Big data and deep learning: extracting and revising chemical knowledge from data

Q1

The corrections made in this section will be reviewed and approved by master copier.

Q2 Giuseppina Gini[1], Chiakang Hung[1], Emilio Benfenati[2]

[1]~~Politecnico di Milano, DEIB Piazza Leonardo da Vinci, Milano, Italy~~Politecnico di Milano, DEIB, Piazza Leonardo da Vinci, Milano, Italy

[2]Laboratory of Environmental Chemistry and Toxicology, Istituto di Ricerche Farmacologiche Mario Negri IRCCS, Milano, Italy

5.1 Introduction

Computational toxicology is a large spectrum of methods aimed at assessing the toxicity/safety of chemical substances using computers and models instead of wet experiments. Computational toxicology is the result of two research lines: mathematical chemistry and the 3Rs (Replace, Reduce, Refine~~,~~) principle, defined in the fifties of the last century, and devoted to finding ways to reduce animal use in assessing chemicals.

Mathematical chemistry (Basak, 2013) and chemoinformatics, started with the definition of molecular structures and mainly worked on defining numerical chemical descriptors, that is, human-engineered features representing various characteristics such as geometry, topology, and energy distribution of molecules. In a few decades of research, thousands of molecular descriptors have been defined, usually in relation to the rapid development of predictive methods for toxicology and new molecule design (Todeschini and Consonni, 2009).

Chemical properties can be studied using physical models and simulation. Those methods are common for the design of new molecules, but are not affordable when the properties of interest are biological effects; in fact, knowledge about the living systems (for instance, the receptor and the mechanism used to cause the effect) is seldom available. For modeling biological effects, (quantitative) structure–activity relationship ((Q)SAR) models are commonly used. QSARs are based on the postulate that similar molecules exhibit similar physical and biological activit~~y~~ies (Johnson and Maggiora, 1990), and are the main nonphysical predictive models in use (Gini, 2016).

QSAR methods built on molecular descriptors appeared around the middle of the last century. Those models were initially simple regressions, using very few, and possibly simple, chemical descriptors. Choosing informative descriptors for the task at hand is a key aspect and requires deep insights into chemical and biological properties, as interactions between molecules, reactions and enzymes involved, and metabolic degradation of the molecules should be somehow represented.

Usually building QSARs starts with computing a large number of chemical descriptors, then applying a method to reduce them to a few, and adopting a computational technique to build the model that predicts new assays.~~.~~

SAR is instead based on the idea that specific functional groups, often defined by experts and called structural alerts (SAs), are responsible for molecular behavior, so their presence is checked to predict the activity. SAR and QSAR are usually mixed, as many QSAR methods use fingerprints, and often the explanation of QSAR models considers the functional subgroups too.

For regulatory purposes, toxicologists use expert-intensive methods like read-across, which are accepted, and often integrated into final decisions by panels of experts (Benfenati et al., 2019). In read-across, the target molecule is compared with a couple (or a few) of similar molecules for which the test has been performed, and the property value is predicted. The problem with this method is human bias; different experts select different similar molecules, or look for different substructures as responsible for the toxicity, or give a different weight to properties (Benfenati et al., 2016).

Human bias is reduced when using QSAR models that learn from a larger population of molecules. However, two factors make QSARs not widely accepted: the lack of confidence in statistical methods, and the shortage of data available that makes it problematic to cover the entire chemical space. Toxicity data are available for just a very small fraction of the entire chemical space of the molecules of biological interest. It has been estimated that this chemical Q3 space exceeds 10^{60} molecules (~~Kirkpatrik and Ellis, 2004~~Kirkpatrick and Ellis, 2004), while the toxicological characterization of chemicals is available for a few thousand molecules.

Laboratory tests are expensive, time-consuming, and in some cases forbidden by regulations. This situation pushes for the adoption of computational methods that make use of available data and knowledge and can integrate results from living organisms and cellular lines. Adopting (Q)SAR becomes a necessity, and making QSARs effective is more and more pursued.

Machine learning (ML) algorithms are being adopted to replace the statistical methods, as ML can be fully data-driven and does not need to make a hypothesis about the mathematical function that ~~better~~ at best explains the data. Connectionist models, based on distributed representation and parallel computation of the process, as neural networks (NNs), received attention and criticism in QSAR.

The situation is changing as large quantities of in vitro test values are made available, and they challenge traditional modeling methods. In the last decade very big NNs, called deep neural networks (DNNs) provided the top models in the Merck Kaggle challenge in 2012 and in the Tox21 challenge in 2014 for activity prediction. Using DNNs to solve some of the open issues in computational toxicology is a very active field.

There are still major challenges to obtain toxicology models that can replace animal experiments as follows:

- One challenge is to improve the process of constructing QSAR models. Big data are more often available, and the present way to build QSAR models takes too much time and human expertise. Instead of computing thousands of numerical descriptors and selecting a few ones as features, a simplified process would be to use directly as input the molecular structure and let the algorithm to automatically extract the features.

- Another challenge is to avoid, or to reduce, the bias of the expert that characterizes the SAR approach; usually the functional subgroups responsible for the activity are proposed by experts on the basis of the study of some mechanism of action. However, they are not a complete list and are not universally applicable. For instance, Toxtree (Benigni and Bossa, 2008) uses a few tens of functional subgroups as mutagenicity alerts, and indicates the percentage of their presence in toxic molecules; in some cases, the presence is less than 50%. The role of SAs in fact is to give the expert a reason for toxicity in case this appears, but other mechanisms at the organ or cellular level can make such alerts transformed into other chemicals before they act.

- The last challenge is to provide an explanation of the result obtained by the model (Gini, 2018). Considering that toxicology models are used in many processes and regulations, some kind of explanation to ground the result obtained into general expert knowledge is important. The common way of interpreting QSAR problems by a priori choosing simple and interpretable descriptors can work for explanatory QSARs, but not for predictive QSARS.

Other open problems to take into account are data availability and data quality. Open and public databases are often available. However, a lot of data are proprietary, often for the same endpoint results using different standards can be found, and different databases may contain different values for the same experimental test. This leads to extra efforts to create a reliable data set. In the following, the data set~~s~~ for predicting Ames test results ~~are~~is taken from ~~other publications~~the literature; it includes a large set of proprietary data provided by the Japan Ministry of Industry ~~as part of an~~ for the first Ames I~~i~~nternational challenge (Honma et al., 2019).

Section 5.2 presents the basic definitions of the DL architectures, which are special organizations of multilayers neural networks, such as recurrent neural network~~s~~ (RNN~~s~~), convolutional neural network (CNN), and graph convolutional neural network (GCN) (Wu et al., 2019) used to generalize CNN~~s~~ to handle graph data. How to use DNNs on chemical input is shortly reviewed in Section 5.3. Section 5.4 discusses a case study on mutagenicity models developed in the three before mentioned architectures. Section 5.5 shows how DL methods can be interpreted. The discussion continues in Section 5.6, where the results obtained are put into a perspective.

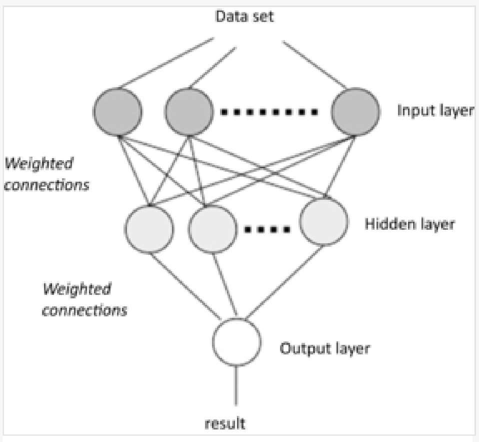
5.2 Basic methods in neural networks and deep learning

5.2.1 Neural networks

NNs are biologically inspired programming paradigms equivalent to a mathematical function that maps a given input to the desired output. A NN is made up of processing nodes, called neurons, arranged in layers; each neuron receives inputs and converts them to a single output, which is sent to the neurons of the next layer. The number of hidden layers determines the complexity of the function that the network can approximate; nets with one hidden layer can approximate any nonlinear function (Chen and Chen, 1995). Weights on the connections are learned to adjust the output of the network iteratively during training. In the basic feedforward architecture (Fig. 5.1) each neuron is connected to all the neurons of the next layer; the connections go in one direction, from input toward output neurons.

- The top layer, *input layer*, contains neurons that just pass the information to the next layer.
- The bottom layer, *output layer*, contains the output neurons, which give the result of the network computation.
- The middle layer, *hidden layer*~~; it~~, contains neurons that make intermediate processing, and then transfer the weights to the following layer. Usually, a bias neuron is added to each hidden layer.

Figure 5.1



A feedforward neural network. Neurons in each layer are fully connected through weighted connections to neurons in the next layer.

The output of a node is the result of applying the *activation function* $h(x)$ to its input. Activation functions should be continuous and continuously differentiable. Only nonlinear activation functions allow NNs to compute nontrivial functions.

Currently, the most used activation function is rectified linear unit (ReLU), as in Eq. (5.1)

$$\mathrm{Re}LU(x) = \max(0, x) \tag{5.1}$$

ReLU is not continuously differentiable. Its derivative is 0 for $x<0$, undefined for $x=0$, and 1 for $x>0$.

Leaky ReLU, Eq. (5.2), adds a small slope to the negative values to have a nonzero gradient.

Previous Version

$$\mathrm{LeakyRe}LU\,(x) = \text{if } x > 0\text{then } x,\ \text{else } 0.001\,(x)$$

Updated Version

$$\mathrm{LeakyRe}LU\,(x) = \text{if } x > 0 \text{ then } x,\ \text{else } 0.001\,(x) \tag{5.2}$$

5.2.2 Neural network learning

NNs are mainly supervised methods and are trained using labeled data to approximate the wanted function. During training, weights and thresholds are changed so to reduce the error on the training data.

NNs are usually trained using backpropagation of the gradient descent of the loss function with respect to the weights (Werbos, 1994). Gradient descent is an iterative optimization method that tries to minimize the objective function (error function) by changing the rate of inclination of a slope. It is applied many times using the training data, with the aim of finding the function that fits at best the data. Gradient descent with backpropagation is not guaranteed to find the global minimum of the error function, but only a local minimum.

The weights updates are done by Eq. (5.3)

$$W_{\mathrm{ij}}\,(t+1) = W_{\mathrm{ij}}\,(t) + \eta\left(\frac{\mathrm{dC}}{\mathrm{dw_{ij}}} + \mathrm{x}\,(t)\right) \tag{5.3}$$

where wij is the weight of the connection i to j, η is the learning rate, C is the loss function, $x\ (t)$ is a stochastic term.

After a random initialization of the weights, training data are passed through the net and the output is computed. The error gradient is then used in a backward direction to change the weights.

Different loss functions can be used, such as mean square error (MSE) for classifiers and cross-entropy for models whose output is a probability value [Eqs. (5.4) and (5.5)].

$$\mathrm{MSE} = \frac{1}{2 * n}\sum_{x} |\,|y\,(x) - a|\,|^2 \tag{5.4}$$

where w is the collection of weights of the network, n is the total number of training inputs, and a is the vector of outputs from the network when x is input.

$$\mathrm{Cross-entropy} = -1n\sum_{x}(y \ln a + (1-y)\ln(1-a)) \tag{5.5}$$

where: a is the output of the neuron, y is the true label, and ln is the natural log.

For training different settings of the hyperparameters are usually tried in order to find the best net.

~~Epochs: o~~One epoch is when the entire data set is passed once forward and backward through the net. A too-small number of epochs makes the net to underfit the data, a too-big value makes it to overfit. Big data sets are divided into batches as follows:

- Batch size: it is the number of data present in a batch.
- Iterations: it is the number of batches needed to complete one epoch.
- Learning rate: determines the step size at every iteration. It is denoted by the symbol η, and has a value in (0, 1). η controls how much to change the model in response to the estimated error each time the model weights are updated as in Eq. (5.3). Smaller η require more training epochs.

Different regularization methods are used to avoid overfitting. For big networks and large data sets, the most used is dropout during training. It means that nodes of the networks, with a predefined probability, are randomly deleted (in practice set to zero) so to avoid that the net learns too much of the data and eventually becomes unable to generalize. Dropout has also a theoretical interpretation in terms of uncertainty estimation. Gal and Ghahramani (2016) presented some theoretical justification, showing that a NN with arbitrary depth and nonlinearity, with dropout applied before every weight layer, is mathematically equivalent to an approximation of the probabilistic deep Gaussian process. They showed that model uncertainty can be obtained from NN models using dropout at inference time and not just during training and that the dropout probability characterizes the posterior uncertainty. Of course, this is just an approximation of Bayesian reasoning. Optimization algorithms, as Adam (Kingma and Ba, 2017), help to minimize the loss function.

5.2.3 Deep learning and multilayer neural networks

Big networks with many inner layers can approximate more complex functions. However using feedforward fully connected neural nets with many layers is not efficient, for many reasons. First of all, determining the number of neurons in each layer is usually done using trial and errors, and the process is very long. The effectiveness of DNNs depends on taking advantage of the structure of the data to find the net architecture. Deep learning (DL) is a field in machine learning based on NN that work as a representation learning (Bengio et al., 2013). The most known of such networks are shortly illustrated.

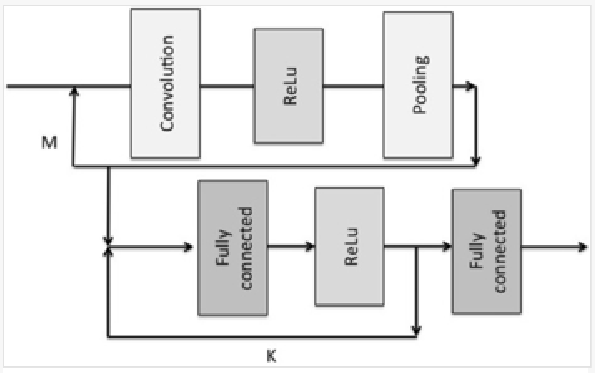
5.2.3.1 Convolutional neural network

A CNN is a network that breaks down an input, typically an image, into smaller pieces and extracts the feature to be used to make a classification decision (LeCun and Bengio, 1995).

CNNs combine convolutional layers, pooling layers, fully connected layers, and activation layers (Fig. 5.2). CNNs are effective in using the geometric information contained in the input.

- Convolutional layers use a set of fixed-sized weight matrices, called filters, which perform element-wise multiplication on the image pixels. Weights are learned. Convolutional layers use strides (positive numbers) to accelerate the dimension reduction; the default stride 1 moves one pixel at a time, and higher strides move more pixels.
- Pooling layers usually come after or in between convolutional layers to reduce the dimension of the original input. Average pooling smoothens the features taking the average, while max pooling picks the largest value to extract distinctive features.
- Fully connected layers flatten the previous layer and connect to all nodes of the previous layer to each of its nodes. Then the output is fed into a nonlinear layer.

Figure 5.2



The convolutional neural network. M and K are the number of times each block is performed.

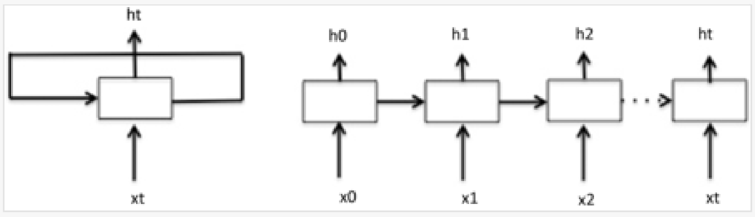
Q4 CNNs are better than humans in image classification (LeCun et al., 2015); a possible explanation is they make use of the two main bio-inspired principles: population coding and statistical coding. Population coding is the mechanism used in the brain to represent sensory information through a group of spatially organized neurons, in which neighboring neurons have similar activity. Statistical coding, a special kind of population coding, is observed in the areas of the brain devoted to the preprocessing of the sensory data, by reducing the dimensionality of the input space.

5.2.3.2 Recurrent neural network

Feedforward networks can receive a fixed-sized number of data to the hidden layers. The inability to handle variable length input, as in the case of texts, and the necessity to consider long-term dependencies give rise to using recurrent units with feed-forward NN.

RNNs (Williams et al., 1986), in Fig. 5.3, use the same function and the same set of parameters for every time step: at each time step, the previous hidden state and the current input are fed through the function to update the hidden state. The loss is defined as the sum of the loss from each time step. RNNs are effective in using the temporal interconnections present in the data.

Figure 5.3



Recurrent neural network as a cycle and unfolded at each time step.

Q5 Backpropagation through time is used for training. During backpropagation, gradients in RNN tend to vanish or explode as time length increases. Exploding gradient means a big increase in the norm of the gradient because of long-term components growing exponentially larger than the short-term ones. The vanishing gradient is the opposite. Different techniques are available (Bengio et al., 2013) to solve them. In particular, using ReLU as the activation function prevents the derivative from shrinking the gradients when x is larger than zero.

Long short-term memory (LSTM) is a kind of RNN developed by Hochreiter and Schmidhuber (1997). It has a more complicated recurrent unit, containing an input gate, a forget gate, and an output gate to control information that gets passed on and stored.

In bidirectional RNN two directions of the next time of the hidden states are used; one goes from old to new and the other goes from new to old. In this way, the learning process can both maintain the past and look to future information.

5.2.3.3 Graph convolutional neural networks

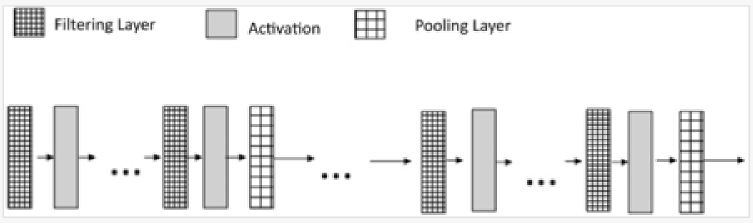
A graph is a couple $G=<V, E>$, where V is a finite set of nodes and E, the set of ~~arches~~edges, is a subset of V^2. A fully connected feedforward NN can represent graphs as sets of nodes. However, this representation does not consider the information of the edges, which connect a few of the nodes in a nonregular way.

Extracting features from graphs using CNNs is not easy, as the graph nodes do not have the regularity of pixels. However, the advantages of CNNs, that is, local connection, shared weights and multilayer refinement can be useful for graphs too. A GCN (Zhou et al., 2019) encodes the nodes of the graph into an embedding space that approximates similarity in the original network. Mathematically a GCN applies convolutions; while CNNs apply two-dimensional convolutions to the image pixels, GCNs apply convolutions to node features. It means a GCN passes a filter over the graph, looking for essential vertices and edges. Graph filtering refines the node feature, graph and pooling generates a smaller graph.

Images are represented as matrices, the products of pixels and weights. Graphs too are represented as matrices, the product of adjacency matrix and weights. A GCN in general considers an adjacency matrix concatenated with a matrix of node features, and instead of a window to select the neighboring pixels it uses an aggregation function. The matrix of the node features contains node information for features such as atom weights, etc. Neighborhood aggregation function aggregates the node features using mean, concatenate, or sum functions.

A sequence of those filter layers produces an effect similar to CNN (Kipf and Welling, 2017). The pooling layer converts a graph into subgraphs to represent higher-level representations. It is computationally expensive to use all hidden states from all the nodes for prediction, hence the need of downsampling. The pooling operation, as max, mean, or sum, can be combined with attention mechanisms. Readout generates graph-level representation by summing up all the hidden states in the graphs, and a fully connected layer is used to generate the output. Fig. 5.4 shows the similarity between GCN and CNN.

Figure 5.4



The structure of a graph convolutional neural network.

GCNs generalize convolutions in the graph domain by operating on groups of spatially close node neighbors. One of the challenges of this approach is to define an operator which works with different-sized neighborhoods and maintains the weight-sharing property of CNNs. Hamilton et al. (2017) introduced GraphSAGE, a method for computing node representations by sampling a fixed-size neighborhood of each node, and then by aggregating them, using for instance the mean over all the sampled neighbors' feature vectors.

5.2.4 Attention mechanism

The attention mechanism, observed in animals' visual cortex, means focusing on a specific part of the sensorial data to help interpret the scene. The same process is used while reading a sentence: first we focus on the subject and the verb, then we move to articles and adjectives. In the same way, a NN can define the important parts of the input it receives in order to make a correct prediction. Cho et al. (2015) first proposed an RNN with the addition of an attention mechanism in order to reduce the computation resources needed to achieve good accuracy. The idea is to use a layer connected to the RNN that receives the context vector and calculates the weight this has on the final prediction. Then softmax transforms it in a probability.

The attention mechanisms can deal with variable-sized inputs, focusing on the most relevant parts of the input to make decisions. The attention mechanism used to compute a representation of a single sequence, named self-attention, has been useful in sentence representation and translation.

The attention mechanisms can deal with variable-sized inputs, focusing on the most relevant parts of the input to make decisions. The attention mechanism used to compute a representation of a single sequence (named self-attention) has been useful in sentence representation and translation.

Q6 Velickovic et al. (2017) introduced the graph attention network (GAT), which is an attention-based architecture to perform node classification of graph-structured data. The idea is to compute the hidden representations of each node in the graph, by attending over its neighbors, following a self-attention strategy.

A single graph attentional layer is used in the GAT architecture. The input to the attentional layer is a set of node features; the output is a new set of node features (of potentially different cardinality). The attention mechanism is a single-layer feedforward NN. In order to obtain sufficient expressive power to transform the input features into higher-level features, a learnable linear transformation is required. To that end, Leaky ReLU, parameterized by a weight matrix, is applied to every node at the beginning. Then the e_{ij} coefficients that indicate the importance of node j's features to node ~~I~~i, are computed for a limited number of neighboring j nodes. To make coefficients comparable across the various nodes, the e_{ij} are normalized using the softmax function, so producing the new coefficients a_{ij}=softmax(e_{ij})

Softmax takes as input a vector **z** of *K* real numbers and normalizes it into a probability distribution consisting of *K* probabilities proportional to the exponentials of the input numbers. It turns the vector of *K* real values *zi* into a vector of *K* real values that sum up to 1, thus constituting a probability distribution [Eq. (5.6)].

$$s(z)i = \frac{e^{zi}}{\sum_{j=1}^{K} e^{zj}} \tag{5.6}$$

The normalized attention weights are used as the coefficients of the hidden states in the update function.

To stabilize this learning process, multihead attention is used. It means that the *eij* are calculated *K* (for $K \geq 1$) times and then the features are concatenated or averaged.

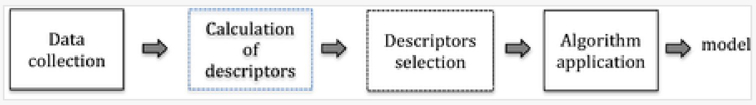
The benefits of GAT architecture are many; it is computationally efficient in storage and number of operations [as it is linear in (vertices+edges)], is not dependent on graph size, it is localized, it is inductive because it does not depend on global graph structure.

Many DNNs end in a penultimate layer that outputs real numbers. Softmax converts those numbers to a normalized probability distribution, which can be more easily interpreted. For this reason, it is usual to append a softmax function as the final layer of the NN, even in case the attention mechanism is not used.

5.3 Neural networks for quantitative structure–activity relationship: input, output, and parameters

In QSAR modeling NNs have been often used as algorithms to create models (Devillers, 1996; Gini and Katrizky, 1999). In classical QSAR, this means assigning to the input neurons the descriptors values. As chemical descriptors are produced in large numbers, usually a selection method is used to select the relevant ones to construct the model (Fig. 5.5)

Figure 5.5



The steps in creating a quantitative structure–activity relationship model using machine learning methods.

5.3.1 Input

NNs are used with numerical values as input. While chemical descriptors are numbers, whole molecules are structured data. Developing NNs for graph input has been for a long time a research target. Early examples of graph input to NN for chemical problems are in (Micheli et al., 2001). Those early methods did not have a large impact for two reasons: the network computation was heavy, and the input was ad hoc format.

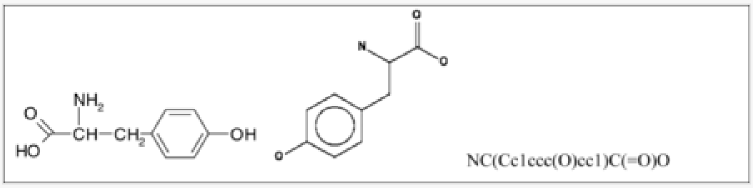
Today DNNs allow avoiding chemical descriptors as well as crafting ad hoc representations. Three easily available formats for molecules can be used: SMILES (Weininger et al., 1989), images, and chemical graphs. DNNs that learn directly useful features from data have been successfully developed for chemical applications, especially in drug discovery (Zhang et al., 2017).

The ambitious goal of DNN-based QSAR is designing a new architecture, which depends only on raw data and no other a priori or expert knowledge. The knowledge self-generated during training can be extracted in order to compare it with the existing one or to analyze it as new knowledge.

5.3.2 Chemical graphs and their representation

In mathematical chemistry a chemical graph, as in Fig. 5.6, represents the structural formula of a chemical compound. A chemical graph is an undirected labeled graph, whose vertices correspond to the atoms and edges correspond to the chemical bonds. Vertices are labeled with the kinds of corresponding atoms and edges are labeled with the types of bonds. Visiting the chemical graph in top-down order produces a string, called SMILES, that is equivalent to the chemical graph (Weininger et al., 1989). The reverse is not true, as more SMILES correctly represent the same chemical graph.

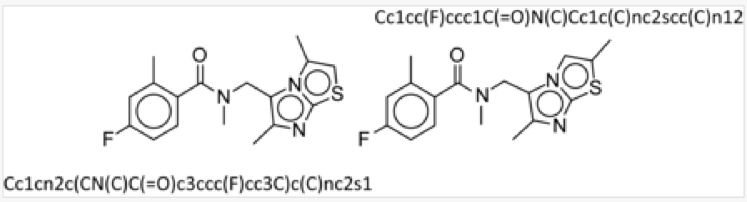
Figure 5.6



From *left* to *right*: a chemical structure, its H-depleted chemical graph, where unmarked vertices represent Carbon atoms, and its SMILES.

Another problem is that a small change in the structure can produce a large change in the SMILES, as in Fig. 5.7. In this case the similarity of the chemical graphs seems lost (Gini, 2020).

Figure 5.7


Cc1cc(F)ccc1C(=O)N(C)Cc1c(C)nc2scc(C)n12
Cc1cn2c(CN(C)C(=O)c3ccc(F)cc3C)c(C)nc2s1

Two similar molecules and their different SMILES.

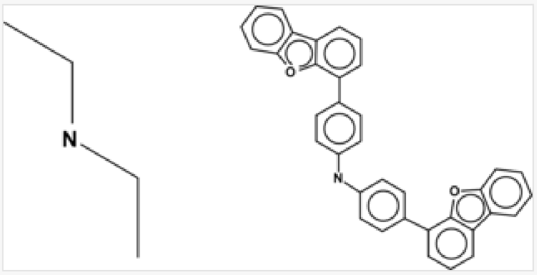
5.3.2.1 SMILES as input

NNs able to receive directly the information from public repositories are an important tool, and SMILES are commonly available in chemical data sets. RNN are apt to work on strings. In order to avoid ambiguity, a bidirectional cell in the RNN allows having both the previous and the future input, in order to easily distinguish the differences. Strings in the input are transformed into numbers (word embedding), usually by a NN, using a technique called Word2Vec. Examples of DNNs with SMILES as input are in Goh et al. (2018) and ~~and Gini and Zanoli (2020)~~.Gini et al. (2019).

5.3.2.2 *Images of two-dimensional structures as input*

From a data set composed of SMILES strings, it is easy to generate a data set of images using the public library RDKit. Two examples are in Fig. 5.8. It is obvious that a low resolution (for instance, 80×80 pixels) is enough for the first molecule, critical for the second one. As the image generated by RDKit uses black for the bonds, blue and red for atoms different from Carbon (not represented), the color images in RGB at low resolution require 80×80×3 pixels, as reported in (Gini and Zanoli, 2020) and Goh et al. 2017)~~. Using such images is~~DELETE TO THE END OF
Q7 SENTENCE reported in ~~Gini et al. (2019)~~Gini et al., 2019; Gini and Zanoli, 2020 and Goh et al. (2017).

Figure 5.8

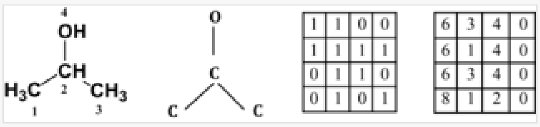


Two examples of drawings generated by RDKit.

5.3.2.3 *Chemical graphs as input*

Graphs and SMILES contain the same information. However, graphs are not affected by the lack of unicity of SMILES. This is not a problem when training, as it is easy to use the same algorithm to generate canonical SMILES, but it can be a problem when using the trained net. Moreover, using chemical graphs presents the advantage of reasoning directly on topological similarity. GCNs (Wu et al., 2019) can accept chemical graphs represented as (largely empty) adjacency matrices, where the rows and the columns are the node labels and the edge are represented by entry 1 if two nodes share an edge. Each node (atom) can have a set of features, as a number of hydrogen bonds, valence, aromaticity, and so on. The feature values are inserted in a feature matrix. In GCN the adjacency matrix of the molecule and the feature matrix (Fig. 5.9) are concatenated to create the input.

Figure 5.9



A chemical graph, adjacency matrix, and feature matrix (the columns indicate atom type, #H, valence, aromaticity).

GCN performs on the matrices the same operations that CNN performs on the images; instead of extracting edges and shapes, it extracts subgraphs.

5.3.3 Output

The output of the networks can be real numbers or class labels. Even in case~~,~~ a class label is desired, this label can be obtained from a regression net that predicts numbers in (0, 1). The advantage is that it is possible to set different thresholds to get the classes, and the output of the model can be interpreted as the probability for the compound to belong to a specific class.

Two loss functions during the training process are needed: MSE if the output is a class, and cross-entropy if the output is a real number.

5.3.4 Performance parameters

As for any model the subdivision of training and test sets, and the use of n-fold cross-validation are common methods to assess the prediction ability. The performance of the obtained classifier is measured on the test and training sets, or in cross-validation, using a bunch of the following parameters.~~:~~ Accuracy [Eq. (5.7)] ~~:~~alone is not a reliable metric because it will yeld misleading results if the data set is unbalanced.

Previous Version

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FN} + \text{FP}}$$

Updated Version

$$\begin{aligned}\text{accuracy} &= \frac{\text{TruePositives+TrueNegatives}}{\text{TruePositives+TrueNegatives+FalseNegatives+FalsePositives}} \\ &= \frac{\text{TP+TN}}{\text{TP+TN+FN+FP}}\end{aligned} \tag{5.7}$$

~~Accuracy alone is not a reliable metric because it will yield misleading results if the data set is unbalanced.~~ Other parameters, in Eq. (5.8), are sensitivity (which is the proportion of real positive cases that are correctly predicted), specificity, precision (or positive predictive value), and $F1$, which is the harmonic mean of precision and sensitivity.

$$\text{sensitivity} = \text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$F1 = \frac{2 * \text{TP}}{2 * \text{TP} + \text{FP} + \text{FN}} \tag{5.8}$$

The Matthews correlation coefficient (MCC), in Eq. (5.9), is a well-balanced measure; it is a correlation coefficient between the observed and predicted binary classifications, and it returns a value between −1 and +1. A coefficient of +1 represents a perfect prediction, 0 means no better than random prediction and −1 indicates total disagreement between prediction and observation.

$$\text{MCC} = \frac{\text{TP} * \text{TN} - \text{FP} * \text{FN}}{\sqrt{(\text{TN} + \text{FP})(\text{TN} + \text{FN})(\text{TP} + \text{FP})(\text{TP} + \text{FN})}} \tag{5.9}$$

5.4 Deep learning models for mutagenicity prediction

Mutagenicity refers to a chemical or physical agent's capacity to cause mutations, that is, genetic alterations. The most adopted in vitro test for mutagenicity is the Ames test (Ames, 1984) on different engineered strains of the bacteria

Salmonella typhimurium; it is largely used for the registration of chemicals, and to screen impurities in pharmaceuticals and metabolites in pesticides.

5.4.1 Structure–activity relationship and quantitative structure–activity relationship models for Ames test

Many SAs have been defined for mutagenicity both by experts and by data-mining tools.

Toxtree is the open implementation of the Benigni and Bossa (2008) rules for mutagenicity and carcinogenicity. Toxtree processes a query chemical and answers whether SAs for carcinogenicity are found or not. SAs for carcinogenicity are used, as the action mechanism of genotoxic carcinogenicity apply also to the mutagenic activity in bacteria. SAs can only point out chemicals potentially toxic, whereas no conclusions about nontoxic chemicals are possible. Thus the SAs are not a discriminant model, differently from QSAR models. The majority of SAs are direct-acting carcinogens, while others relate to genotoxic carcinogens that become toxic after metabolic transformations (Plošnik et al., 2016). The situation is more complicated, as other factors may influence the activity of a chemical, as molecular weight, reactivity, solubility, physical state, and the molecule's geometry. Moreover, the SAs-based models do not work equally efficiently in discriminating between active and inactive chemicals in the same chemical class; the reason is that the SA models lack subrules detailed enough to describe these modulations. Indeed o reason on difficult chemical classes, in a read-across approach, a few hundred SAs have been created both by expert judgments and statistical tools (Gini et al., 2014). SARpy is such a statistical tool that automatically extracts substructures from data sets (Gini et al., 2013). The substructures are accepted as positive SAs after statistical analysis, and substructures never found in toxic chemicals are named negative SAs.

More than 60 QSAR models, either free or commercial, are available to predict the output of the Ames test. Most of them are built using a data set of about 6000 molecules of chemicals of industrial interest, made freely available by Kazius et al. (2005) and Hansen et al. (2009).

For assessing some of those models, a comparison has been done using a new large set provided by Masamitsu Honma, from the Division of Genetics and Mutagenesis in the National Institutes of Health Sciences (NIHS) of Japan, who distributed to participating institutes about 12,000 newly tested substances, most of them negatives, for the first AMES/QSAR International challenge (Honma et al., 2019). This comparison (Benfenati et al., 2018) considered a few Ames prediction models, constructed on the Kazius and Hansen data sets, and used as a test set ~~for~~ the new NIHS data. Table 5.1 indicates some of the considered models, the algorithm used, and the number of predicted classes.

Table 5.1

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Seven mutagenicity models tested on an a large test set in Benfenati et al. (2018).

Model name and provider	Method	Predicted classes
CAESAR (IRFMN-in VEGA)	Chemical descriptors for SVM + SA search	Pos, Neg, Suspect Pos
SARpy (IRFMN-in VEGA)	SA search	Pos, Neg, Possible Neg
ISS (IRFMN-in VEGA)	SA search	Pos, Neg, Unknown
KNN (IRFMN-in VEGA)	k-NN	Pos, Neg
SAm (Nestlè)	SA search	Pos, Neg
AIm (Nestlè)	k-NN	Pos, Neg
AZAMES (Swetox)	Chemical descriptors - conformal prediction	Pos, Neg, Both, Unknown

The results of those seven models on a test set of 2427 molecules of the NIHS data, which were not part of the training set of any of the considered models, are in Table 5.2.

Table 5.2

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Results of seven of the tested models on a test set of 2427 molecules.

Model/performance	CAESAR	ISS	SARpy	KNN	SAm	Aim	AZAMES
Total predictions	2427	2427	2427	2427	2427	*2399*	*2415*
Accuracy	0.67	0.70	0.66	0.67	0.79	0.76	**0.85**
Sensitivity	**0.65**	0.59	0.57	0.51	0.56	0.40	0.37
Specificity	0.68	0.72	0.68	0.70	0.82	0.81	**0.92**
MCC	0.23	0.22	0.18	0.15	**0.31**	0.17	0.29

Note: In bold the best value for each parameter.

The coverage for two of the models is not full, as they do not take decisions for dubious molecules. Those performances can characterize the state of the art of predictive models for the Ames test. The MCC value is not higher than 0.31.

5.4.2 Deep learning models for Ames test

Three kinds of DL models for the Ames test are presented, each using as input SMILES, images of the two-dimensional structures, or chemical graphs.

The data sets used for all of them is a large data set, which includes public and proprietary data (NIHS data), plus data collected through PubMed, and whose statistical details are in Zanoli (2018). It is quite unbalanced, as reported in Table 5.3.

Table 5.3

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Characteristics of Ames-Zanoli dataset.

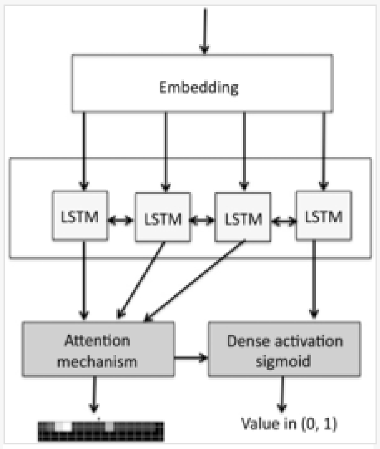
	Number	Percentage
Positive	8127	33.85%
Negative	15,876	66.14%
Total	24,003	100%

5.4.2.1 Learning from SMILES

The basic RNN structure cannot remember input values for a long gap of time/words; LSTM units make RNN capable Q8 of remembering values over arbitrary time. SmilesNet (Gini, 2018; Zanoli, 2018) is a bidirectional RNN-LSTM model that analyzes SMILES. It was trained with the data set of 24,003 molecules, randomly divided into training (80%) and test (20%) sets.

In SmilesNet (Fig. 5.10) the input SMILES go through word embedding, then into a bidirectional RNN that gives inputs to the attention mechanism and to the final dense layer for prediction. The attention mechanism highlights the parts of the string that are important for reaching the result.

Figure 5.10



SmilesNet architecture.

The dense layer uses the softmax function to produce a value in (0, 1). Setting a threshold at 0.5 (or another value) transforms the result into a binary classification.

The best net, optimized using Adam, has the following parameters: 100 epochs, 400 neurons, batch size ~~of~~ 32, learning rate 0.001, dropout probability 0.2. On the test set of 20% chemicals, the performances of SmilesNet are in Table 5.4, which shows also the results of the two models presented next..

Table 5.4

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Performances of the three deep learning models on the testing set.

Model	MCC	Specificity	Sensitivity
Toxception	0.53	0.62	0.87
SmilesNet	0.63	0.74	0.82
C-Tox	0.70	0.76	0.83

5.4.2.2 Learning from images

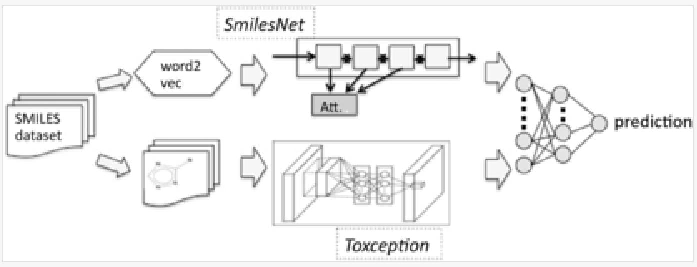
Q9 Toxception (Gini ~~et al.,~~and Zanoli, 2020) is a CNN using the images of the molecules generated from the canonical Q10 SMILES written by VEGA (Benfenati et al., 2013). The net uses the inception architecture, which contains blocks of filters of different dimensions, including 1×1, and the concept of the residual network (He et al., 2016).

The data set has been divided into 80% for training and 20% for testing. To reduce computation time the size of the input image is reduced from the original 299×299×3 pixels to 80×80×3. The best network, optimized using Adam, has the following parameters: neurons in the first layer 16, 200 epochs, batch size 32, learning rate 0.001, dropout probability 0.2. The results of Toxception on the test set of 20% randomly selected molecules are in Table 5.4.

5.4.2.3 *Integrating features from SMILES and images*

SmilesNet and Toxception behave in the same way: they extract the features from data and use them in a final layer to produce the classification. They can be used as standalone models or be combined. Gini et al. (2019) found that the results improve just by taking the features from the trained nets and using them to learn the final classification through a simple two-layer NN, as illustrated in Fig. 5.11. This final model, called C-Tox, positively compares to both Toxception and SmilesNet, as reported in Table 5.4. The value of the MCC for those three models is quite good.

Figure 5.11



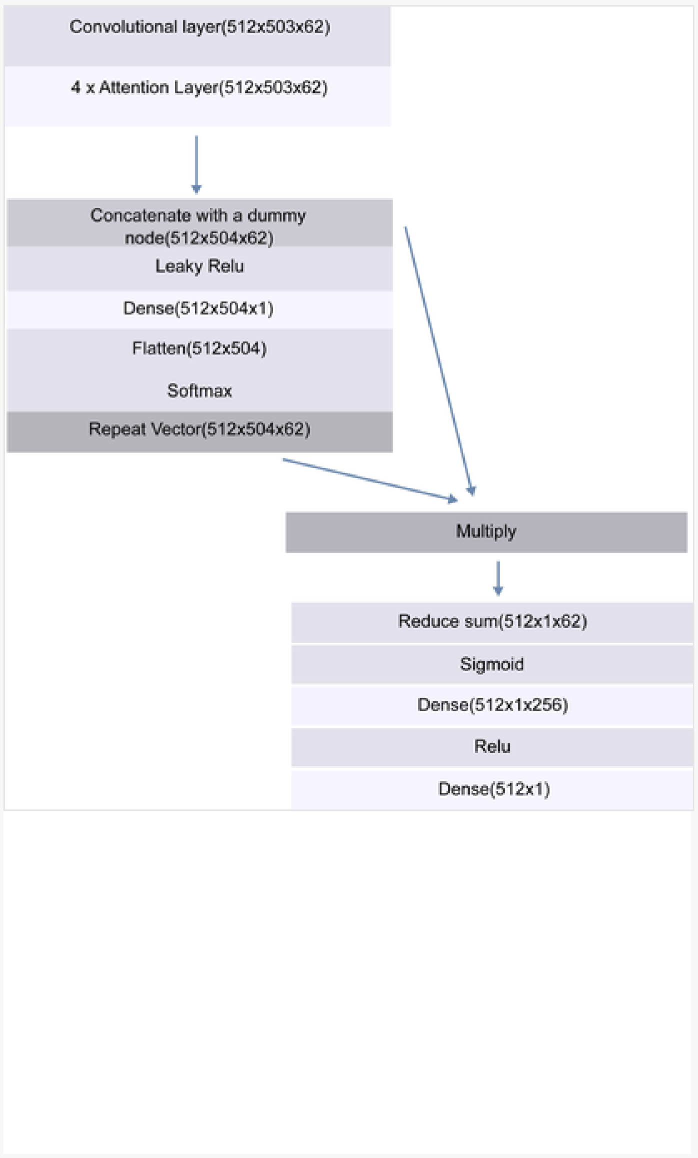
C-Tox combines the features extracted by Toxception and SmilesNet (removing the final fully connected layer) and produces a classification. The extracted features are visualized in Toxception and SmilesNet.

The features extracted by the networks produce also an explanation of the results, as discussed in the following section.

5.4.2.4 *Learning from chemical graphs*

The last developed DL model (Hung, 2020) uses directly chemical graphs in the GCN architecture as illustrated in Fig. 5.12. It has convolutional layers, attention layers, readout layers to sum up the hidden states, and a fully connected NN to produce the prediction.

Figure 5.12



Convolutional layer(512x503x62)
4 x Attention Layer(512x503x62)
Concatenate with a dummy node(512x504x62)
Leaky Relu
Dense(512x504x1)
Flatten(512x504)
Softmax
Repeat Vector(512x504x62)
Multiply
Reduce sum(512x1x62)
Sigmoid
Dense(512x1x256)
Relu
Dense(512x1)

G-Ames architecture. It has a convolutional layer (batch_size×503×62), four attention layers, and dense fully connected layers to give the output. A concatenation with a dummy node is used to extract from the attention layer the relevant subgraphs.

In the used data set the longest SMILES contains 503 characters, so the largest graph has less than 503 nodes, and this number characterizes the convolutional layer.

The training set of G-Ames is composed of 80% of the data set of 24,003 SMILES; the test set is 20%, which corresponds to 4800 SMILES. Data are randomized with seed 0 using the library NumPy. The values of the G-Ames hyperparameters have been experimentally determined using Adam. They are number of layers 2, batch size 512, and learning rate 0.01. In the attention layer, which computes the hidden state of each node using weighted neighborhood aggregation, the value of the attention head is 2. The pooling strategy at readout is the sum, which performs the weighted sum of the hidden state of each node, given the attention score function. The attention score function uses both the hidden state of the nodes and the target state, using a dummy node to link all the nodes, and feeds the fully connected layer. The statistics of G-Ames are in Table 5.5.

Table 5.5

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Performance of G-Ames on the test set.

TP	FP	FN	TN	Acc	MCC	Specificity	Precision	Recall	F1
1172	500	416	2712	0.81	0.58	0.84	0.70	0.74	0.72

To better understand the coverage of the chemical space, an analysis of the prediction accuracy of the chemical classes present in the data set has been done. The classes are determined using ClassyFire, a general-purpose free application that considers a hierarchy of chemical classes (Djoumbou Fenang et al., 2016). The chemical characterization of the test set is in Table 5.6.

Table 5.6

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Previous version

Expand

Percentages of chemical classes in the test set.

Organoheterocyclic	**0.2319**

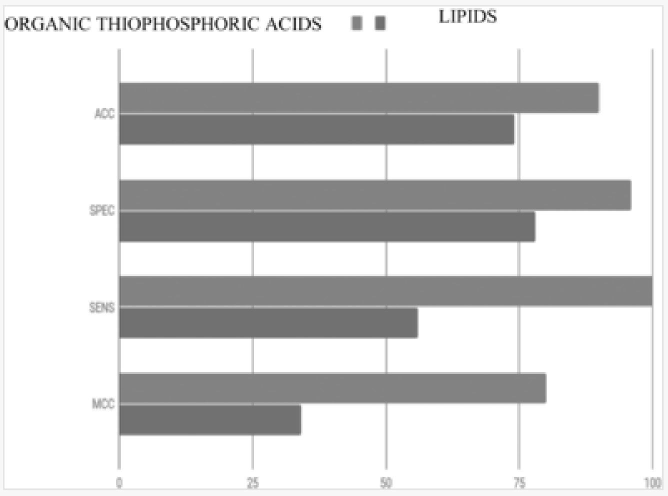
Updated version

Percentages of chemical classes in the test set.

~~Organoheterocyclic~~Class	**~~0.2319~~Percentage in the test set**
Organoheterocyclic	23.19
Benzene and substituted derivatives	~~0.1727~~17.27
Other benzenoids	~~0.~~10,6~~0~~
Others	~~0.0979~~9.79
Carboxylic acids and derivatives	~~0.0~~8.1~~5~~
Ethers	~~0.0~~4.9~~0~~
Lipids and lipid-like molecules	~~0.0~~3~~93~~.85
Organonitrogen compounds	~~0.0~~3.85~~5~~
Benzoic acids	~~0.0~~3.~~27~~27
Other organic oxygen compounds	~~0.0~~3.2~~0~~
Phenylpropanoids and polyketides	~~0.0~~3.2~~0~~
Other organic acids and derivatives	~~0.0243~~2.43
Organohalogen compounds	1.98~~0.0198~~
Biphenyls and derivatives	~~0.0173~~1.73
Phenols	~~0.0106~~1.06
Alcohols and polyols	0.~~0079~~79
Hydrocarbons	0.~~0058~~58
Organic thiophosphoric acids	~~0.0008~~0.08

In G-Ames the best-predicted class is organic thiophosphoric acids, the worst predicted class is lipids and lipid-like ~~molecules~~. The performances for those two chemical classes are in Fig. 5.13.

Figure 5.13



The performances of the best and worst predicted chemical classes in G-Ames.

5.5 Interpreting deep neural network models

The acceptance of a predictive model depends on its statistical properties and on its interpretability. In modern QSAR the interpretation is often done a posteriori, by extracting the functional elements that contribute to the observed toxicity value (Polishchuk, 2017).

Simple structural elements, as coded in chemical descriptors, are routinely used to interpret QSAR models. Parts of the SMILES can be used instead. Using simple symbols to code parts of the SMILES, Toropov et al. (2012) observed how they are correlated to increase or decrease of mutagenicity of the molecules. In particular, the presence of the attribute "(", which means the presence of any branching in an aromatic system, and the attribute "2", which means presence of any two rings, can be interpreted as a quite probable increase of mutagenicity. Also, the presence of nitrogen is an indicator of a decrease in mutagenicity.

Mixing the statistical tools of QSAR and the expert knowledge of SAR can indeed make DNN models interpretable. The method requires extracting from the trained net the substructures of the molecule that mainly contribute to its toxicity, then comparing those substructures with available knowledge.

In the following, those steps are illustrated for the three kinds of DNN before being presented. In fact, any CNN, RNN, or GCN extracts the features that are then used to classify the input. Those features are in nonlinear relationships with the property. The combination of the features, more than the presence of a specific one, characterizes the output result. Therefore the knowledge extracted could describe the complex phenomenon of toxicity but it is not apt to become a set of SAs as in SAR methods.

Anyhow, comparing the obtained features with other known SAs can help understanding the role of substructures in toxicity. In particular, the SAs considered in the following are the ones obtained by Toxtree (Benigni and Bossa, 2008) and SARpy (Gini et al., 2013). SAs of Toxtree are explaining both in vivo and in vitro carcinogenicity and mutagenicity data. In databases including chemicals from diverse chemical classes, the Toxtree models agree around 65% with rodent carcinogenicity data, and 75% with Salmonella mutagenicity data. SAs of SARpy are derived from Salmonella mutagenicity data only.

5.5.1 Extracting substructures

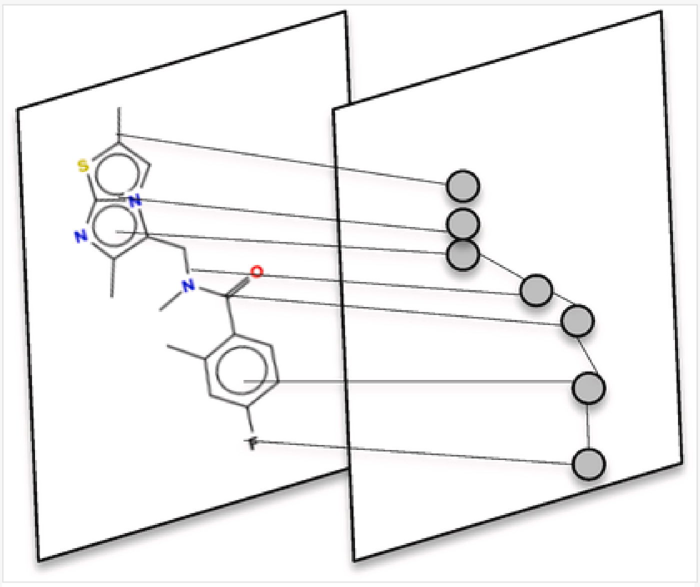
Using images as input means that initially~~,~~ the neurons represent pixels, ~~that~~and after convolutions, pooling and ReLU the neurons represent features distorted in a nonlinear way and in lower number (LeCun et al., 2015). Going deeper into the network, the neurons get information from larger parts of the image and from other neurons so learning more complicated features like groups of atoms.

The weight vectors passed to the output neurons contain a value in (0, 1), which gives the predicted output. The final layer in Toxception contains a correlation matrix, whose values can be visualized by highlighting which parts of the molecule have higher values (Gini et al., 2019). Q11

From the attention layer of SmilesNet, it is possible to get the correlation values of each SMILES character in the classification layer. As the inputs are SMILES, it is possible to apply directly to them the correlation matrix of the attention mechanism, so obtaining the weight that each character has on the final prediction. Substrings, containing at least three characters, have been considered. Each substring appears in the data set many times, often both in toxic and nontoxic molecules; this finding is expected as the presence of SAs alone can give ambiguous results (Gini et al., 2014).

For G-Ames the extraction of subgraphs is even easier. The effect of using convolutions and pooling on the graph is that of transforming groups of near nodes to one node, as in Fig. 5.14.

Figure 5.14



Graph reduction in the convolution layers of graph convolutional neural network.

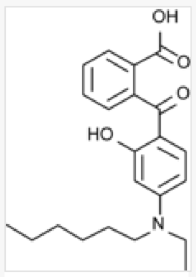
The extraction of subgraphs in GCN has been automatically done by the following procedure:

- For each SMILES, a weight matrix is extracted from the attention part of the readout layer. The attention part consists of an FC layer, which performs weighted sum, and a softmax layer that transforms the weights to make them sum to 1.
- For each SMILES the fragments with maximum weights are extracted. The maximum length of the fragment can be assigned by the user.
- For each fragment, the atoms that have a threshold greater than 0.5 are extracted.

G-Ames outputs the image of the test molecule with the identified substructure in red (Fig. 5.15).

Figure 5.15

Figure Replacement Requested



Visualization (dark color here, in red in the program interface) of a SA identified by G-Ames in the SMILES: O=C(O)c2ccccc2(C(=O)c1ccc(cc1(O))N(CC)CCCCCC).

Replacement Image: fig 15.jpg

Replacement Instruction: I have added in the figure an explaining word

5.5.2 Comparison of substrings with SARpy SAs

SARpy mines data sets to extract SAs candidates (in number of thousands) and selects the ones (in number of hundreds) that are statistically stronger. SARpy prefers long substructures, so to cope with the activity landscape, while SmilesNet returns all substrings found with a high weight. G-Ames returns all subgraphs up to a maximum number of nodes.

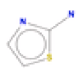
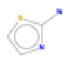
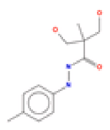
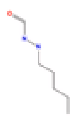
The extracted substructures cannot be automatically taken as SAs. However, they can be compared against the set of known mutagenicity SAs. In case of coincidence, the confidence in the model can improve. In the reverse case, the absence of coincidence does not imply that the model is wrong; however, other considerations will be necessary to explain the result.

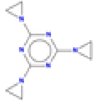
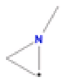
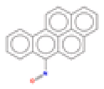
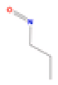
Table 5.7 illustrates four molecules. Each row considers a positive molecule (predicted with toxicity probability >0.5), a substring found by SmilesNet, and the most similar substructure alert (using Tanimoto similarity) found by SARpy. The four molecules are predicted as mutagenic by Toxtree which recognizes them, respectively, as primary aromatic amine, hydrazine, epoxide, and arizidine, aromatic nitroso group.

Table 5.7

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Examples of substrings extracted by SmilesNet compared with SARpy SAs.

MOLECULE	Structure	p tox	Substructure in SmilesNet	SARpy SA
Nc1sccn1		0.88	Nc1nccs1 	Nc1nccs1
O=C(NNc1ccc(cc1)C)C(C)(CO)CO		0.92	cccccNNC=O 	O=CNNc1ccccc1
C1CN1c2nc(nc(n2)N3CC3)N4CC4		0.88		C1CN1

			cN1*C1 	
O=Nc1c2ccccc2c3ccc4cccc5ccc1c3c45		0.86	cccN=O 	O=Nc1ccccc1

SmilesNet and SARpy can extract toxic fragments and fragments associated with the absence of toxicity; the same substring may appear in molecules with different toxicity probability, as the prediction depends on the whole structure, not only on the presence of the SA.

Table 5.8 illustrates some examples of the coherence of the alerts found by SmilesNet and SARpy in correctly predicting the mutagenicity property.

Table 5.8

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Coherence between the substring found by SmilesNet and the SARpy equivalent fragment for correctly predicted positive and negative molecules.

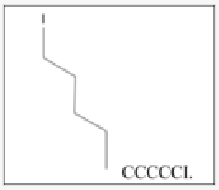
Molecule in SMILES	Predicted SmilesNet probability	substring=SARpy fragment	SARpy_prediction
NC(N)=S	0.163	NC=S	Negative
O=S(=O)(OCC(F)(F)F)C	0.243	CC(F)F	Negative
NC1=NNC=N1	0.327	c1nc[nH]n1	Negative
O=C(OCC)C(=CNC1CC1)C(=O)c2cc(F)c(F)cc2(F)	0.347	NC1CC1	Negative
OC(=O)CS	0.406	O=CCS	Negative
N#CCCSCc1nc(N=C(N)N)sc1	0.470	CC#N	Negative
Nc1sccn1	0.884	Nc1nccs1	Positive
CCCI	0.980	CCI	Positive

Note: More cases are in Gini (2020).

5.5.3 Comparison of substructures with Toxtree

Another way to look at the results of the DL models is to see whether they find some expert-defined SAs, as the ones available in Toxtree. Twenty-five Toxtree SAs can explain the 1290 substrings extracted from SmilesNet. An example is in Fig. 5.16, while Table 5.9 shows those SAs associated with the number of similar substrings, and their probability of being mutagen, with three different thresholds. Although SAs code a rule of toxicity, they have been found mostly in experimentally nonmutagenic molecules (Honma et al., 2019). Indeed, this low accuracy is reflected in the numerical distribution of the substring's probability of toxicity.

Figure 5.16



An example of substring extracted by SmilesNet. The fragment CCCCCI is associated with a probability 0.98 of being mutagen, and 0.02 of being not mutagenic. In Toxtree it is considered as SA8: Aliphatic halogens.

Table 5.9

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Toxtree SAs matched with SmilesNet substrings.

Toxtree SAs	Matched SmilesNet fragments	Prob<0.5	Prob>=0.5	Prob>=0.8
SA1 Acyl halides	11	2	9	1
SA10 alfa, beta unsaturated carbonyls	77	37	40	4
SA11 Simple aldehyde	151	54	97	35
SA13 Hydrazine	150	84	66	16
SA14 Aliphatic azo and azoxy	17	15	2	0
SA15 Isocyanate and isothiocyanate groups	15	8	7	3
SA16 Alkyl carbamate and thiocarbamate	11	4	7	2
SA2 Alkyl(C<5) or benzyl ester of sulfonic or phosphonic acid	30	21	9	0
SA21 Alkyl and aryl N-nitroso groups	48	30	18	0
SA22 Azide and triazene groups	31	25	6	0
SA23 Aliphatic N-nitro	5	5	0	0
SA25 Aromatic nitroso group	10	4	6	4
SA27 Nitro aromatic	3	3	0	0
SA28 Primary aromatic amine, hydroxyl amine, and its derived esters (with restrictions)	167	74	93	26
SA28bis Aromatic mono- and dialkylamine	16	6	10	6
SA28ter Aromatic N-acyl amine	37	0	37	15
SA29 Aromatic diazo	1	0	1	0
SA3 N-methylol derivatives	6	2	4	0
SA4 Monohaloalkene	20	11	9	0
SA5S or N mustard	2	0	2	0
SA61 Alkyl hydroperoxides	8	4	4	0
SA64 Hydroxamic acid derivatives	9	5	4	0
SA7 Epoxides and aziridines	64	33	31	7
SA8 Aliphatic halogens	354	195	159	44
SA9 Alkyl nitrite	8	7	1	0

Note: The columns “Prob” show the number of matched substrings that have that probability of being mutagens.

Using G-Ames, an analysis has been done considering the chemical classes, where the extracted subgroups are compared with Toxtree SAs. TP (positive according to Toxtree as they contain at least a SA), FP, TN, FN are counted, and the statistics of the local models are reported in Table 5.10.

Table 5.10

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

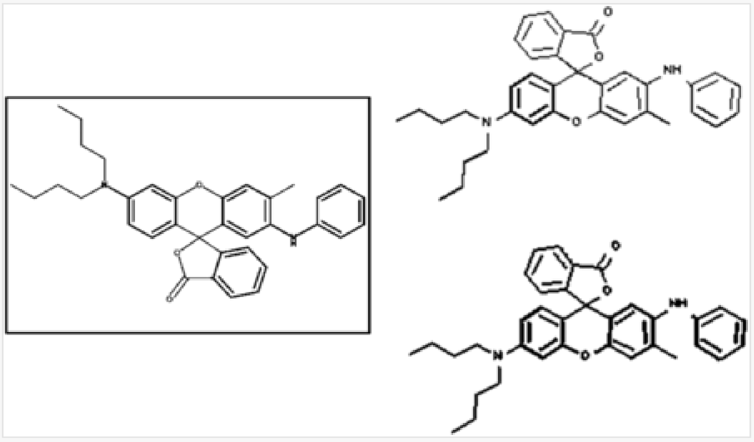
The statistics of Toxtree SAs grouped by the chemical classes of the molecules.

Statistics\classes	Alcohols and polyols	Benzene and substituted derivatives	Benzoic acids and derivatives	Carboxylic acids and derivatives	Ethers
Accuracy	0.92	0.69	0.66	0.86	0.79
MCC	0.54	0.27	0.17	0.52	0.37
Specificity	0.97	0.75	0.75	0.93	0.92
Precision	0.67	0.45	0.39	0.65	0.61
Recall	0.5	0.43	0.54	0.56	0.41
F1	0.67	0.39	0.45	0.65	0.61

Statistics\classes	Lipids and lipid-like molecules	Organohalogen	Organoheterocyclic	Organonitrogen	Phenols	Phenylpropanoids and polyketides
Accuracy	0.77	0.78	0.63	0.82	0.68	0.71
MCC	0.18	0.57	0.16	0.58	0.38	0.20
Specificity	0.94	0.90	0.68	0.88	0.64	0.83
Precision	0.47	0.84	0.37	0.73	0.5	0.42
Recall	0.19	0.64	0.49	0.69	0.77	0.37
F1	0.47	0.84	0.37	0.73	0.5	0.42

Fig. 5.17 shows the SA (QSA18_Ames.Polycyclic Aromatic Hydrocarbons) identified in Toxtree in the SMILES O=C5OC2(c4ccc(cc4(Oc1cc(c(cc12)Nc3ccccc3)C))N(CCCC)CCCC)c6ccccc56. G-Ames identifies nearly the same substructure for fragment-size of 20. For fragment-size of 10, the substructure extracted is smaller but the same SA can be identified.

Figure 5.17



On the *left* the image generated by Toxtree for the SMILES: O=C5OC2(c4ccc(cc4(Oc1cc(c(cc12)Nc3ccccc3)C))N(CCCC)CCCC)c6ccccc56. On the *right*, in dark, the two substructures identified by G-Ames with fragment-size of 10 and 20, respectively.

Table 5.11 shows some examples of the substructures extracted by G-Ames, their corresponding Toxtree SAs, and the predicted mutagenicity class.

Table 5.11

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Substructure extracted by the model validated by Toxtree.

SMILES extracted by G-Ames	Matched SA in Toxtree	Model predicted value
Cc1ccc([N+](=O)[O-])cc1	SA27_Ames	Mutagenic
S/C(Cl)=C(\Cl)CCl	SA8_Ames	Mutagenic
ccccOCCO	SA24_Ames	Mutagenic
c1ccc2ncccc2c1	SA69_Ames	Mutagenic
O=CCO	SA11_Ames	Mutagenic
C\C=N/N	SA13_Ames	Mutagenic
Nc1ccccc1	SA28_Ames	Mutagenic
O=C1OCC(C(Br)Br)=C1Cl	SA65_Ames	Mutagenic
CC=CC(=O)Cl	SA1_Ames	Mutagenic
COC(=O)[C@H](C)CN=[N+]=[N-]	SA22_Ames	Mutagenic
O=C1OCC(C(Br)Br)=C1Cl	SA8_Ames, SA65_Ames	Mutagenic
CC=CC(=O)Cl	SA1_Ames, SA10_Ames	Mutagenic
Oc1ccccc1	None	Nonmutagenic
FC(F)(F)c1ccccc1	None	Nonmutagenic
NCc1ccccn1	None	Nonmutagenic
FC(F)c1ccccc1	None	Nonmutagenic
C[N+](C)(C)CCCN	None	Nonmutagenic
CCOCCC(=O)C(F)F	None	Nonmutagenic
CN=Nc1ccccc1	None	Nonmutagenic

N#Cc1ccc([nH])c(n)c1	None	Nonmutagenic
FCOC(F)(F)C(F)(F)Br	None	Nonmutagenic
NCC(F)(F)F	None	Nonmutagenic

The substructures that are predicted as mutagenic correspond to known SAs in Toxtree. The number of occurrences in the positive substructures for each Toxtree SA is in Table 5.12. 22 SAs of Toxtree are not found by G-Ames, as indicated in Table 5.12. It is possible that the test set does not include SMILES with these alerts. Another possible reason is that many SMILES experimentally nonmutagenic are identified as mutagenic in Toxtree, so they are not counted in the occurrences.

Table 5.12

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Toxtree SAs identified by G-Ames in the Ames test set.

Toxtree SAs	Occurrences identified by G-Ames	Toxtree SAs	Occurrences identified by G-Ames
SA10_Ames	26	*SA37_Ames*	*0*
SA11_Ames	29	SA38_Ames	8
SA12_Ames	*0*	*SA39_Ames*	*0*
SA13_Ames	10	*SA3_Ames*	*0*
SA14_Ames	6	SA4_Ames	6
SA15_Ames	*0*	*SA57_Ames*	*0*
SA16_Ames	2	*SA58_Ames*	*0*
SA18_Ames	*0*	*SA59_Ames*	*0*
SA19_Ames	*0*	SA5_Ames	2
SA1_Ames	12	*SA60_Ames*	*0*
SA21_Ames	34	SA61_Ames	1
SA22_Ames	11	*SA62_Ames*	*0*
SA23_Ames	5	*SA63_Ames*	*0*
SA24_Ames	21	SA64_Ames	2
SA25_Ames	11	SA65_Ames	4
SA26_Ames	*0*	*SA66_Ames*	*0*
SA27_Ames	115	*SA67_Ames*	*0*
SA28_Ames	77	*SA68_Ames*	*0*
SA28bis_Ames	11	SA69_Ames	4
SA28ter_Ames	*0*	*SA6_Ames*	*0*
SA29_Ames	*0*	SA7_Ames	34
SA2_Ames	8	SA8_Ames	91
SA30_Ames	*0*	SA9_Ames	1

Note: In *italics* SAs not found.

Table 5.13 considers the statistics on the test set using only the Toxtree SAS. Note that there are many SMILES in the Ames test set that have experimental values different from the ones predicted by Toxtree.

Table 5.13

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Statistics on the test set of applying only the SAs identified by Toxtree.

Acc	MCC	Specificity	Precision	Recall	F1
0.68	0.23	0.74	0.38	0.51	0.38

Note: The values are worse than the results of G-Ames.

Finally, a comparison of the substrings found by SmilesNet and the substructures found by G-Ames is illustrated in Table 5.14 for some molecules of Tables 5.7 and 5.8. For the considered molecules the predictions agree but the substructures can be quite different. The reasons are many. Only the most important substructure is considered, but a few are indeed extracted for each molecule. Moreover, they may change according to the parameters used in the extraction, and on the attention weights that are computed in different ways.

Table 5.14

The table layout displayed in this section is not how it will appear in the final version. The representation below is solely purposed for providing corrections to the table. To preview the actual presentation of the table, please view the Proof.

Comparison of substructures extracted from SmilesNet (SN) and G-Ames (G) for molecules predicted in the same class by both the models.

SMILES	SN pred	G pred	SN substring	G substructure	comment
O=Nc1c2ccccc2c3ccc4cccc5ccc1c3c4	1	1	cccN=O	cccccccccc	
O=C(Nc1c(nc(cc1SC)C)SC)CBr	1	1	=C(Nc1c(nc(cc1	CcccSC	
Nc1ccc2nc3c(N)cccc3nc2c1	1	1	Nc1ccc2nc3	Nc1ccccc1	
O=Nc1c2ccccc2c3ccc4cccc5ccc1c3c45	1	1	O=Nc1c2c	cccccccccc	
Nc2ccc(c1ccc(N)cc1C)c(c2)C	1	1	Nc2cc	cc(c1ccc(N)cc1C)	
IC(I)I	1	1	C(I)I	IC(I)I	SmilesNet <G-Ames
NC(N)=S	0	0	NC=S	NC(N)=S	SmilesNet <G-Ames
O=S(=O)(OCC(F)(F)F)C	0	0	CC(F)F	CS(=O)(=O)OCC(F)(F)F	
NC1=NNC=N1	0	0	CC(F)F	Nc1nc[nH]n1	
O=C(OCC)C(=CNC1CC1)C(=O)c2cc(F)c(F)cc2(F)	0	0	NC1CC1	C=CN	
OC(=O)CS	0	0	O=CCS	C	No meaningful substructure in G-Ames
N#CCCSCc1nc(N=C(N)N)sc1	0	0	CC#N	CCCS	

A comparison of the substructures obtained from the different models shows that some molecules are differently classified and/or are classified giving more importance to different subparts. This is expected, as the result is obtained from different nonlinear transformations of the input. Big SAs can better account for the property, in case of SAs-based models, but short substructures and their combination are of interest in many QSARs that use fingerprints. The fragments extracted from DNNs are somehow similar to fingerprints more than to full SAs. The previously reported paper (Toropov et al., 2012) illustrates a similar situation, where small structural features can help in explaining the behavior.

5.6 Discussion and conclusions

This section recalls the main points presented and unifies them in the prospective of understanding the role of DL in the QSAR domain.

The mutagenicity models presented in the previous sections show some novelties in input, output, and methods with respect to the state of the art of QSAR modeling. In particular:

1. The input takes directly the chemical two-dimensional structure. SMILES (~~as~~ used in SmilesNet) and their equivalent chemical graphs (~~as~~ used in Toxception) are commonly used and easily understandable. Available chemical libraries automatically compute adjacency and feature matrices, used by G-Ames. This last format allows ~~for~~ avoiding ~~the possible~~any problem with SMILES input, that is, the nonunique SMILES representation for the same molecule, and the possible lack of SMILES similarity for similar molecules. Moreover, ~~they~~SMILES are not affected by low image resolution as in the case of using images.

2. There is no need of computing chemical descriptors; the feature matrix can contain simple information and eventually be empty.

3. Neither fingerprints nor functional subgroups are a priori selected.

4. Results are possibly expressed with uncertainty estimation.

5. Results are analyzed in “universal” chemical classes, that is, in classes that are not constructed around given functional subgroups relevant to the property under study.

6. Functional subgroups found as important are automatically extracted for helping in the model interpretation. Their concordance with available knowledge can improve confidence in the results.

7. The models work as QSARs, so they produce an output for every chemical.

The DL models before presented are trained without adding any external knowledge besides the SMILES and the experimental Ames test results. This choice has advantages and drawbacks.

Advantages are obvious, since the steps of feature computation and reduction are not needed, and no fingerprints with a priori choice of substructures are ~~generated~~computed. The knowledge extracted from the network is auto-generated and not biased by a priori choices.

Drawbacks are also obvious since DL models should be “opened” to extract the features and so to explain somehow what the network has learned. Another possible drawback is that the training of the network requires a long computation time and is better on hardware-enhanced computers. The main equations of a NN are matrix/vector operations, which are apt for execution on massively parallel hardware architectures, such as GPUs, to speed up the training process. Anyhow, GPUs and the needed open-source software are easily available. Another drawback is that DL models are good if they can use large training sets. Finally, NN architecture design is still an art, not a science; playing with the DL libraries requires some training.

The characteristics of C-Tox, SmilesNet, and Toxception models, presented in Section 5.4, match the points 2, 3, and 6 listed above. The data interpretation of those models uses the attention layer, takes the segments with high weights, then transformed into SMILES for human understanding. The G-Ames model has all the mentioned characteristics.

Another DL model for mutagenicity has been developed at MultiCase company (Chakravarti and Radha Mani Alla, 2019) using the NIHS mutagenicity data set. It also divides the external test set into chemical classes, using 53 chemical classes (not indicating how the classes have been defined), and calculates the prediction sensitivity and specificity within the classes. It is not possible to make a full comparison with the results of Section 5.4, as the classes are differently defined. It also uses the attention layer to extract substructures, but no statistical analysis of the found alerts is presented.

All the DL models have statistics not worse and generally better than the traditional models.

5.6.1 A future for deep learning models

QSAR is based on the hypothesis that the chemical structure is responsible for the activity; it follows that similar molecules are expected to have similar properties. This simple assertion is contradicted in many cases, where similar molecules behave in a different way. The similarity is such a subtle property that it cannot be defined in a universal way and independently on the use (Gini, 2020). DL makes the similarity hypothesis integrated into the process of feature extraction. It is also interesting to observe that even using SMILES as input the statistical results are very similar to the ones obtained by graphs. This is a further confirmation that the feature extraction of DL makes the similarity hypothesis less crucial.

The hidden neurons of DNNs may represent previously known SAs, as illustrated for SARpy and Toxtree, which usually are humanly engineered by experts. We may expect that hidden neurons contain also novel groups, yet undiscovered.

Cichy and Kaiser (2019) say that the general idea that DNNs results cannot be explained should be corrected:

> "The answer to a question such as "why does a DNN unit behave such and such" is not "because it represents this or that feature of the world," but "because the unit needs to respond such in order to fulfill its function in enabling a particular objective, such as object recognition." That is, the nature of the explanation is teleological."

They finally underline that DNNs have an important aspect that lacks in many other models: they can be used for exploration. The exploratory power of DNNs is a strong point in favor of using them in cases when a theory is missing, as in most cases of toxicity.

Q12 The concluding sentence is taken from Buckner and Garson (2019):

> "Over the centuries, philosophers have struggled to understand how our concepts are defined. It is now widely acknowledged that trying to characterize ordinary notions with necessary and sufficient conditions is doomed to failure. Connectionist models seem especially well suited to accommodating graded notions of category membership of this kind. Nets can learn to appreciate subtle statistical patterns that would be very hard to express as hard and fast rules."

References

The corrections made in this section will be reviewed and approved by master copier.

Ames, B.N., 1984. The detection of environmental mutagens and potential. Cancer 53, 2030–2040.

Basak, S.C., 2013. Philosophy of mathematical chemistry: a personal perspective. HYLE–Int. J. Philos. Chem. 19 (1), 3–17.

Benfenati, E., Manganaro, A., Gini, G., 2013. VEGA-QSAR: AI inside a platform for predictive toxicology, Wokshop Popularize Artif. Intell. (PAI) 2013, Torino Dec. 5, 2013, pp. 21–28, http://ceur-ws.org/Vol-1107/.

Benfenati, E., Belli, M., Borges, T., Casimiro, E., Cester, J., Fernandez, A., et al., 2016. Results of a round-robin exercise on read-across. SAR. QSAR Env. Res. 27 (5), 371–384.

Benfenati, E., Golbamaki, A., Raitano, G., Roncaglioni, A., Manganelli, S., Lemke, F., et al., 2018. A large comparison of integrated SAR/QSAR models of the Ames test for mutagenicity. SAR. QSAR Env. Res. 29 (8), 591–611.

Benfenati, E., Chaudhry, Q., Gini, G., Dorne, J. L., G., 2019. Integrating in silico models and read-across methods for predicting toxicity of chemicals: a step-wise strategy. Environ. Int. 131, 105060.

Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: a review and new perspectives. IEEE Trans. PAMI 35 (8), 1798–1828.

Benigni, R., Bossa, C., 2008. Structure alerts for carcinogenicity, and the Salmonella assay system: a novel insight through the chemical relational databases technology. Mutat. Res. 659 (3), 248–261.

Buckner, C., Garson, J., 2019. Connectionism. The Stanford Encyclopedia of Philosophy. <https://plato.stanford.edu/archives/fall2019/entries/connectionism/>.

Chakravarti, S.K., Radha Mani Alla, S., 2019. Descriptor free QSAR modeling using deep learning with long short-term memory neural networks. Front. Artif. Intell. 2 (17).

Chen, T., Chen, H., 1995. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans. Neural Netw. 6 (4), 911–917.

Cho, K., Courville, A.C., Bengio, Y., 2015. Describing multimedia content using attention-based encoder-decoder networks. IEEE Trans. Multimed. 17 (11), 1875–1886.

Cichy, R.M., Kaiser, D., 2019. Deep neural networks as scientific models. Trends Cog Sci. 23 (4), 305–317.

Devillers, J. (Ed.), 1996. Neural Networks in QSAR and Drug Design. Academic Press, San Diego, CA.

Djoumbou Fenang, Y., Esner, R., Knox, C., Chepeley, L., Hastings, J., Owen, G., et al., 2016. ClassyFire: automated chemical classification with a comprehensive, computable taxonomy. J. Chemoinf 8, 61.

Gal, Y., Ghahramani, Z., 2016. Dropout as a bayesian approximation: representing model uncertainty in deep learning. In: Proceedings of The 33rd International Conference on Machine Learning, PMLR 48:1050-1059.

Gini, G., Katrizky, A. (Eds.), 1999. Predictive toxicology of chemicals: experiences and impact of AI tools. In: Papers from the AAAI Spring Symposium on Predictive toxicology SS-99-01. AAAI Press, Menlo Park, CA.

Gini, G., Ferrari, T., Cattaneo, D., Bakhtyari, N.G., Manganaro, A., Benfenati, E., 2013. Automatic knowledge extraction from chemical structures: the case of mutagenicity prediction. SAR. QSAR Env. Res. 24 (5), 365–383.

Gini, G., Franchi, A.M., Manganaro, A., Golbamaki, A., Benfenati, E., 2014. ToxRead: a tool to assist in read across and its use to assess mutagenicity of chemicals. SAR. QSAR Env. Res, 25 12, 999–1011.

Gini, G., 2016. QSAR methods. In: Benfenati, E. (Ed.), In Silico Methods for Predicting Drug Toxicity. Springer, Clifton, N.J., pp. 1–20.

Gini, G., 2018. QSAR: what else? In: Nicolotti, O. (Ed.), Computational Toxicology. Methods in Molecular Biology. Humana Press, New York, NY, pp. 79–105.

Gini, G., 2020. The QSAR similarity principle in the deep learning era: confirmation or revision? Found. Chem. 22, 383–402.

Gini, G., Zanoli, F., Gamba, A., Raitano, G., Benfenati, E., 2019. Could deep learning in neural networks improve the QSAR models? SAR. QSAR Env. Res. 30 (9), 617–642.

Gini, G., Zanoli, F., 2020. Machine learning and deep learning methods in ecotoxicological QSAR modeling. In: Roy, K. (Ed.), Ecotoxicological QSARs. Springer Nature, Berlin-Heidelberg.

Goh, G., Hodas, N., Siegel, C., Vishnu, A., 2018. SMILES2vec: an interpretable general-purpose deep neural network for predicting chemical properties, arXiv:1712.02034v2 [stat.ML].

Goh, G., Siegel, C., Vishnu, A., Hodas, N.O., Baker, N., 2017. Chemception: a deep neural network with minimal chemistry knowledge matches the performance of expert developed QSAR/QSPR models. Arvix.org/abs/1706.06689.

Hamilton, W.L., Ying, R., Leskovec, J., 2017. Inductive representation learning on large graphs. In: Proceedings Neural Information Processing Systems (NIPS).

Hansen, K., Mika, S., Schroeter, T., Sutter, A., ter Laak, A., Steger-Hartmann, T., et al., 2009. Benchmark data set for in silico prediction of Ames mutagenicity. J. Chem. Inf. Model. 49 (9), 2077–2081.

He, K., Zhang. X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.

Honma, M., Kitazawa, A., Cayley, A., Williams, R.V., Barber, C., Hanser, T., et al., 2019. Improvement of quantitative structure-activity relationship (QSAR) tools for predicting Ames mutagenicity: outcomes of the Ames/QSAR International Challenge Project. Mutagenesis 34 (1), 3–16.

Hung, C. 2020. Bayesian Graph Neural Network with uncertainty estimation to predict mutagenicity of chemicals. Master Thesis in Computer Science and Engineering. Politecnico di Milano, Italy.

Johnson, A.M., Maggiora, G.M., 1990. Concepts and Applications of Molecular Similarity. John Willey & Sons, New York.

Kazius, J., McGuire, R., Bursi, R., 2005. Derivation and validation of toxicophores for mutagenicity prediction. J. Med. Chem. 48, 312–320.

Kingma, D.P., Ba, J., 2017. Adam: a method for stochastic optimization, arXiv:1412.6980[cs.LG].

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: Proceedings International Conference on Learning Representations (ICLR 2017).

LeCun, Y., Bengio, Y., 1995. Convolutional networks for images, speech, and time series. In: Arbib, M.A. (Ed.), The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10.

~~Kirkpatrik~~Kirkpatrick, P., Ellis, C., 2004. Chemical space. Nature 32 (16), 823.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Micheli, A., Sperduti, A., Starita, A., 2001. Analysis of the internal representations developed by neural networks for structures applied to quantitative structure-activity relationship studies of benzodiazepines. J. Chem. Inf. Comput. Sci. 41, 202–218.

Plošnik, A., Vračko, M., Sollner Dolenc., M., 2016. Mutagenic and carcinogenic structural alerts and their mechanisms of action. Arh. Hig. Rada Toksikol. 2016 (67), 169–182.

Polishchuk, P.G., 2017. Interpretation of QSAR models: past, present and future. J. Chem. Inf. Model..

RDKit: Open-Source Cheminformatics Software. <https://www.rdkit.org>.

Todeschini, R., Consonni, V., 2009. Molecular Descriptors for Chemoinformatics. Wiley-VCH.

Toropov, A.A., Toropova, A.P., Benfenati, E., Gini, G., Leszczynska, D., Leszczynski, J., 2012. Calculation of molecular features with apparent impact on both activity of mutagens and activity of anticancer agents. Anti-Cancer Agents Med. Chem. 12.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., 2017. Graph attention networks. In: Proceedings ICLR.

Weininger, M., Weininger, A., Weininger, J.L., 1989. SMILES. 2. Algorithm for generation of unique SMILES notation. J. Chem. Inf. Model. 29, 97–101.

Werbos, P.J., 1994. The Roots of Backpropagation: From Ordered Derivatives to Neural Networks And Political Forecasting. Wiley, New York.

Williams, R.J., Hinton, G.E., Rumelhart, D.E., 1986. Learning representations by back-propagating errors. Nature. 323, 533–536.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S., 2019. A comprehensive survey on graph neural networks. Preprint arXiv:1901.00596v3 [cs.LG].

Zanoli, F., 2018. T-Tox: a new deep learning model to predict mutagenicity of chemicals. Master thesis in Computer Science and Engineering. Politecnico di Milano, Italy.

Zhang, L., Tan, J., Han, D., Zhu, H., 2017. From machine learning to deep learning: progress in machine intelligence for rational drug discovery. Drug. Discovery Today 22 (1), 1680–1685.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., et al., 2019. Graph neural networks: a review of methods and applications. arXiv:1812.08434v4 [cs.LG].

Q13
Further reading

The corrections made in this section will be reviewed and approved by master copier.

~~Toxtree <http://toxtree.sourceforge.net/>.~~

Abstract

The recent advancements in machine learning and deep learning (DL) methods are making it possible to create systems that automatically mine patterns and learn from data. Applications of those methods in chemistry, in particular quantitative structure–activity relationships (QSAR) and drug discovery, are already available. While DL can be applied as a conventional way of learning from chemical descriptors, the potentialities of the method are far more. In particular, the capabilities of DL to autonomously extract, through multiple transformations, the structural elements that are correlated with the property under investigation can help in discovering the link between a chemical and its biological/physical effects. After presenting the principal DL methods developed for chemical problems, the focus is on a study case in mutagenicity prediction that uses directly the chemical graph, either as SMILES, graphs, or images, and applies convolutional and recurrent networks. The knowledge extracted from the networks is analyzed and compared with the accepted structural alerts for mutagenicity. The next challenges and the future of DL for QSAR are finally discussed.

Keywords: Deep learning; deep neural networks; feature extraction; structural alerts; model interpretation; mutagenicity

Queries and Answers

Q1

Query: Please check the chapter title and amend if necessary.

Answer: it is ok

Q2

Query: Please verify and confirm the authorship details (author name and surname, affiliation, spelling of author's name and author's order) and amend if necessary.

Answer: done

Q3

Query: The spelling of the author name(s) in the text has been changed to "Kirkpatrik and Ellis, 2004" to match the reference list. Please check the spelling, and correct if necessary.

Answer: I have changed both to Kirkpatrick

Q4

Query: Please note that the reference citation "LeCun and Bengio, 2015" here has been changed to "LeCun et al., 2015". Kindly check and correct if necessary."

Answer: it is ok

Q5

Query: Please note the acronym BPTT is removed from here as it is not used elsewhere in the text.

Answer: it is ok

Q6

Query: "Velickovic et al., (2018)" has been changed to "Velickovic et al. (2017)" as per the reference list. Please confirm.

Answer: it is ok

Q7

Query: Please note that the reference citation "Gini, Zanoli (2019)" here has been changed to "Gini et al. (2019)". Kindly check and correct if necessary."

Answer: in 5.3.2.2 the references have been exchanged. I cannot complete the deletion and I have indicated what to delete (the systen does not allow to me to delete).

Q8

Query: Please note Gini and Zanoli 2018 is changed to Gini, 2018 and Zanoli, 2018 as per the reference in the list. Please check the edits for correctness.

Answer: it is ok

Q9

Query: The reference given here are cited in the text but is missing from the reference list – please make the list complete or remove the reference from the text: “Gini et al., 2020.

Answer: I have corrected the reference using another reference from the list

Q10

Query: "Benfenati et al., 2014" has been changed to "Benfenati et al., 2013" as per the reference list. Please confirm.

Answer: it is ok

Q11

Query: Please note that the reference citation "Gini and Zanoli, 2019" here has been changed to "Gini et al., 2019". Kindly check and correct if necessary."

Answer: It should be corrected using instead

(Gini and Zanoli, 2020)

Q12

Query: Please note that the reference citation "Buckner et al. (2019)" here has been changed to "Buckner and Garson (2019)". Kindly check and correct if necessary."

Answer: it is ok

Q13

Query: Further reading section contains references that are not cited in manuscript text. Please cite the references or confirm if we could delete it? Further reading will be retained if the references are not cited or deleted.

Answer: I have deleted this link because it may change after a time