Prediction of Membrane Permeability of Molecules Using Machine Learning

Seokgyun Ham, Xin Wang, Nilotpal Chakraborty, Anri Karanovich

Abstract

Understanding the permeability of small drug-like molecules across a lipid membrane is fundamental in pharmaceutical applications. Physics-based models have found that the main physio-chemical properties determining the permeability are bulk partitioning free energy ΔG and pK_a, but they are expensive to compute experimentally or computationally. Hence, some research exists in the area of using solely molecular structure information to predict permeability. In this work, by using different combinations of the main physio-chemical properties and molecular structure information, we predict the membrane permeability of small molecules using three machine learning (ML) algorithms, the Lasso model, the Multi-layer Perceptron (MLP) model, and the combined model of the Lasso and MLP. Molecular descriptors and fingerprints, obtained from the SMILES string of a molecule, are used with/without the key features (ΔG and pK_a) as input to the ML models. The Lasso model shows R^2 values of 0.80 and 0.72 with and without the key features, respectively. With the key features, the prediction of the MLP model is very accurate (R^2=0.99), while it decreases without the key features (R^2=0.89). The combined Lasso-MLP model shows a slightly better prediction without the key features (R^2=0.90). Our results validate the possibility of predicting the permeability well simply based on data extracted from the SMILES string, without the main physio-chemical properties.

1 Team Member Contributions

Each author's name is abbreviated as the following:
Seokgyun Ham: SH, Xin Wang: XW, Nilopal Chakraborty: NC, Anri Karanovich: AK

1.1 The project report writing

- Abstract: SH
- Introduction: SH, AK
- Computational methods - Dataset: NC
- Computational methods - ML algorithms - Lasso: AK
- Computational methods - ML algorithms - MLP: XW
- Results - Lasso: NC, AK
- Results - MLP: SH, XW
- Conclusion: SH

1.2 Project work

- Data pre-processing: SH, NC
- Lasso: NC, AK
- MLP - only MLP: XW
- MLP - Lasso-MLP: SH, XW

1.3 Source code

- Lasso: 5 major tasks performed by the **main.py** file - 1, 4 by AK and 2,3,5 by NC. In addition there are two modules - **module 1** and **module 2**. There are 8 functions in **module 1**, 5 of them by AK and 3 of them by NC. **module 2** is by NC. More details can be found in source codes.
- MLP and Lasso-MLP (MLP_and_Lasso_MLP.py): SH, XW

2 Introduction

Quantitative Structure-Activity Relationship (QSAR) is a computational method for revealing relationships between structural properties of chemical compounds and physio chemical properties such as biological activities. Historically many endeavors in modeling QSAR have been made, and new chemical compounds and drugs have been discovered such as penicillin [17]. However, the process is essentially based on trial-and-error, inherently consuming enormous time and resources needed for experiments [4]. For example, according to Inokuchi et. al,[34], the average development cost for a new drug is currently estimated to be around 2.8 billion. Moreover, modeling QSAR requires top-notch level of physicochemical insights, slowing its process further. Thanks to the rapid development of computer hardware and artificial intelligence technology, simulation- and data-driven science have been utilized to accelerate the process of finding new drug candidates effectively. Several studies have shown that the machine-learning-based approaches predict physical properties based on molecular structure and successfully reduce the space of candidate materials [4, 15].

Machine learning techniques have been successfully applied in a great variety of studies of molecular features and attributes, from prediction of the physico-chemical properties such as total energy and potential energy surface[6, 5], excitation energies[26], polarizability[11, 33], dipole moment and partial charges[32, 11] etc., to finding the reactions paths for molecular synthesis[10], drug discovery and design[35, 20], search for potential catalyst candidates[22], organic LED material discovery[13], and many others.

A particular research interest lies in the prediction of small molecules' permeability through a lipid membrane, which could be useful for designing drugs with desired cellular absorption properties. The experimental estimation of the permeaiblity [29, 3], can be very costly and time-consuming, and unable to efficiently probe the vast chemical space of potential drug molecules, which can span as many as $10^{60} - 10^{100}$ molecules [25]. Therefore, physical modeling and computational approaches are often preferred. Within the inhomogeneous solubility-diffusion model [2, 28, 9] the inverse of permeability can be estimated as:

$$P^{-1} = \int_z \exp\left(\frac{G(z)/k_B T}{D(z)}\right) dz \tag{1}$$

with z being a coordinate perpendicular to the membrane's surface, $G(z)$ being the potential of the mean force (PMF) profile, and $D(z)$ is local diffusivity. Both $G(z)$ and $D(z)$ can be estimated from computational physical models, including the molecular dynamics simulations [16]. However, these simulations can in turn be computationally expensive[18].

One can therefore see the appeal of using a machine learning model, which, when trained, could potentially predict the permeability of small molecules very quickly and without intermediate calculations, just from its structure. Important work in this direction was done by Chen, Shen, and Li[9], who performed two supervised machine learning methods - LASSO and deep neural network (DNN) - on a database of small drug-like molecules taken from published literature[20, 14]. Their raw data set included unimers, dimers and trimers and had over 770k molecules, however they sub sampled the data set to train their ML models. The structural and chemical properties of each molecule were transcribed into a set of numerical values in form of a 1024-bit long Morgan fingerprint[27] and a set of 200 molecular descriptors, all of which were automatically generated with the RDKit package [1] (for a detailed explanation of molecular fingerprints and descriptors, see the Research Design section). The LASSO model allows to identify the most important features and reduces the effective width of the dataset by throwing the unimportant ones, while pertaining sufficiently good accuracy, resulting in a generalizable and interpretable model. On the other hand, the DNN approach, although not as easy to interpret, gave a more accurate prediction than LASSO.

The work by Chen et al.[9] shows that machine learning techniques can be extremely useful for both the interpretation of existing data and effective prediction of the membrane permeability. Since it is still an emerging field, it may be useful to experiment with the setup for ML application. Firstly, the datasets used in that work, produced by Menichetti et al.[20] and Hoffmann et al. [14], provide more information than just the molecular structure and permeability coefficient. The data set includes such information as free energy difference for the molecule in water and octanol environments, the acidity of the compounds, apKa and bpKa. It would be interesting to include these physio-chemical properties as features, instead of (and in addition to) the molecular structure, and see if it improves the performance of the ML algorithms.

3 Computational methods

3.1 Dataset

The dataset for this study was taken from the study by Menichetti, Kanekal, and Bereau[20], and it is pucblicly available. As mentioned above, the authors in their work try to predict the log of the permeability coefficient P for small molecules using molecular simulations with Martini coarse-grained structure representation [19, 7]. Here, we use the dataset containing the information for systems represented as Martini unimers.

The raw data set looks like figure 1: there are 92456 samples and 8 columns, with Col 8 (Col # means # -th column of the dataset) being the target variable, and 7 other designating various features of each molecule.

These features include:

- **SMILES string** (Col.1) - a representation of the molecular structure in form of a textual string;

SMILES string	Martini representation	Delta_G1	Delta_G2	apKa	bpKa	Acidity	logP
CC	C5	−1.977	−1.656	NOT	NOT	N	0.269
CN	P2	1.455	0.920	NOT	10.08	B	−4.686
CO	P3	1.894	2.106	15.78	−2.46	A	−2.865
CF	Na	−0.549	−0.595	NOT	NOT	N	−1.005
FF	N0	−1.304	−1.009	NOT	NOT	N	−0.191
C=C	N0	−1.235	−1.009	NOT	NOT	N	−0.191

Figure 1: The first six lines of our dataset

- **Martini bead representation** (Col.2) - the coarse-grained model of each molecule [19]. Note that the given dataset contains only the systems represented as unimers, so this feature here is not very informative;
- $\mathbf{\Delta G_1}$ (Col.3) - water/octanol partitioning free energy of the molecule, predicted by ALOGPS (neural network) software [30];
- $\mathbf{\Delta G_2}$ (Col.4) - water/octanol partitioning free energy of the molecule, deduced from the Martini bead picture;
- $\mathbf{apK_a}$ (Col.5) - acid strength of a given acidic compound A, defined as $\mathrm{apK}_a = \log_{10} \frac{[AH]}{[A^-][H^+]}$ ([X] is the concentration of X);
- $\mathbf{bpK_a}$ (Col.6)- acid strength, associated with the given conjugate base B: $\mathrm{bpK}_a = \log_{10} \frac{[BH^+]}{[B][H^+]}$
- **Acidity** (Col.7) - categorical variable denoting if this compound is acidic (A), basic (B), zwitterionic (Z), or neutral (N)

Menichetti et al. justify the choice of these particular features by stating that the diffusivity $D(z)$ in equation 1 does not depend much on the chemical properties of the molecules [8], while ΔG has been shown to correlate strongly with the PMF profile $G(z)$ [21], and adding the acid strength pK_a accounts for contributions of different protonation states [20].

In Fig. 1, there are many samples in Col5 (apK_a) and Col6 (bpK_a) and which have "NOT" values, however these can't be discarded as these have a physical meaning: NOT in Col5 means apK_a higher than 20, and NOT in Col6 means bpK_a than -10. So, we replace those values by 20 and -10 respectively. Since Col 7 (Acidity) is a categorical variable, we do One-Hot Encoding on this feature to produce 3 more columns. The final processed data set (before normalization) is shown in Fig. 2.

	Col1	Col2	Col3	Col4	Col5	Col6	Col8	A	B	N
0	CC	C5	-1.977	-1.656	20	-10	0.269	0.0	0.0	1.0
1	CN	P2	1.455	0.920	20	10.08	-4.686	0.0	1.0	0.0
2	CO	P3	1.894	2.106	15.78	-2.46	-2.865	1.0	0.0	0.0
3	CF	Na	-0.549	-0.595	20	-10	-1.005	0.0	0.0	1.0
4	FF	N0	-1.304	-1.009	20	-10	-0.191	0.0	0.0	1.0

Figure 2: Dataframe after initial preprocessing

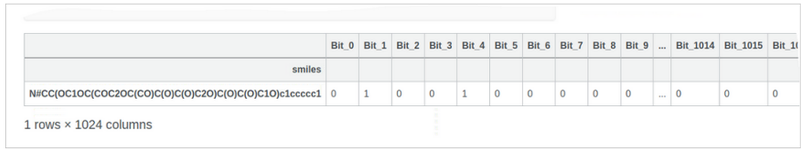
One important thing to note here is that Col1 and Col2 (SMILES and Martini) are unique strings in every row, so they cannot be easily used as model features in their current form. To differentiate between Col1, Col2 and the rest of the features, we group Col3-Col6 and Col A, B, N (Fig. 2) and call these **key features**. These are 7 in number and represent chemical properties of a molecule. Apart from the **key features**, Col 2 (Martini) is dropped, but Col1 (SMILES string) can still be involved. It is possible to extract a lot of features from the SMILES string of any molecule using the cheminformatics package RDkit [1]. This method can be divided into two different approaches - molecular fingerprints and molecular descriptors.

Molecular fingerprints A molecular fingerprint is a 1024-bit long vector encoding the structural properties of the molecule. In our work, we used Morgan-type fingerprints [27], which are generated by an iterative process. First, every atom in the molecule is associated with a set of numerical features, such as atomic weight, number, bond count etc., which are then hashed into a single atomic integer identifier. Next, the identifiers of each given atom *and* its nearest neighbors are hashed into a new identifier for this atom. After the update, the duplicate identifiers are removed, and the process repeats for a pre-set number of iterations (2 in our case). Finally, the resulting identifier list is hashed into a 1024-bit-long fingerprint [1].

From the outlined procedure, it's evident that second-generation atomic identifiers, in addition to the given atom's properties, encode also the nearest-neighbor ones; 3rd-gen also depend on nearest-but-one neighbors, and so on. This way, the Morgan algorithm can capture molecular substructure information, and the fingerprint's bits essentially represent the attachment to various functional groups, such as carboxylic or amino acid group (0 - no attachment, 1 - is attachment).

In our work, the Morgan fingerprints were generated using the RDKit package [1] from the SMILES strings. An example of such generated fingerprint is shown in the figure 3.

[1] For more details check the work by D. Rogers and M. Hahn [27]



	Bit_0	Bit_1	Bit_2	Bit_3	Bit_4	Bit_5	Bit_6	Bit_7	Bit_8	Bit_9	...	Bit_1014	Bit_1015	Bit_1
smiles														
N#CC(OC1OC(COC2OC(CO)C(O)C(O)C2O)C(O)C(O)C1O)c1ccccc1	0	1	0	0	1	0	0	0	0	0	...	0	0	0

1 rows × 1024 columns

Figure 3: An example of a molecular fingerprint in 1024 bits. The first column is the SMILES string of the molecule and the rests are the individual bits of its fingerprint.

Molecular descriptors In addition to the fingerprints, it is also possible to derive a number of other numerical features from the SMILES string, which, however, have a more direct correspondence to the molecules' physico-chemical, rather than structural, properties. These may include such values as molecular weights, number of valence electrons, the magnitude of partial charges, number of bonds, number of atoms of a particular element, etc. These are called "molecular descriptors", and they can be very useful as predictive features for our models. In the RDKit package [1], there are 200 such predictors which can be automatically generated using SMILES. An example of all 200 molecular descriptors generated using RDkit is shown in Fig. 4, which correspond to the SMILES string "C=CCCC#CC#CC#N".


```
[3.518448086764338, 246.41375363598425, 7.6568542494923815, 5.645884208249226, 5.645884208249226, 4.914213562373096, 2.893
6567692709785, 2.8936567692709785, 1.4909303633612523, 1.4909303633612523, 0.7427517596151512, 0.7427517596151512, 0.34012
58798075756, 0.3401258798075756, 0.0, 0.0, 0.0, 0.0, 0.0, 12.841643245852019, 0.0, 12.145241419626155, 0.0, 30.26067295902
1075, 5.261891554738487, 129.057849224, 1.3, 2.1, 2.7, 0.2222222222222222, -1.65, 10, 122.10599999999998, 175.076747998214
07, 8.35, 7.349999999999998, 6.35, 61.360215376005414, 7.961393061854372, 0.18280740590656763, 7.961393061854372, 0.152966
0761027337, 0.7674688208616779, 0.1529660761027337, 0.7674688208616779, -0.1828074059065676 3, 1.4829800000000002, 40.59600
000000001, 129.16199999999998, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 2, 0, 0, 0, 48, 0.0, 0.0, 6.069221312792274, 0.0, 0.0,
0.0, 0.0, 0.0, 5.261891554738487, 6.578935683598497, 11.996454425689524, 12.341255941781654, 18.261690260637298, 0.0, 0, 0
.0, 0.0, 5.261891554738487, 0.0, 0.0, 12.841643245852019, 0.0, 12.654955790432378, 0.0, 29.75095858821485, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 35.01285014295334, 12.841643245852019, 12.654955790432378, 0.0, 0.0, 0.0, 23.79, 0.0, 0.0, 0.0, 7.961393061
854372, 0.0, 9.883576388888889, 1.6748256015369112, 3.441709971025447, 3.5384949766943814, 0.0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0.3145383597730
3723]
```


Figure 4: Numerical values of 200 molecular descriptors

We now have in total up to 1231 different features for each molecule. However, due to the different nature, origin, and cardinality of the three groups of input data (key features, fingerprints, and molecular descriptors), it can be reasonable not to just use all of them at once, but combine in different ways and see how the performance is affected and which data are more important. Therefore, we came up with 7 different ways to group these features - 7 sets of input data, as shown in table 1 - and trained our model architectures on each of these sets separately.

Set	Input selection	No. of inputs
1	key features (7)	7
2	key features (7) + fingerprints (1024)	1031
3	key features (7) + molecular descriptors (200)	207
4	key features + fingerprints + molecular descriptors	1231
5	fingerprints	1024
6	molecular descriptors	200
7	fingerprints + molecular descriptors	1224

Table 1: Seven sets of input selection

3.1.1 Data visualization

In our project proposal, we made a t-SNE plot after doing a PCA on the **key features**. However, this time we have so many more features that doing a similar treatment will not help understanding the data set much. Further, before we get results from running various ML models, it is difficult to ascertain which fingerprints or molecular descriptors predominantly affect the permeability. Menichetti et.al [20] were able to show that ΔG of the Martini unimer and bpKa (Col4 and Col6 in Fig. 1) were the most important features when the SMILES string was not taken into account. Thus, we make a scatter plot of bpKa and ΔG in Fig. 5. The points are color coded in terms of the target variable, $\log_{10} P$.

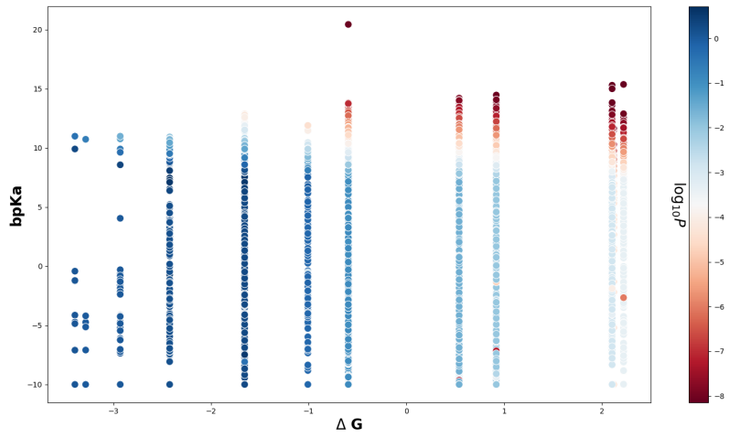


Figure 5: Scatter plot of ΔG and bpKa. The color represents the value of $\log_{10} P$.

3.2 ML algorithms

3.2.1 LASSO method

LASSO[31] is a type of linear regression method, whose objective is to minimize the sum of squares of the residuals ($L = \frac{1}{2n}\sum_i \left(y_i - \sum_j x_{ij} w_j\right) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2$), with respect to the weights w_j, subject to the constraint $\sum_j |w_j| = ||\mathbf{w}||_1 \leq t$, for some parameter t. This condition can also be interpreted, via the use of Lagrange multiplier α, as a problem of minimizing the following function:

$$\tilde{L}(\mathbf{w}) = ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \alpha||\mathbf{w}||_1 \tag{2}$$

In this form, the LASSO model (eq.2) resembles ridge regression, with an ℓ_1 regularization term instead of ℓ_2. Like in ridge regression, the regularization limits the weights' magnitudes, which reduces overfitting and results in a more generalizable model. However, unlike in ridge regresison, the LASSO method is capable of setting certain weights w_j identically to 0. This happens because, with the smooth ℓ_2 term, $\tilde{L}(w_j)$ would only be able to reach its minimum at one of the stationary points ($\partial\tilde{L}/\partial w_j = 0$), while $\ell_1 \propto |w_j|$ has a minimum and a "cusp" at $w_j = 0$ that can also minimize $\tilde{L}(w_j)$, especially if residuals don't vary much with w_j. This property of LASSO allows to effectively "shrink" the unimportant features, thus automatically performing feature selection to provide a simpler and more generalizable model. It also makes the model easier to interpret, and can even be useful for reducing the dataset width for the further applications in other ML methods.

The amount of regularization and feature selection depends on the magnitude of the parameter α (eq.2). The larger is α, the stronger is the regularization and the fewer features "survive". Its exact value should be chosen such that the resulting model would have the best performance on test data. In order to avoid overfitting, it is common ot use k-fold cross-validation on the training set when picking α [12]. After the appropriate α is picked, the optimal weights w_j are calculated using a coordinate-descent method. In our study, we used the scikit-learn package's [24] implementation of the LASSO method with automatic parameter selection - LassoCV - which applies cross-validation to search for an optimal alpha to maximize the coefficient of determination R^2 after the subsequent weight optimization.

3.2.2 MLP model

Another machine-learning approach we used is a fully-connected, feed-forward MLP model. A network placed with some random weights and biases initially is applied to the training dataset, and a loss function, such as MSE or MAE, is estimated on the training and test datasets. Then, using one of the gradient-based backpropagation methods, eg. Adam, the weights and biases within the network will be updated to minimize the loss function until the minimum loss is obtained.

In this work, we used MLP regressor, in scikit-learn package[24], to map different sets of inputs to the permeability. We took two approaches using MLP: one approach is to use MLP solely based on the seven different sets of input selection (see Table 1). The other one called Lasso-MLP model is to feed these inputs which show non-zero weights in Lasso. With this method, the insignificant features are discarded, and therefore the number of the parameters optimized is reduced. This approach was inspired by Mozafari et al.[23]

The entire data set was split into a training set (70%) and a test set (30%). The MLP model was trained and then tested using the training set and the test set, respectively. In order to obtain the best model, GridSearchCV was employed to tune the parameters, eg, hidden layers & nodes, alpha, and activation functions, in MLP regression model. When training models, MLP regression model has a possibility to get overfitting. To avoid this and obtain a precise model, K-fold cross-validation(k = 5) was applied to select the best model. For the model assessment, R^2 was adopted to compare the performance of different models.

The definition of R^2 is shown below[24]:

$$R^2 = 1 - \frac{u}{v} = 1 - \frac{(y\_true - y\_pred) * *2).sum()}{((y\_true - y\_true.mean()) * *2).sum()}$$

4 Results and discussion

4.1 Permeability prediction with Lasso

We performed the LASSO regression method on 7 different sets of the input data, as outlined in Table 1. In each case, the optimal value of the regularization parameter α was found with the help of the LassoCV optimizer with cv= 5-fold cross-validation. Fig. 6 shows the LassoCV result for set 4 inputs and the corresponding value of α that minimizes the mean squared error (MSE) is $2.76E-04$. For this α, the training and cross validation score vs. training examples are shown in Fig. 7. The difference between training and validation score is less and shows sign of good bias-variance trade-off. The Lasso model thus performs well with this set of inputs.

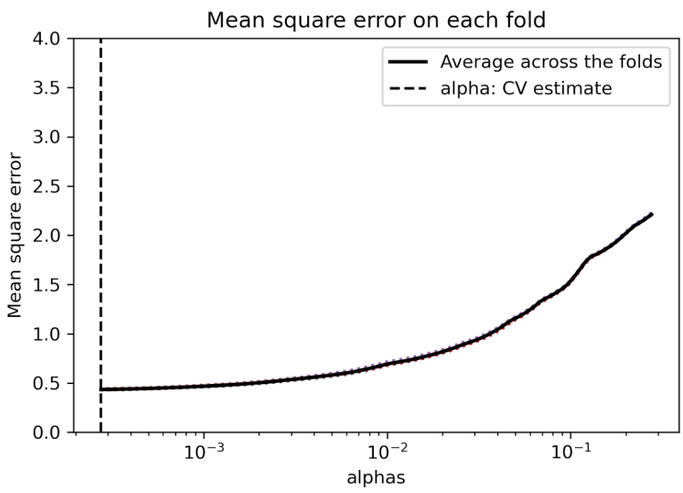


Figure 6: L1 regularization using **LassoCV**

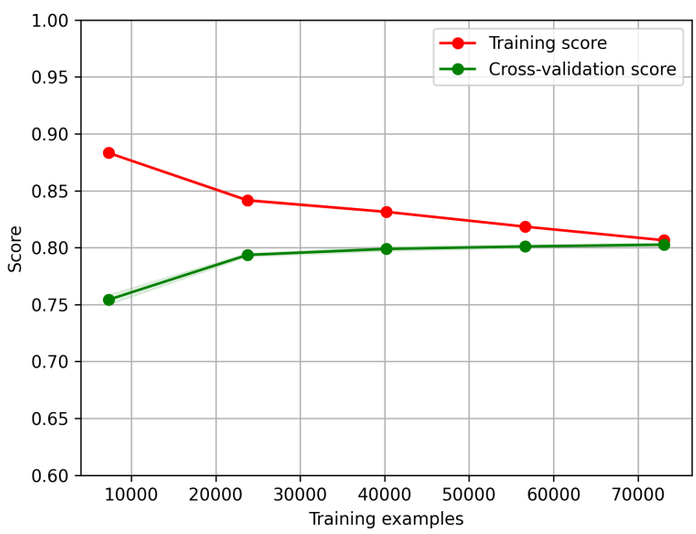


Figure 7: Training and cross validation scores of Lasso model applied to set 4 input

Overall, the performance of the Lasso model applied to set 1-7 (Table 1) are compared in one single bar plot (Fig. 8). With only the **key features** (set 1), the model doesn't perform very well but with only the fingerprints and/or molecular descriptors (sets 5,6,7), the model performs better. Thus, the significance of molecular substructure in permeability prediction comes into the picture. One possible reason for this might be that a linear classifier like Lasso is not built to work well with a target that is non linear (see eq.1, noting that $\Delta G \propto G(z)$). Out of fingerprints and molecular descriptors, the model performs better with the latter as input. We conclude that molecular descriptors are more important than fingerprints in predicting the permeability. This was also validated in [9].

With a train R^2 value of 0.8, the model has the best performance when input is set 4 (**key features** + fingerprints + molecular descriptors). The significance of the Lasso model, however lies in its ability to identify the input features which predominantly affect the target. Fig. 8 also depicts the no. of features with non-zero weights (orange bars) alongside the total no. of input features (pink bars) for sets 1-7 as input. These non-zero features are used as input to the Lasso-MLP model so as to check if using lesser features improves the performance of the model.

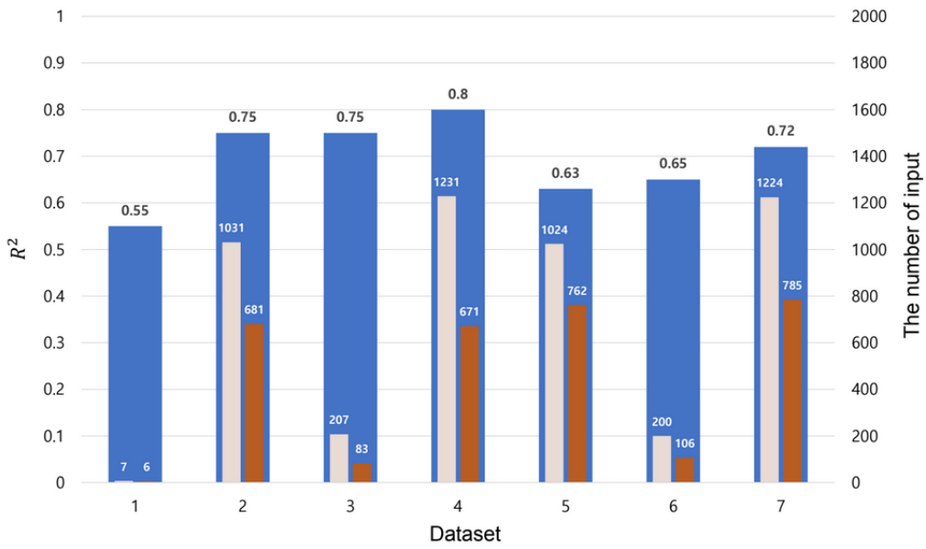


Figure 8: Performance of the Lasso model applied to set 1-7. The blue bars represent the test R^2 values, the pink bars the total number of input features as shown in Table 1, and the orange bars the number of features with non-zero weights

It is also interesting to see which features are the most important in prediction. Fig. 9 shows the evolution of feature weights for set 4. The features with ten largest weights are shown in the legend. We can see that only three **key features**: bpKa (Col 6 in Fig. 1), DGM (ΔG of the Martini unimer, Col 4 in Fig. 1) and apKa (Col 5 in Fig. 1) are the most important. This result just proves our reasoning in 3.1.1. Furthermore, 7 out of the ten largest weights are of molecular descriptors, again confirming the fact that these are more important than fingerprints in permeability prediction.

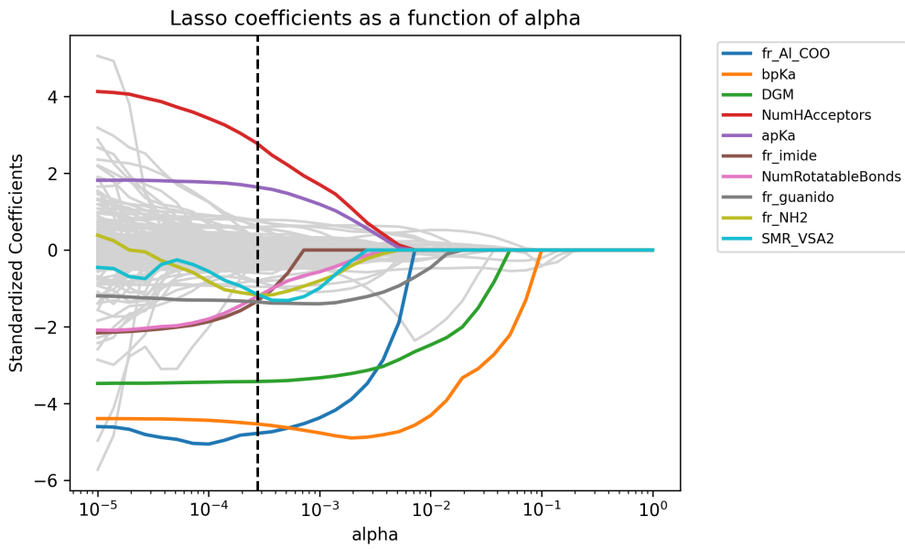


Figure 9: The evolution of the feature weights as the function of the regularization α, for input set 4. Colored are the curves for the 10 largest weights as observed at the optimal α (indicated by the dashed line).

4.2 Permeability prediction with MLP

For each set of inputs, 54 combinations of hyperparameters in total were explored and the best model was selected. All the hyperparameters explored and the best architechture for each model are summarized in Table 2.

Input	Hidden layer sizes	Alpha	Activation function	Best parameters	
				MLP	Lasso+MLP
SET 1	3,(3,2),(5,2)	0.00001, 0.00003, 0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1	'tanh', 'relu'	'activation': 'tanh', 'alpha': 3e-05, 'hidden_layer_sizes': (5, 2)	'activation': 'tanh', 'alpha': 0.001, 'hidden_layer_sizes': (5, 2)
SET 2	(600,100),(500,250,100), (300,100,30)			'activation': 'relu', 'alpha': 1e-05, 'hidden_layer_sizes': (500, 250, 100)	'activation': 'tanh', 'alpha': 0.003, 'hidden_layer_sizes': (500, 250, 100)
SET 3	(70,25,8),(100,50,25), (100,25)			'activation': 'relu', 'alpha': 3e-05, 'hidden_layer_sizes': (100, 50, 25)	'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (70, 25, 8)
SET 4	(600,100),(500,250,100), (300,100,30)			'activation': 'relu', 'alpha': 1e-05, 'hidden_layer_sizes': (500, 250, 100)	'activation': 'tanh', 'alpha': 0.003, 'hidden_layer_sizes': (500, 250, 100)
SET 5	(600,100),(500,250,100), (300,100,30)			'activation': 'relu', 'alpha': 1e-05, 'hidden_layer_sizes': (500, 250, 100)	'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (500, 250, 100)
SET 6	(70,25,8),(100,50,25), (100,25)			'activation': 'tanh', 'alpha': 1e-05, 'hidden_layer_sizes': (100, 50, 25)	'activation': 'relu', 'alpha': 0.003, 'hidden_layer_sizes': (100, 50, 25)
SET 7	(600,100),(500,250,100), (300,100,30)			'activation': 'relu', 'alpha': 3e-05, 'hidden_layer_sizes': (500, 250, 100)	'activation': 'relu', 'alpha': 0.001, 'hidden_layer_sizes': (500, 250, 100)

Table 2: All the parameters explored to find the best architecture and the hyperparameters

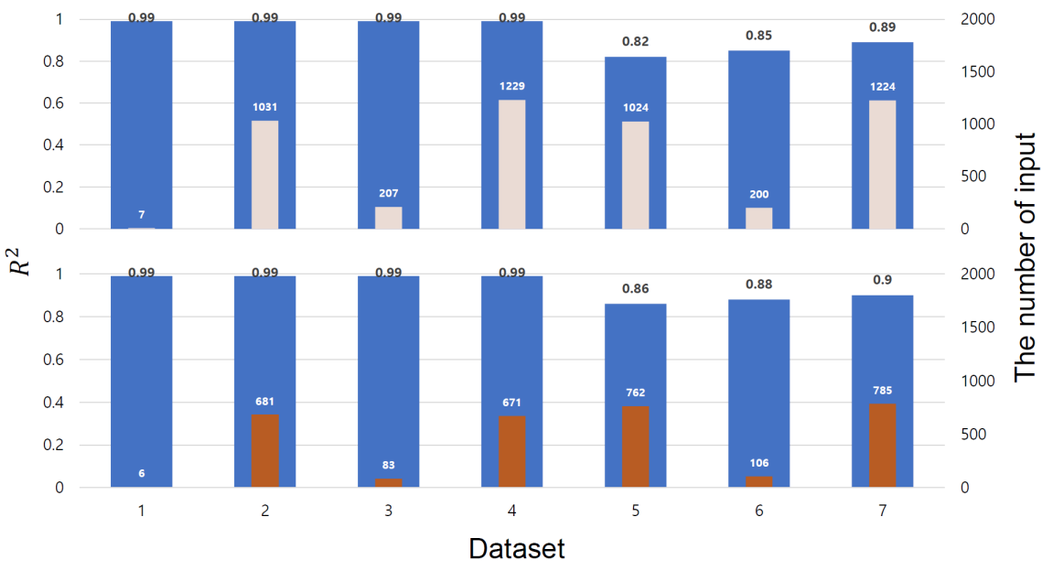


Figure 10: Performance of the MLP models with the different sets. The upper side presents the performance of the MLP model, and the lower side presents that of the Lasso-MLP model. The blue bars represent test R^2, the pink bars the total number of input features, and the orange bars the number of features with non-zero weights obtained from Lasso

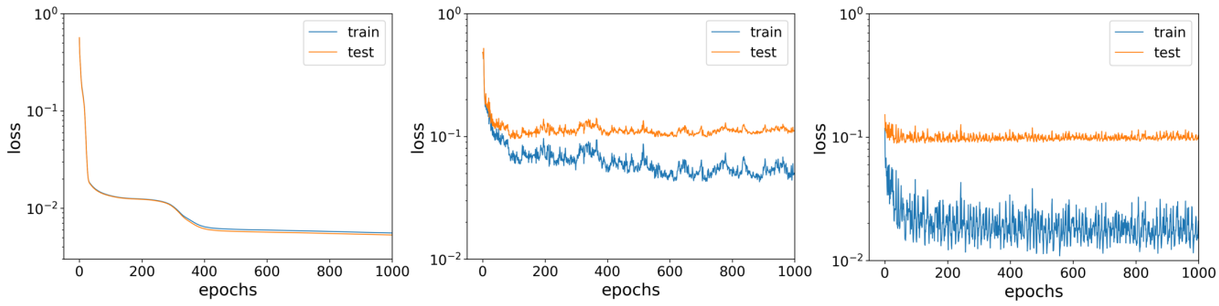


Figure 11: Learning curves of MLP models and Lasso-MLP model with/without **key features** (a) MLP model with set 1; (b) MLP model with set 7; (c) Lasso-MLP model with set 7

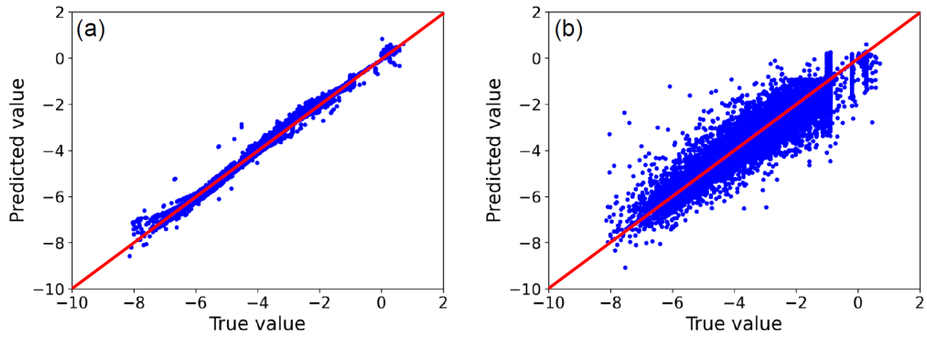


Figure 12: The predictability of the MLP models with/without key dataset. (a) MLP model with set 1; (b) Lasso-MLP model with set 7

The R^2 values of MLP and Lasso-MLP models fed by different input are shown in Fig. 10. With the **key features** included (sets 1-4 in the upper half of Fig. 10), MLP model show a high accuracy to predict the permeability, where R^2 are all 0.99. Although there are only 7 features in set 1, MLP model predicts the permeability accurately. It indicates that MLP model can serve as a precise model for the prediction of permeability when only the **key features** are known. The R^2 values for set 2-4 are 0.99 each, as R^2 would increase or at least not decrease as the number of independent variables increases. Without **key features** as input, the performance of MLP model drops to below 0.89 (set 5-7 in upper half of Fig.10). So with the MLP model, there is considerable overfitting if there are inputs other than the 7 **key features**. This was not captured by the Lasso model and is an important result. In addition, we found that three features: ΔG_M, apK_a and bpK_a (Col 4-6 in Fig. 1) out of seven in set 1 contribute the most to the performance of MLP model when training, echoing with the results of Lasso (Fig. 9). Similarly, MLP model applied to set 7 (fingerprints and molecular descriptors) has a higher R^2 than set 5 (fingerprints), proving again that R^2 would increase when adding more independent variables.

The Lasso-MLP model is just the MLP model applied to the features with non-zero weights obtained from Lasso and its performance is shown in the lower half of Fig. 10. Overall, the performance trends are similar to the ones for MLP, with differences reflecting in the increased R^2 values for input sets 5-7. This shows that the Lasso-MLP model performs better than only MLP in that it uses less number of features and still gives a higher accuracy.

Fig. 11 depicts some representative learning curves of MLP and Lasso-MLP models with/without **key features**. In Fig. 11(a), the goodness of fit of the MLP model using **key features** is presented by two points: 1) loss functions of both training and test sets decrease rapidly within small epochs and then be stable as the number of iterations increases; 2) the gap of learning curves between the training set and test set is small. The fits of MLP and Lasso-MLP model using input set 7 shown in Fig. 11(b) and (c), respectively, though not ideal, are acceptable. Overall, the MLP and Lasso-MLP models with/without **key features** are well-trained.

Fig. 12 demonstrates the predictability of MLP and Lasso-MLP models with/without **key features**. A high correlation between **key features** and permeability in MLP model (Fig. 12(a)), and that between the features of fingerprints & molecular descriptors with non-zero weights obtained from Lasso and permeability in Lasso-MLP model (Fig. 12(b)) are seen. The predictability of MLP model when only the **key features** were used as input, is more accurate. Agreed that, both the models can perform really well with only 7 features, but they can also predict the permeability fairly well only by using features with non-zero weights extracted from a SMILES string. This can be seen from the high R^2 value of 0.9 in set 7 of the lower half of Fig. 10 and it presents a way to predict permeability without finding the **key features** by expensive experimentation/computations.

5 Conclusion

In this work, we present the prediction of the membrane permeability based on several choices of the input dataset, using Lasso, MLP, and the Lasso-MLP machine learning algorithms. The Lasso models show 0.80 of R^2 with all inputs (the key properties (ΔG and pKa), molecular descriptor, and molecular fingerprints), and 0.72 of R^2 with the molecular descriptor and molecular fingerprints inputs. Since the permeability equation contains non-linearity, such a linear classifier shows its limitation despite the inclusion of ΔG and pKa. Also, the number of non-zero weights is saved at each set to feed these as input to the Lasso-MLP model.

Interestingly, the MLP models show 0.99 of R^2 with only the seven key feature inputs (55 parameters are used), suggesting that the MLP model can predict permeability very accurately when the key properties are given. Without the key properties dataset but with the molecular descriptor and fingerprint inputs, the MLP model shows 0.89 of R^2, a higher score than any Lasso model. The Lasso-MLP model with 64% of the original molecular descriptor and fingerprint inputs shows a little increased R^2, 0.90. In addition, all ML algorithms used in the work suggest that the models with the molecular descriptor input show better accuracy than those with molecular fingerprint input.

We believe that the less accurate prediction of MLP-based models originates from the nature of the permeability data (the output). The permeability values are obtained from coarse-grain molecular dynamics simulations, and each small molecule, ranging from 30 to 160 Da, is simplified to one MARTINI bead. Therefore, the number of unique values of permeability data is inherently limited. The data shows 838

distinct values out of 91372 samples. With the proper mapping from the all-atom to the coarse-grained model (the standard is up to 4:1), improved results can be anticipated.

Overall, our work indicates that adding **key features** including ΔG and pKa significantly affects the accuracy of the prediction models of all ML algorithms. Despite its importance, MLP and Lasso-MLP models predict the permeability fairly well by only using features simply extracted from a SMILE string. This finding may suggest a way to calculate permeability accurately without quantifying such expensive ΔG and pKa properties experimentally or computationally.

References

[1] RDKit: Open-source cheminformatics. https://www.rdkit.org.

[2] Axel Andrés et al. "Setup and validation of shake-flask procedures for the determination of partition coefficients (log D) from low drug amounts". In: *European Journal of Pharmaceutical Sciences* 76 (2015), pp. 181–191.

[3] Axel Andrés et al. "Setup and validation of shake-flask procedures for the determination of partition coefficients (logD) from low drug amounts". In: *European Journal of Pharmaceutical Sciences* 76 (2015), pp. 181–191. ISSN: 0928-0987. DOI: https://doi.org/10.1016/j.ejps.2015.05.008. URL: https://www.sciencedirect.com/science/article/pii/S0928098715002043.

[4] Simon Axelrod et al. "Learning matter: Materials design with machine learning and atomistic simulations". In: *Accounts of Materials Research* 3.3 (2022), pp. 343–357.

[5] Jörg Behler. "Neural network potential-energy surfaces in chemistry: a tool for large-scale simulations". In: *Physical Chemistry Chemical Physics* 13.40 (2011), pp. 17930–17955.

[6] Jörg Behler and Michele Parrinello. "Generalized neural-network representation of high-dimensional potential-energy surfaces". In: *Physical review letters* 98.14 (2007), p. 146401.

[7] Tristan Bereau and Kurt Kremer. "Automated parametrization of the coarse-grained Martini force field for small organic molecules". In: *Journal of chemical theory and computation* 11.6 (2015), pp. 2783–2791.

[8] Timothy S Carpenter et al. "A method to predict blood-brain barrier permeability of drug-like compounds using molecular dynamics simulations". In: *Biophysical journal* 107.3 (2014), pp. 630–641.

[9] Guang Chen, Zhiqiang Shen, and Ying Li. "A machine-learning-assisted study of the permeability of small drug-like molecules across lipid membranes". In: *Phys. Chem. Chem. Phys.* 22 (35 2020), pp. 19687–19696. DOI: 10.1039/D0CP03243C. URL: http://dx.doi.org/10.1039/D0CP03243C.

[10] Connor W Coley et al. "SCScore: synthetic complexity learned from a reaction corpus". In: *Journal of chemical information and modeling* 58.2 (2018), pp. 252–261.

[11] Felix A Faber et al. "Prediction errors of molecular machine learning models lower than hybrid DFT error". In: *Journal of chemical theory and computation* 13.11 (2017), pp. 5255–5264.

[12] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent". In: *Journal of statistical software* 33.1 (2010), p. 1.

[13] Rafael Gómez-Bombarelli et al. "Design of efficient molecular organic light-emitting diodes by a high-throughput virtual screening and experimental approach". In: *Nature materials* 15.10 (2016), pp. 1120–1127.

[14] Christian Hoffmann et al. "Controlled exploration of chemical space by machine learning of coarse-grained representations". In: *Phys. Rev. E* 100 (3 Sept. 2019), p. 033302. DOI: 10.1103/PhysRevE.100.033302. URL: https://link.aps.org/doi/10.1103/PhysRevE.100.033302.

[15] Takuya Inokuchi, Ryosuke Okamoto, and Noriyoshi Arai. "Predicting molecular ordering in a binary liquid crystal using machine learning". In: *Liquid Crystals* 47.3 (2020), pp. 438–448.

[16] Wooseong Kim et al. "A selective membrane-targeting repurposed antibiotic with activity against persistent methicillin-resistant Staphylococcus aureus". In: *Proceedings of the National Academy of Sciences* 116.33 (2019), pp. 16529–16534.

[17] Sunyoung Kwon et al. "Comprehensive ensemble in QSAR prediction for drug discovery". In: *BMC bioinformatics* 20.1 (2019), pp. 1–12.

[18] Christopher T Lee et al. "Simulation-based approaches for determining membrane permeability of small compounds". In: *Journal of chemical information and modeling* 56.4 (2016), pp. 721–733.

[19] Siewert J Marrink et al. "The MARTINI force field: coarse grained model for biomolecular simulations". In: *The journal of physical chemistry B* 111.27 (2007), pp. 7812–7824.

[20] Roberto Menichetti, Kiran H Kanekal, and Tristan Bereau. "Drug–membrane permeability across chemical space". In: *ACS central science* 5.2 (2019), pp. 290–298.

[21] Roberto Menichetti et al. "In silico screening of drug-membrane thermodynamics reveals linear relations between bulk partitioning and the potential of mean force". In: *The Journal of Chemical Physics* 147.12 (2017), p. 125101.

[22] Benjamin Meyer et al. "Machine learning meets volcano plots: computational discovery of cross-coupling catalysts". In: *Chemical science* 9.35 (2018), pp. 7069–7077.

[23] Zeinab Mozafari, Mansour Arab Chamjangali, and Mohammad Arashi. "Combination of least absolute shrinkage and selection operator with Bayesian Regularization artificial neural network (LASSO-BR-ANN) for QSAR studies using functional group and molecular docking mixed descriptors". In: *Chemometrics and Intelligent Laboratory Systems* 200 (2020), p. 103998.

[24] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[25] Pavel G Polishchuk, Timur I Madzhidov, and Alexandre Varnek. "Estimation of the size of drug-like chemical space based on GDB-17 data". In: *Journal of computer-aided molecular design* 27.8 (2013), pp. 675–679.

[26] Raghunathan Ramakrishnan et al. "Electronic spectra from TDDFT and machine learning in chemical space". In: *The Journal of chemical physics* 143.8 (2015), p. 084111.

[27] David Rogers and Mathew Hahn. “Extended-connectivity fingerprints”. In: *Journal of chemical information and modeling* 50.5 (2010), pp. 742–754.

[28] Wataru Shinoda. “Permeability across lipid membranes”. In: *Biochimica et Biophysica Acta (BBA)-Biomembranes* 1858.10 (2016), pp. 2254–2265.

[29] Lloyd R Snyder, Joseph J Kirkland, and Joseph L Glajch. *Practical HPLC method development*. John Wiley & Sons, 2012.

[30] Igor V Tetko and Vsevolod Yu Tanchuk. “Application of associative neural networks for prediction of lipophilicity in ALOGPS 2.1 program”. In: *Journal of chemical information and computer sciences* 42.5 (2002), pp. 1136–1145.

[31] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[32] Oliver T Unke and Markus Meuwly. “PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges”. In: *Journal of chemical theory and computation* 15.6 (2019), pp. 3678–3693.

[33] David M Wilkins et al. “Accurate molecular polarizabilities with coupled cluster theory and machine learning”. In: *Proceedings of the National Academy of Sciences* 116.9 (2019), pp. 3401–3406.

[34] Kevin Yang et al. “Analyzing learned molecular representations for property prediction”. In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3370–3388.

[35] Alex Zhavoronkov et al. “Deep learning enables rapid identification of potent DDR1 kinase inhibitors”. In: *Nature biotechnology* 37.9 (2019), pp. 1038–1040.