 6.6). This indicates that there is still room for improvement, indeed ensembles of different machine learning methods, including neural networks have been used to achieve competitive results on bioactivity prediction tasks. More context will be discussed below.

Putting this work into context

As touched upon, ChEMBL has fueled a diverse array of publications, here we limit ourselves to the most relevant and recent ones in the context of this paper. For instance Mervin *et al.*[25] constructed a (Bernoulli) Bayesian model on both ChEMBL and PubChem data. To balance the class sizes, a sphere exclusion algorithm was used to extract putative inactive data. A different threshold was used (10 μM, pChEMBL = 5) than in our study, and PubChem data in addition to the ChEMBL data was used.

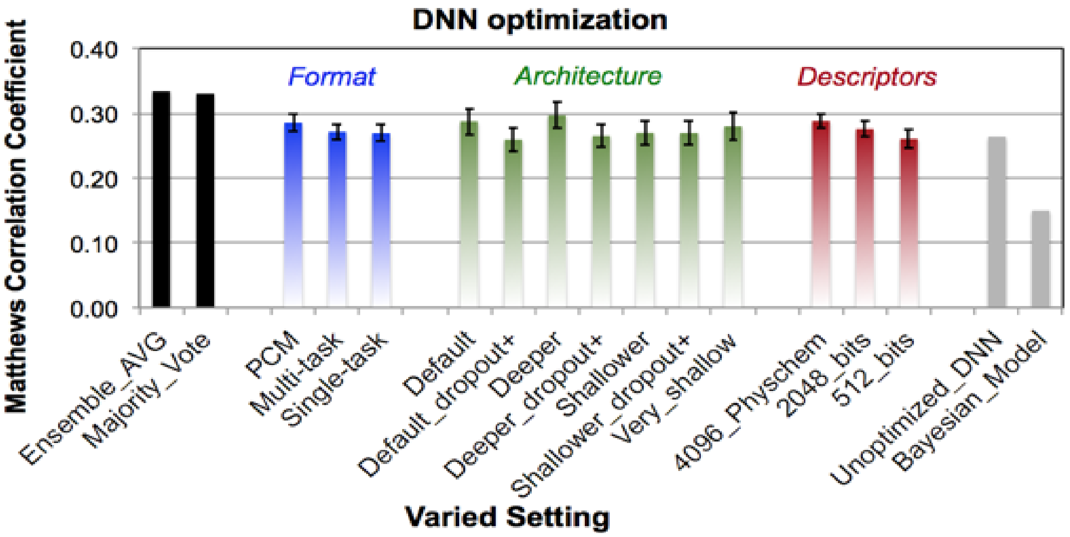


Figure 6.6: Average performance of the individual DNN grouped per method, architecture and descriptors. The black bars represent the ensemble methods (average value and majority vote). The grey bars on the right indicate the previous best performing DNN and NB model. We observed PCM to be the best way to model the data (blue bars), architecture 3 to be the best performing (green bars), and usage of 4096 bit descriptors with additional physicochemical property descriptors to perform the best (red bars). Using ensemble methods further improves performance.

In our hands inclusion of inactive molecules also enhanced the performance for the Naive Bayes models. A later study that was performed on ChEMBL data benchmarked the performance of a number of different algorithms.[26] Similar to our study the authors performed a temporal validation (on a later ChEMBL version) as a more realistic estimate of model performance. They also found that their method, potency-sensitive influence relevance voter (PS-IRV) outperformed other methods such as RF and SVM. However, we argue that our work in data curation, having data with only the highest confidence level from ChEMBL, leads to better performance (as can be seen in the AUC values obtained by them on their full set). IRV has been benchmarked before,[27] and can be seen as an extension of K-nearest neighbors in a shallow neural network. In that study random molecules were added (presumedly inactive), in addition to the experimentally inactive molecules to boost results. Inclusion of more data, and more specifically inactive molecules or molecules that have only low quality activity measurements is a line of future investigation we also aim to pursue.

Regarding the DNN, the influence of network architecture has been studied before,[16] where it was noted that the number of neurons especially impacts the performance of deeper structured networks. This corresponds to our observations where the deepest and widest network performed the best. Further finetuning of the architecture might be worthwhile; in multi-task networks trained for the Tox21 challenge up to 4 layers with 16,384 units were used.[19] Additionally it was found that multiclass outperformed binary class networks, and similar gains in performance were observed on the joint (multi-task) DNN published by Ma *et al*.[16]

Finally, work by Unterthiner *et al.* demonstrated similar DNN performance (AUC 0.83 versus 0.85 here) but worse NB performance (AUC 0.76 versus 0.83 here) compared to our work.[18] This divergence is potentially caused by the fact that their dataset was extracted with less curation of high quality ChEMBL data, which was the main reason for assembling this benchmark dataset. Moreover, Unterthiner *et al.* used much larger feature input vectors, requiring ample compute power to use the non-DNN based algorithms. We have shown that we can achieve similar performance on a smaller dataset with less fingerprint features, suggesting that there is much room for improvement by hyperparameter optimization. Furthermore Unterthiner *et al.* used a cost function weighted based on the dataset size. In our hands, experimentation choosing different weights inversely proportional to the target dataset size did not improve the performance of the models; however this is an avenue which can further be explored. Finally we have shown that usage of (simple) ensemble methods outperformed a single DNN alone, hence more sophisticated ensemble methods and inclusion of different models is a worthy follow up.

DNN have also been applied with promising results to the prediction of Drug-Induced Liver Injury,[28] although a different descriptor was used than the conventional fingerprints, i.e. directed acyclic graph recursive neural networks.[29] Similarly, convolutional networks were recently applied to the graphs of molecules. This outperformed standard extended connectivity fingerprints (ECFP).[30] Interestingly, contrary to extended connectivity fingerprints, such graphs are directly interpretable.

Conclusions

We have created and benchmarked a standardized set based on high quality ChEMBL data (version 20). We have tested and compared a diverse group of established and novel bioactivity modeling methods (descriptors, algorithms, and data formatting methods). Finally we have explored the potential of DNN by tuning their parameters and suggested ways for further improvement.From our results we draw a number of conclusions. Most of the methods and algorithms can create predictive models. Training time versus accuracy is a less relevant issue as the best performing methods required less than 10 hours. Commonly used 'random split' partitioning might sketch an overly optimistic performance estimate. We propose to split training and tests sets based on time-based differences or compound/publication clustering), providing a more challenging and more realistic performance estimate.

Usage of DNN based models increases model prediction quality over existing methods such as RF and NB models. As an added benefit, this improved performance is not obtained at the expense of highly increased training times due to GPU speedup.

We have shown that the widest and deepest DNN architectures produced the best results in combination with the most features. There is certainly still room for improvement as we have not reached hardware (memory) limitations or extreme training times. Moreover, the model ensembles of the 63 individual models further enhanced the results yielding performance that was almost 27% better than the best performing model prior to tuning, indicating that indeed better results are possible.

Taken together we anticipate that methods discussed in this paper can be applied on a routine basis and can be fine-tuned to the problem (e.g. target) of interest. Moreover, due to low training time and high performance we anticipate that DNN will become a useful addition in the field of bioactivity modeling.

References

1. Gaulton A, Bellis LJ, Bento AP, Chambers J, Davies M, Hersey A, et al. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Res*. 2012;40: D1100–7.
2. Bento AP, Gaulton A, Hersey A, Bellis LJ, Chambers J, Davies M, et al. The ChEMBL bioactivity database: an update. *Nucleic Acids Res*. 2014;42: D1083–90.
3. Papadatos G, Gaulton A, Hersey A, Overington JP. Activity, assay and target data curation and quality in the ChEMBL database. *J Comput Aided Mol Des*. 2015;29: 885–896.
4. Kramer C, Kalliokoski T, Gedeck P, Vulpetti A. The experimental uncertainty of heterogeneous public K(i) data. *J Med Chem.* 2012;55: 5165–5173.
5. Jacob L, Laurent J, Brice H, Véronique S, Jean-Philippe V. Virtual screening of GPCRs: An in silico chemogenomics approach. *BMC Bioinformatics.* 2008;9: 363.
6. Sliwoski G, Kothiwale S, Meiler J, Lowe EW Jr. Computational methods in drug discovery. *Pharmacol Rev.* 2014;66: 334–395.
7. Nidhi, Nidhi, Meir G, Davies JW, Jenkins JL. Prediction of Biological Targets for Compounds Using Multiple-Category Bayesian Models Trained on Chemogenomics Databases. *J Chem Inf Model.* 2006;46: 1124–1133.
8. Schneider S, Provasi D, Filizola M. The Dynamic Process of Drug-GPCR Binding at Either Orthosteric or Allosteric Sites Evaluated by Metadynamics. *Methods Mol Biol.* 2015;1335: 277–294.
9. Bender A, Young DW, Jenkins JL, Serrano M, Mikhailov D, Clemons P a., et al. Chemogenomic Data Analysis: Prediction of Small-Molecule Targets and the Advent of Biological Fingerprints. *Comb Chem High Throughput Screen.* 2007;10: 719–731.
10. Afzal AM, Mussa HY, Turner RE, Bender A, Glen RC. A multi-label approach to target prediction taking ligand promiscuity into account. *J Cheminform.* 2015;7: 24.
11. van Westen GJP, Wegner JK, IJzerman AP, van Vlijmen HWT, Bender A. Proteochemometric modeling as a tool to design selective compounds and for extrapolating to novel targets. *Med Chem Commun.* 2011;2: 16–30.
12. Cortés-Ciriano I, Isidro C-C, Ain QU, Vigneshwari S, Lenselink EB, Oscar M-L, et al. Polypharmacology modelling using proteochemometrics (PCM): recent methodological developments, applications to target families, and future prospects. *Med Chem Commun.* 2015;6: 24–50.
13. Wikberg JE, Lapinsh M, Prusis P. Proteochemometrics: A tool for modelling the molecular interaction space. *Chemogenomics in Drug Discovery - A Medicinal Chemistry Perspective.* 2004; 289–309.
14. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;529: 484–489.
15. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521: 436–444.
16. Ma J, Sheridan RP, Liaw A, Dahl GE, Svetnik V. Deep neural nets as a method for quantitative structure-activity relationships. J Chem Inf Model. 2015;55: 263–274.
17. Ramsundar B, Kearnes S, Riley P, Webster D, Konerding D, Pande V. Massively Multitask Networks for Drug Discovery *[Internet]. arXiv [stat.ML]*. 2015. Available: http://arxiv.org/abs/1502.02072
18. Unterthiner T, Mayr A, Klambauer G, Steijaert M, Wegner JK, Ceulemans H, et al. Deep learning as an opportunity in virtual screening. *Proceedings of the Deep Learning Workshop at NIPS.* 2014;
19. Mayr A, Klambauer G, Unterthiner T, Hochreiter S. DeepTox: Toxicity Prediction using Deep Learning. *Front Environ Sci Eng* China. Frontiers; 2015;3. doi:10.3389/fenvs.2015.00080
20. Accelrys Software Inc. Pipeline Pilot (Version 9.2). Scitegic;
21. R Development Core Team. R: A Language and Environment for Statistical Computing. R *Foundation for Statistical Computing*. 2006;
22. Landrum G. RDKit: Cheminformatics and Machine Learning Software *[Internet]*. 2013. Available: http://www.rdkit.org
23. Scikit S-LD. Scikit-learn: Machine learning in python. *J Mach Learn Res*. 2011;12: 2825–2830.
24. Davies M, Nowotka M, Papadatos G, Atkinson F, van Westen G, Dedman N, et al. MyChEMBL: A Virtual Platform for Distributing Cheminformatics Tools and Open Data. Challenges. 2014;5: 334.
25. Mervin LH, Afzal AM, Drakakis G, Lewis R, Engkvist O, Bender A. Target prediction utilising negative bioactivity data covering large chemical space. *J Cheminform*. 2015;7: 51.
26. Lusci A, Browning M, Fooshee D, Swamidass J, Baldi P. Accurate and efficient target prediction using a potency-sensitive influence-relevance voter. *J Cheminform.* 2015;7: 63.
27. Swamidass SJ, Azencott C-A, Lin T-W, Gramajo H, Tsai S-C, Baldi P. Influence relevance voting: an accurate and interpretable virtual high throughput screening method. *J Chem Inf Model.* 2009;49: 756–766.
28. Xu Y, Dai Z, Chen F, Gao S, Pei J, Lai L. Deep Learning for Drug-Induced Liver Injury. *J Chem Inf Model.* 2015;55: 2085–2093.

29. Lusci A, Pollastri G, Baldi P. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *J Chem Inf Model.* 2013;53: 1563–1575.
30. Duvenaud DK, Maclaurin D, Iparraguirre J, Bombarell R, Hirzel T, Aspuru-Guzik A, et al. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R, editors. *Advances in Neural Information Processing Systems* 28. Curran Associates, Inc.; 2015. pp. 2224–2232.
31. Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, et al. Theano: a CPU and GPU math expression compiler. *Proceedings of the Python for scientific computing conference* (SciPy). Austin, TX; 2010. p. 3.
32. Dieleman S, Schlüter J, Raffel C, Olson E, Sønderby SK, Nouri D, et al. Lasagne: First release. Zenodo: Geneva, Switzerland. 2015;
33. Sander Dieleman JS. Lasagne. In: *Github [Internet]*. [cited Dember 18th 2015]. Available: https://github.com/Lasagne/Lasagne
34. Daniel Nouri BB. Nolearn. In: *Github [Internet]*. [cited December 18th 2015]. Available: https://github.com/dnouri/nolearn
35. Bemis GW, Murcko MA. The properties of known drugs. 1. Molecular frameworks. *J Med Chem*. 1996;39: 2887–2893.
36. Arup K. Ghose *., Vellarkad N. Viswanadhan, Wendoloski JJ. Prediction of Hydrophobic (Lipophilic) Properties of Small Organic Molecules Using Fragmental Methods: An Analysis of ALOGP and CLOGP Methods. *J Phys Chem A.* 1998;102: 3762–3772.
37. Shrake A, Rupley JA. Environment and exposure to solvent of protein atoms. Lysozyme and insulin. *J Mol Biol.* 1973;79: 351–371.
38. Ertl P, Rohde B, Selzer P. Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties. *J Med Chem.* 2000;43: 3714–3717.
39. van Westen GJP, Gaulton A, Overington JP. Chemical, Target, and Bioactive Properties of Allosteric Modulation. *PLoS Comput Biol.* 2014;10: e1003559.
40. Rogers D, Hahn M. Extended-connectivity fingerprints. *J Chem Inf Model.* 2010;50. doi:10.1021/ci100050t
41. Landrum G. Fingerprints in the RDKit [Internet]. *RDkit User Group Meeting 2012*; London. Available: http://rdkit.org/UGM/2012/Landrum_RDKit_UGM.Fingerprints.Final.pptx.pdf
42. Van Westen GJP, Swier RF, Cortes-Ciriano I, Wegner JK, IJzerman AP, Van Vlijmen HWT, et al. Benchmarking of Protein Descriptors in Proteochemometric Modeling (Part 2): Modeling Performance of 13 Amino Acid Descriptors. *J Cheminform.* 2013;5: 42.
43. Van Westen GJP, Swier RF, Wegner JK, IJzerman AP, Van Vlijmen HWT, Bender A. Benchmarking of Protein Descriptors in Proteochemometric Modeling (Part 1): Comparative Study of 13 Amino Acid Descriptors. *J Cheminform.* 2013;5: 41.
44. Mugumbate G, Abrahams KA, Cox JAG, Papadatos G, van Westen G, Lelièvre J, et al. Mycobacterial Dihydrofolate Reductase Inhibitors Identified Using Chemogenomic Methods and In Vitro Validation. *PLoS One.* 2015;10: e0121492.
45. MartÌnez-JimÈnez F, Papadatos G, Yang L, Wallace IM, Kumar V, Pieper U, et al. Target Prediction for an Open Access Set of Compounds Active against Mycobacterium tuberculosis. *PLoS Comput Biol.* 2013;9: e1003253.
46. Manning CD, Raghavan P, Schütze H, Others. Introduction to information retrieval. *Cambridge university press* Cambridge; 2008.
47. van Westen GJ, Swier RF, Cortes-Ciriano I, Wegner JK, Overington JP, Ijzerman AP, et al. Benchmarking of protein descriptor sets in proteochemometric modeling (part 2): modeling performance of 13 amino acid descriptor sets. *J Cheminform.* 2013;5: 42.
48. Sutskever I, Martens J, Dahl G, Hinton G. On the importance of initialization and momentum in deep learning. *Proceedings of the 30th international conference on machine learning* (ICML-13). 2013. pp. 1139–1147.
49. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J Mach Learn Res. JMLR.org*; 2014;15: 1929–1958.
50. Matthews BW. Comparison of the Predicted and Observed Secondary Structure of t4 Phage Lysozyme. *Biochim Biophys Acta.* 1975;405: 442–451.