Learned Feature Generation for Molecules

Patrick Winter[1,2], Christian Borgelt[1,3], and Michael R. Berthold[1,2,4]

[1] Department of Computer and Information Science, University of Konstanz, 78457 Konstanz, Germany
{patrick.winter,christian.borgelt,michael.berthold}@uni-konstanz.de
[2] Konstanz Research School Chemical Biology (KoRS-CB), Konstanz, Germany
[3] Department of Computer Science, Otto-von-Guericke University, 39106 Magdeburg, Germany
[4] KNIME AG, 8005 Zurich, Switzerland

Abstract. When classifying molecules for virtual screening, the molecular structure first needs to be converted into meaningful features, before a classifier can be trained. The most common methods use a static algorithm that has been created based on domain knowledge to perform this generation of features. We propose an approach where this conversion is learned by a convolutional neural network finding features that are useful for the task at hand based on the available data. Preliminary results indicate that our current approach can already come up with features that perform similarly well as common methods. Since this approach does not yet use any chemical properties, results could be improved in future versions.

Keywords: Convolutional neural networks · Feature generation · Molecular features · Virtual screening.

1 Introduction

High-throughput screens [5] are large-scale, biological experiments to find molecules that show a desired biological activity. Even though they are mostly automated, they are still expensive and time consuming. For this reason, machine learning methods are used for virtual screening to select a subset of molecules that are most likely to show activity. This is done by formulating a binary classification problem with the classes *active* and *inactive*. A diverse subset is tested in the lab and the results are used as training data for the classifier. The molecules with unknown activity are then classified, and the probability of a molecule belonging to the active class is assumed to be the probability of the molecule showing actual activity. Based on this the top-n molecules are picked for actual testing in the lab, thus reducing the number of actual tests to be conducted.

Most classifiers need numerial features to work. In such cases, the molecular structure gets converted into numerical features using a feature generator. The most common feature generators for molecules are based on a static algorithm that creates the same output for the same molecules without taking the specific

classification task into account. Once the features have been created a classifier is learned to distinguish active molecules from inactive ones.

Dynamic approaches, that generate features for a specific classification task like substructure mining [11] do also exist. Here substructures are selected based on their frequency and how well they discriminate between the different classes.

The method that we propose uses a network that uses convolutions to generate features from the molecules structure and then classifies based on these features using dense layers. By training the feature generation and classification together, the feature generation will learn features which are useful for the specific classification task. These features could potentially outperform handcrafted features for the task that they are built for.

1.1 Fingerprints

The most common approach to feature generation for molecules is the use of fingerprints [23]. These fingerprints are built using domain knowledge. A simple example for a fingerprint is the MACCS fingerprint [6], which represents 166 predefined aspects of the molecule's structure as a bit vector. A different approach can be seen in circular fingerprints, such as the extended connectivity fingerprint [19], which encodes the occurrence of different substructures in the molecule as a bit or counting vector (see Section 2.1 for details). Many years of research and extensive expert knowledge went into the creation of many different fingerprints. Therefore the selection of the best fingerprint for a specific problem is not obvious.

Riniker et al. created a benchmark [18] comparing 14 different fingerprints on a variety of data sets. The results showed that the top 12 fingerprints had no significant difference on average, even though their performance on individual data sets did differ. This indicates that there is no gold standard fingerprint that can be relied upon to give the best performance most of the time. Since the features given to the classifier determine how well it is able to distinguish between the classes, it would be desirable if those features not be based on a static decision as to which feature generator to use, but instead were learned automatically based on the task that needed to be solved.

1.2 Image Processing

Approaches for automatically learning useful features for images using convolutional neural networks [14] have been around for a while. But it was only after a convolutional neural network won the ImageNet challenge in 2012 [12] and fast implementations, especially those that utilize graphics processing units (GPUs), became available, that these networks started replacing the old methods that used handcrafted features [16].

These convolutional neural networks take the RGB values of the image as input with little to no preprocessing. They then learn convolutional layers that abstract this input into features that are useful to the classification that is performed by dense layers at the end. In this way decisions on how to best generate

useful features are made, based on what the classifier needs in order to improve the separation of the classes.

To understand more about what a neural network has actually learned, there are multiple methods. One of them is the class activation map [25], which visualizes the patterns in an image that were responsible for the predicted class.

2 Related Work

Although learning the feature generation is now commonplace for images, this has not yet been the case for molecular structures, where the use of (handcrafted) fingerprints is still the most common approach. Even many approaches using neural networks use them for classification only and still use molecular descriptors and fingerprints as input (e.g., [15, 24, 17]).

Some work has been done to use graph neural networks [2, 10] for learning on molecules. In this work however we focus on the use of traditional, grid based convolution networks similar to the ones used in image processing. This way we can build on the extensive research done in this field.

2.1 Fingerprint Examples

Common molecular fingerprints rely heavily on human expert knowledge. For example, the MACCS [6] fingerprint is based on a list of 166 manually selected aspects of a molecule's structure. The presence or absence of each aspect is then checked for each molecule and represented in that molecule's fingerprint as a bit. The aspects are based on domain knowledge and assumed to be especially descriptive of a molecule's behavior.

Another approach to fingerprints is the extended connectivity fingerprint. It is based on the idea of encoding the occurrence of specific substructures into the fingerprint. For the encoding, the algorithm iterates over each heavy atom in the molecule's structure and looks at the properties of all the atoms contained in a given radius around this center atom. Just which properties are computed is configurable. The properties are then hashed into a single value in the range of 1 to n with n being the length of the generated fingerprint. This value is now used as the position in the fingerprint for substructures with these aspects. The value at this position is set to 1 for binary fingerprints or counted up by 1 for counting fingerprints. The problem with the extended connectivity fingerprint is that multiple different substructures can end up with the same hash value. As a consequence different substructures can end up setting the same bit: two different substructures can thus appear to be the same. The fingerprint is also dependent on selecting the right parameters for the radius, the measured molecular properties, and the length of the fingerprint.

2.2 Neural Networks

Convolutional neural networks are an approach for learning features. They consist of a collection of different layer types. The earlier part of the network learns

how to convert the input into useful features and the later part learns how to classify the data based on the generated features.

Neural networks have a tendency to overfit the training data. To counter this, dropout layers [22] can be used. During training they randomly deactivate a specified amount of neurons to force the network to work with the remaining information instead of zeroing in on the most prominent ones and ignoring other opportunities. This leads to a more robust network.

Convolutional layers implement a sliding window over the data and thus learn to abstract the data in a local area. The size of the window and the number of filters per position as well as the step size can be configured and need to fit the problem. Since the same weights are used in all positions, they also implement position invariance.

Often neural networks are applied to problems with a large amount of input neurons and since convolutional layers are usually used to create an increasing amount of output features per position, the network tends to get very big. In order to keep the number of neurons per layer down, max pooling layers downsample the input data, keeping only the most prominent information. This is based on the idea that the presence of patterns is more important than their exact location and information about the most dominant patterns is sufficient.

In a dense layer every neuron is fully connected to every neuron in the previous layer. Dense layers are used to learn a classifier and are therefore usually at the end of a network. A typical use of dense layers are multi-layer perceptrons. They consist of the input layer, a number of hidden dense layers, and a dense layer as output.

In order to understand why a trained network assigns a certain class to a specific image, class activation maps [25] can be used. They visualize the recognized patterns associated with the chosen class in the input image by highlighting the pixels most responsible for the high value in the output neuron for the winning class. This is done by back propagating how much each previous neuron contributed to the activation of a selected neuron. When this is done through the entire network, a heat map is created over the input dimensions. The class activation map can be a useful tool for seeing how the network actually learns the expected patterns and if the network is functioning as desired.

3 Learned Feature Generation

Our new method adopts the ideas from the field of image processing and tries to modify them for use with molecular structures. We are in a similar situation in that we have no universal best method to generate useful features and therefore an approach of letting a network learn which features benefit the specific classification task most seems to be a viable option.

As illustrated in Figure 1 we replace the static feature generator and the classifier with one network. This network learns how to generate useful features in the first part and how to classify the data in the second part. In this way the gen-

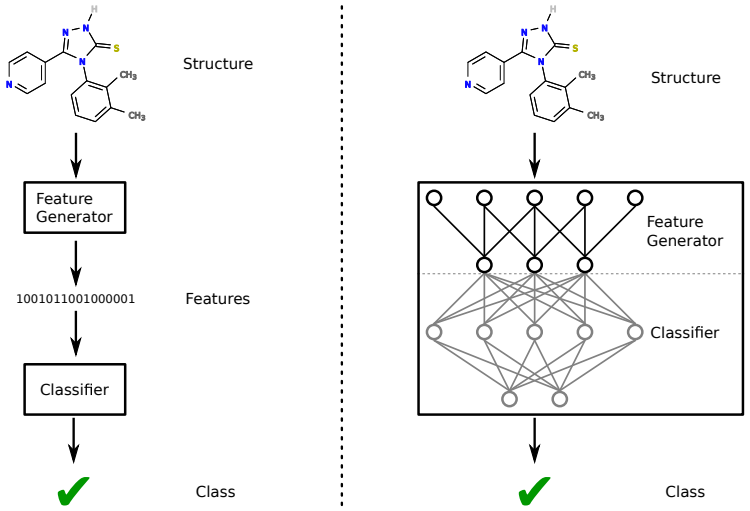


Fig. 1. Classical method (left): A feature generator converts the structure into numerical features. The numerical features are then used to train a classifier. Demonstrated method (right): Feature generation and classification are both done by one neural network. Features are learned by the first part of the network and the classification is learned by the second part.

erated features are learned based on what is useful for the specific classification task.

3.1 Preprocessing

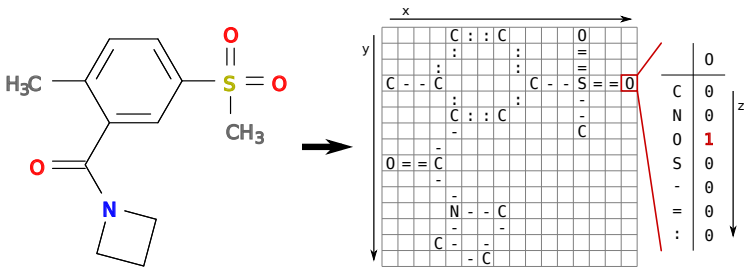


Fig. 2. Grid based data representation (right) using the layout of the 2D renderer (left). Each cell is encoded using a one-hot array resulting in a 3D tensor with 2 dimensions (x and y) for the position and 1 dimension (z) for the features at this position.

As with most other machine learning methods, a neural network needs its input data to be in a numerical format. However, the strength of neural networks

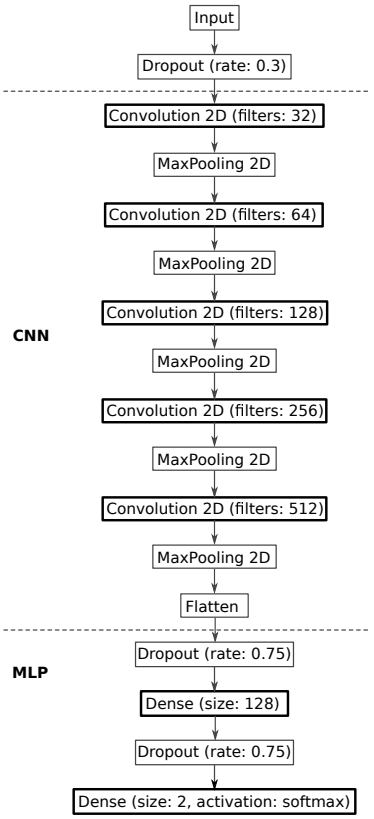
is that the input is allowed to be in a format, which, by itself, does not represent a good abstraction of the content of the data. This abstraction into a useful representation is learned by the convolutional network. The current approach (see Figure 2) encodes the structure into a 2-dimensional grid containing characters that represent the atoms and the bonds between them. Instead of the RGB values for each pixel in an image, every cell is encoded by a one-hot array which marks what character is located at this position. This one-hot array is based on a global dictionary containing all possible characters. If no character is present in the cell, then no bit will be set. The position for each atom is obtained by using the layout engine of the RDKit [13] renderer that is normally used to render molecules as images. This provides a representation of the molecule that is close to a 2D rendered image of the molecule but in a machine readable format. Since atom symbols are directly encoded with single bits instead of a collection of pixels that form the symbol's character we remove the need for the network to reconstruct this information back. In addition we can keep the grid smaller for more performant computation.

Because screening data usually has highly imbalanced classes we oversample the minority class in the training data to learn on an equal distribution of classes. The oversampled data are then shuffled to prevent the network from training too much of a single class in succession. Before training, the data are transformed using rotation and flipping, similar to what is done with images. Each transformation yields a valid representation for the same molecule. As a result, even the oversampled data is presented in many different ways instead of using the same representation of the same molecule multiple times. Training on the transformed data also gives the network a chance to learn rotation invariance. This is important, since the same substructure, in different molecules, can occur in different positions (position invariance handled by the convolutional layers) and differently rotated (rotation invariance handled by learning on differently transformed data).

The transformation is performed by randomly rotating the molecule around the center and then randomly flipping it vertically. These transformations are performed on the original coordinates before being fit into the smaller grid. In some cases a small rotation only moves a single atom in the grid, since it was the only one that passed the threshold into another cell during downsampling. This effect can also occur when the same substructure is contained in a different model and is therefore in a different position and differently rotated. That is why it is important to teach the network tolerance with regard to these smaller shifts. The parameters of the transformation are chosen randomly, with a rotation of 0–359 degrees and with or without flipping. Since different parameters can result in the same representation (not every rotation by one additional degree has an effect) uniqueness is not ensured.

3.2 Network Architecture

The network architecture (Figure 3) is inspired by the structure of VGG networks [21]. The input layer is followed by a dropout layer with a dropout rate



initializer	He uniform
activation	ReLU
window	3x3
padding	same
pool size	2x2
pool stride	2
loss	binary crossentropy
optimizer	Adam
learning rate	0.0001

Fig. 3. Architecture of the used network. The convolutional part of the network (CNN) learns the generation of features while the multi-layer perceptron (MLP) at the end learns the classification.

of 30% to counter overfitting. For feature generation we have 5 blocks of a convolution and a max pooling layer each. The convolutions generate an increasing amount of features while the max pooling downscales the resolution of the data. This way we increasingly transform the low information density with high locality into high information density and very low locality. After a flatten layer that converts the output of the convolutional part of the network into 1 dimension, we obtain the features that are used for classification. A multi-layer perceptron with one hidden layer and an output layer goes on to perform the classification based on these features. The multi-layer perceptron also uses dropout layers with a dropout rate of 75% to increase generalisation. Since the back propagation goes through the entire network, the classifier can influence which features are learned by the convolutional part of the network.

Once the network has been fully trained it can either be used as a whole to perform feature generation and classification together, or otherwise only the convolutional part is used to generate the features. In the latter case the output

of the flatten layer is used as the features. These features can then also be used to work with different classifiers like a random forest.

4 Preliminary Results

In order to evaluate our method we ran two experiments. The purpose of the first one was to check if our network can recognize patterns in the data as expected. We did this by using class activation maps. In the second experiment we compared the classification performance with the performance of existing fingerprints on real world data sets.

4.1 Learned Patterns

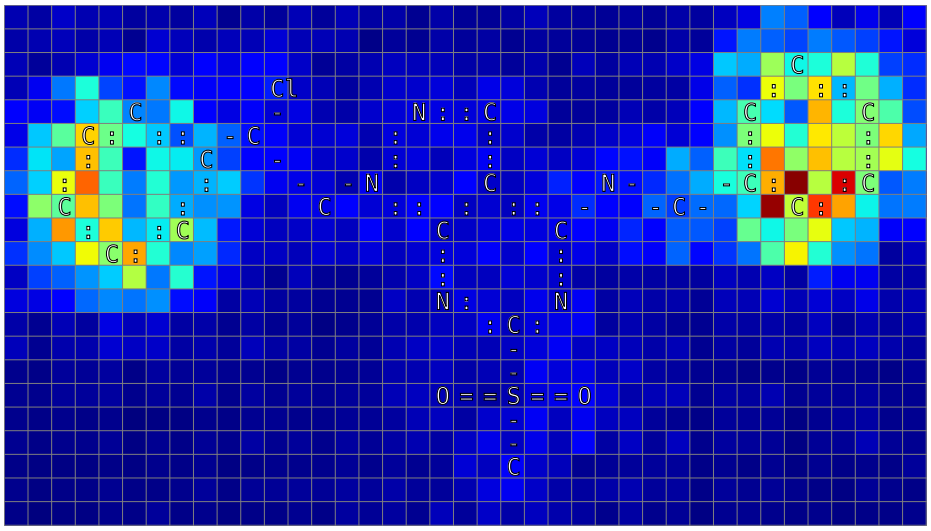


Fig. 4. Class activation map showing the patterns that are responsible for classification. Warmer colors (red > yellow > green > blue) represent a higher importance of a cell to the classification task. We can clearly see that the contained benzene rings are the reason why this molecule was classified as class A.

A data set was split into molecules that either contain a benzene ring (class A) or not (class B). The network then had to learn this classification and would hopefully learn the pattern that was responsible for the split purely on the class information. Looking at the class activation maps for the molecules that were classified as class A (example in Figure 4) we can visually verify that the network picked up the correct pattern, as intended. Looking at the mean activation values for atoms that are part of a benzene ring and atoms that are not we were also able to see a considerable difference (see Figure 5).

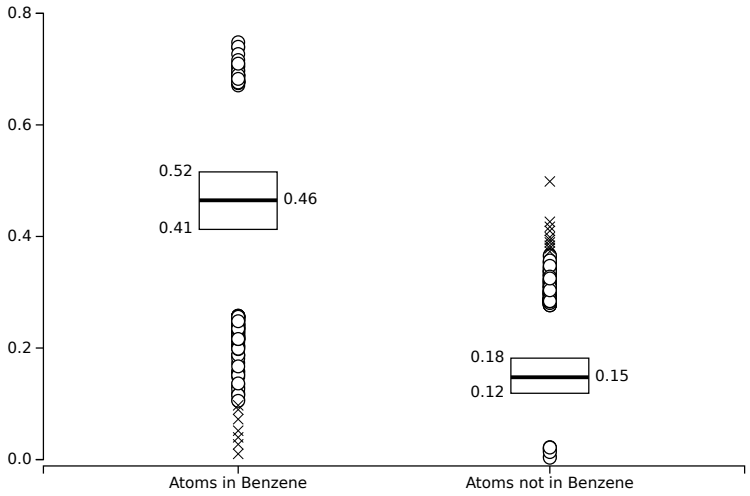


Fig. 5. Activation values of class activation maps for atoms that are contained in the benzene ring substructure responsible for classification and for atoms that are not contained in a benzene ring.

4.2 Benchmark

Table 1. Number of times a method obtained a certain rank in comparison to the other methods.

	ROC Curve AUC				Enrichment Factor at 5%			
Rank	CNN	ECFC0	ECFP4	MACCS	CNN	ECFC0	ECFP4	MACCS
1	16	1	50	21	23	1	57	7
2	19	14	27	28	27	8	24	29
3	20	34	9	25	19	29	4	36
4	33	39	2	14	19	50	3	16

In order to evaluate the performance on real-world data sets we used the data assembled by Riniker et al. [18] to benchmark different fingerprints. We compared our method (CNN) against 3 fingerprints. The binary extended connectivity fingerprint with a diameter of 4 (ECFP4), the counting extended connectivity fingerprint with a diameter of 0 (ECFC0) and the MACCS fingerprint (MACCS). As classifier we used a random forest. The metrics used for evaluation are the ROC curve AUC [4] and the enrichment factor [8] at 5% as suggested by Riniker et al. [18]. The ROC curve AUC measures the performance of the prediction on the entire data set sorted by probability of belonging to the active class. The enrichment factor at 5% is based on how many more active molecules are found in the top 5% of the sorted predictions in comparison to random selection.

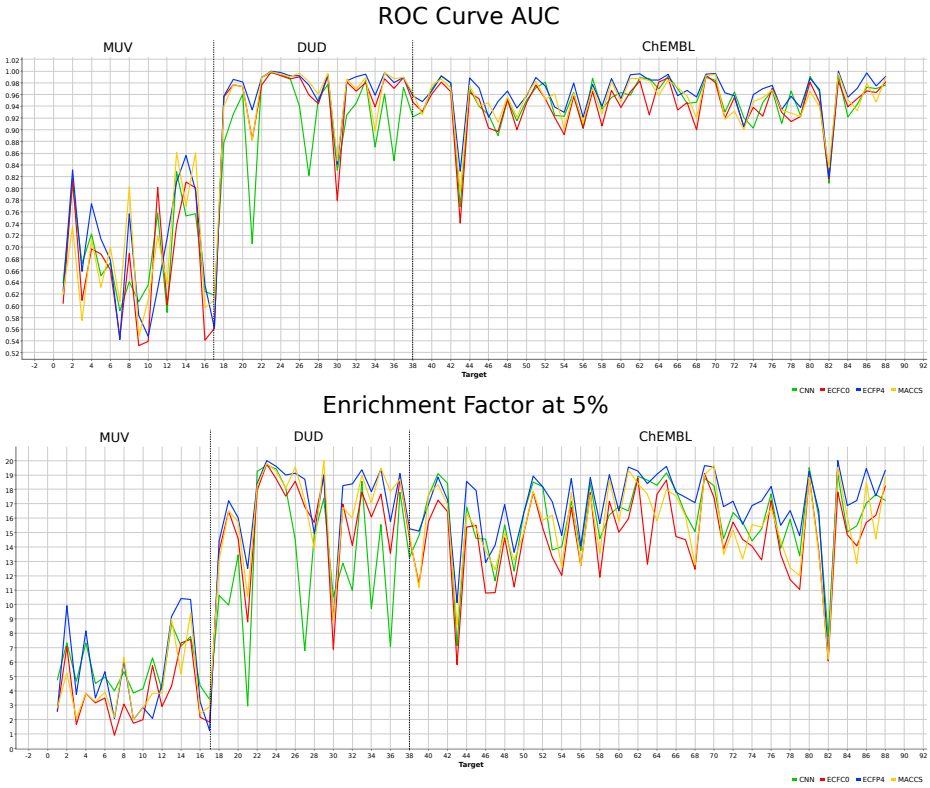


Fig. 6. Results comparing the CNN features with the MACCS, ECFC0 and ECFP4 fingerprints. The ROC Curve AUC (top) measures the performance off the entire prediction while the enrichment factor at 5% (bottom) measures the early recognition.

The data contains 88 single data sets. The data sets come from 3 sources: 17 are from the maximum unbiased validation (MUV) data sets [20], 21 from the directory of useful decoys (DUD) [9, 3], and 50 from the ChEMBL [7, 1] database. Each data set consists of 1,344–15,560 inactive and 30–365 active molecules. 20% of the data was sampled via stratified sampling to create a training set. The remaining 80% were used for testing.

The grid size of the preprocessed data was automatically selected so that all molecules in the specific data set will fit into it. The same is true for the dictionary of characters where only characters that are present in the data set have an index in the one hot-array.

For the neural network we oversampled and shuffled the training data. We trained the network for 100 epochs with different random seeds for the transformation in every epoch. In this way the network could only see the same molecule with the same representation if the transformation, by random chance, was done with the same or very similar parameters.

In order to compare only the performance of the learned features with the fingerprints without the performance difference in classifiers, we extracted the features from the trained networks. We then trained a random forest for each fingerprint and also for the features generated by the network. Each random forest had 10,000 trees. We used soft voting and a minimum leaf size of 10 to retrieve a fine granular class probability for sorting.

We ran every experiment 10 times and used the mean as result. Figure 6 shows the results for both the entire prediction as well as for early recognition. Table 1 shows how well the method compare against each other. The results indicate that the CNN features perform similarly well as the fingerprints. Considering how much expert knowledge had to be put into the creation of the fingerprints, this is already an achievement.

5 Conclusion and Future Work

We created a method that represents a molecule's structure as a 2D grid and uses a convolutional neural network to convert this representation into a set of features that are useful for the learned classification task. Using class activation maps we were able to see that the network was actually able to recognize the pattern responsible for the class in a generated data set. In a bigger evaluation on 88 data sets we were able to achieve results similar to fingerprints. Considering how many years of research and how much expert knowledge went into the creation and refinement of these fingerprints, this is already a promising result.

Our next step is to add chemical properties to the input data. This would give the network the opportunity to also learn something about the chemistry of the molecules, and thus should end up in a boost to the classification performance.

Acknowledgements. This work was partially funded by the Konstanz Research School Chemical Biology and KNIME AG.

References

1. ChEMBL, https://www.ebi.ac.uk/chembl/
2. Deepchem, https://deepchem.io/
3. DUD - A Directory of Useful Decoys, http://dud.docking.org/
4. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition **30**(7), 1145–1159 (1997)
5. Broach, J.R., Thorner, J., et al.: High-throughput screening for drug discovery. Nature **384**(6604), 14–16 (1996)
6. Durant, J.L., Leland, B.A., Henry, D.R., Nourse, J.G.: Reoptimization of mdl keys for use in drug discovery. Journal of chemical information and computer sciences **42**(6), 1273–1280 (September 2002)
7. Gaulton, A., Bellis, L.J., Bento, A.P., Chambers, J., Davies, M., Hersey, A., Light, Y., McGlinchey, S., Michalovich, D., Al-Lazikani, B., et al.: Chembl: a large-scale bioactivity database for drug discovery. Nucleic acids research **40**(D1), D1100–D1107 (2011)

8. Halgren, T.A., Murphy, R.B., Friesner, R.A., Beard, H.S., Frye, L.L., Pollard, W.T., Banks, J.L.: Glide: a new approach for rapid, accurate docking and scoring. 2. enrichment factors in database screening. Journal of medicinal chemistry **47**(7), 1750–1759 (2004)
9. Irwin, J.J.: Community bbenchmark for virtual screening. Journal of computer-aided molecular design **22**(3-4), 193–199 (2008)
10. Kearnes, S., McCloskey, K., Berndl, M., Pande, V., Riley, P.: Molecular graph convolutions: moving beyond fingerprints. Journal of computer-aided molecular design **30**(8), 595–608 (2016)
11. Klopman, G.: Artificial intelligence approach to structure-activity studies. computer automated structure evaluation of biological activity of organic molecules. Journal of the American Chemical Society **106**(24), 7315–7321 (1984)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
13. Landrum, G.A., et al.: RDKit: Open-source cheminformatics. https://www.rdkit.org/ (2006)
14. Le Cun, Y., Matan, O., Boser, B., Denker, J.S., Henderson, D., Howard, R., Hubbard, W., Jacket, L., Baird, H.: Handwritten zip code recognition with multilayer networks. In: Pattern Recognition, 1990. Proceedings., 10th International Conference on. vol. 2, pp. 35–40. IEEE (1990)
15. Mayr, A., Klambauer, G., Unterthiner, T., Hochreiter, S.: Deeptox: toxicity prediction using deep learning. Frontiers in Environmental Science **3**, 80 (2016)
16. Nixon, M.S., Aguado, A.S.: Feature extraction & image processing for computer vision. Academic Press (2012)
17. Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., Pande, V.: Massively multitask networks for drug discovery. arXiv preprint arXiv:1502.02072 (2015)
18. Riniker, S., Landrum, G.A.: Open-source platform to benchmark fingerprints for ligand-based virtual screening. Journal of Cheminformatics **5**(1), 26 (May 2013)
19. Rogers, D., Hahn, M.: Extended-connectivity fingerprints. Journal of chemical information and modeling **50**(5), 742–754 (April 2010)
20. Rohrer, S.G., Baumann, K.: Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. Journal of chemical information and modeling **49**(2), 169–184 (2009)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
22. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
23. Todeschini, R., Consonni, V.: Handbook of molecular descriptors, vol. 11. John Wiley & Sons (2008)
24. Unterthiner, T., Mayr, A., Klambauer, G., Steijaert, M., Wegner, J.K., Ceulemans, H., Hochreiter, S.: Deep learning as an opportunity in virtual screening. In: Proceedings of the deep learning workshop at NIPS. vol. 27, pp. 1–9 (2014)
25. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2921–2929 (2016)