JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

pubs.acs.org/jcim

Deep Architectures and Deep Learning in Chemoinformatics: The Prediction of Aqueous Solubility for Drug-Like Molecules

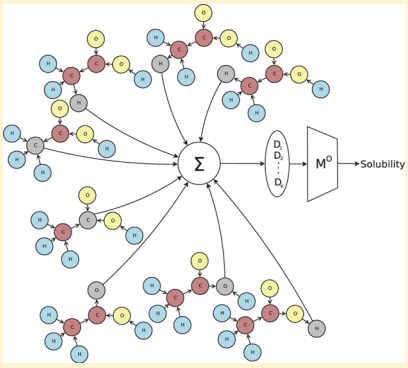
Alessandro Lusci,*[,†] Gianluca Pollastri,[†] and Pierre Baldi*[,‡]

[†]School of Computer Science and Informatics, University College Dublin, Belfield, Dublin 4, Ireland

[‡]Department of Computer Science, University of California, Irvine, Irvine, California 92697, United States

Supporting Information

ABSTRACT: Shallow machine learning methods have been applied to chemoinformatics problems with some success. As more data becomes available and more complex problems are tackled, deep machine learning methods may also become useful. Here, we present a brief overview of deep learning methods and show in particular how recursive neural network approaches can be applied to the problem of predicting molecular properties. However, molecules are typically described by undirected cyclic graphs, while recursive approaches typically use directed acyclic graphs. Thus, we develop methods to address this discrepancy, essentially by considering an ensemble of recursive neural networks associated with all possible vertex-centered acyclic orientations of the molecular graph. One advantage of this approach is that it relies only minimally on the identification of suitable molecular descriptors because suitable representations are learned automatically from the data. Several variants of this approach are applied to the problem of predicting aqueous solubility and tested on four benchmark data sets. Experimental results show that the performance of the deep learning methods matches or exceeds the performance of other state-of-the-art methods according to several evaluation metrics and expose the fundamental limitations arising from training sets that are too small or too noisy. A Web-based predictor, AquaSol, is available online through the ChemDB portal (cdb.ics.uci.edu) together with additional material.



INTRODUCTION

Shallow machine learning methods, such as shallow neural networks or kernel methods,[1] have been applied to chemoinformatics problems with some success, for instance, for the prediction of the physical, chemical, or biological properties of molecules[2−5] or the outcome of chemical reactions.[6,7] As more data becomes available and more complex problems are tackled, more complex models and deep machine learning methods may also become useful. Here, we provide a brief introduction to deep learning methods, demonstrate a recursive approach for deriving deep architectures for the prediction of molecular properties, and illustrate the approach on the problem of predicting aqueous solubility.

Aqueous Solubility Prediction. Aqueous solubility prediction is important in drug discovery and other applications. Given that over 80% of human blood consists of water, absorption of molecules with poor water solubility is low. Therefore, early identification of molecules with poor water solubility properties in a drug discovery pipeline can reduce the risk of failure.[8] Over the last few decades, several methods have been developed for the in silico prediction of aqueous solubility. Most of these methods are QSAR (quantitative structure−activity relationship) methods[9] with the general form

$$\text{Activity} = F(\text{structure}) = M(E(\text{structure})) \quad (1)$$

The function $F()$ is typically factorized into two subfunctions: the *encoding function* E and the *mapping function* M. The enconding function E transforms input molecules, which are naturally described by undirected graphs representing their chemical structure, into feature vectors of fixed length (e.g., fingerprints). This step is necessary in order to obtain a representation that is suitable for standard regression/classification tools such as neural networks (NN) or support vector machines (SVM) that can be used to learn the mapping function M from training examples.

These approaches depend crucially on the choice of molecular features. The first example of a computational method applied to the prediction of aqueous solubility dates back to 1924 when Fühner noticed that adding methyl groups to a series of homologous compounds tends to decrease solubility.[10] Adding methyl groups increases molecular size, and thus, molecular size became a key feature in the prediction of aqueous solubility.[11] Over the years, several other molecular features were found to correlate with aqueous solubility, including polar surface area,[12] octanol−water partition coefficient,[13−15] melting point,[16] hydrogen bond count,[17] and various molecular connectivity indexes.[18−20] For instance, the octanol−water partition coefficient log P_{octanol} is the logarithm of the ratio of the concentrations of a molecule in the two phases of a mixture of octanol and water at equilibrium.[21,22] It is often taken as a measure of the ability of a molecule to

Received: March 28, 2013
Published: June 24, 2013

   dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

traverse a lipdid membrane. The GSE[23] method uses a linear equation to combine the log $P_{octanol}$ and the melting point (T_m) to predict aqueous solubility. Even if such a method were to give satisfactory results, it displaces the problem of predicting aqueous solubility to the problem of measuring or predicting both log $P_{octanol}$ and T_m, which is not entirely satisfactory. Thus, other methods try to predict aqueous solubility using also topological and structural descriptors derived from the molecular graph.[24] An example is the prediction method combining log P, first order valency connectivity indices ($^1\chi^V$), delta chi ($\Delta^2\chi$), and information content (^{2}IC) by Louis et al.[25]

Although nowadays many other descriptors have been incorporated into the prediction tools, no model seems to be able to predict solubility with perfect accuracy.[11] This can be ascribed in part to experimental variability because it has been shown[26,27] that experimental solubility data can contain errors of up to 1.5 log units. Moreover it has been suggested[28] that the average error in experimental solubility data is no lower than 0.6 log units. Another reason behind the current limitations of prediction methods is the size of the available training sets which are very small compared to chemical space and contain a variety of biases. Finally, one cannot be certain that the current molecular descriptors capture all the relevant properties required for solubility prediction.[11]

Deep Learning. Learning is essential for building intelligent systems, whether carbon-based or silicon-based. Furthermore, in both cases, difficult tasks are not solved in a single step but rather require multiple processing stages. Hence the idea of deep learning, i.e., using processing systems that have multiple learnable stages, such as deep multilayer neural networks, for tackling difficult problems. In recent years, deep learning systems have improved the state-of-the-art in almost every field they have been applied to, from computer vision to speech recognition to natural language understanding to bioinformatics.[29−36] Thus, it is natural to try to apply deep learning methods to the prediction of molecular properties.

There are several nonexclusive ways of generating deep architectures for complex tasks, such as autoencoder-based architectures,[29,30,37−40] convolutional architectures,[41,42] and recursive architectures.[43−45] When the data consists of points with the same format and size, such as vector of fixed length or images of fixed size, then one can use a deep stack of neural networks to process the data. In addition, one can use stacks of autoencoders, which can be trained in an unsupervised way,[29] to automatically extract features and initialize the weights of the architecture, while taking advantage of usually plentiful unlabeled data. One can also use weight-sharing within each processing stage to derive convolutional architectures with the right invariance properties. These convolutional architectures have been extensively used in computer vision, for instance, in character recognition. Autoencoder-based and convolutional architectures can be applied to the prediction of molecular properties provided molecules are represented by vectors of fixed length, such as molecular fingerprints. While potentially useful for chemoinformatics, these approaches still rely heavily on a good encoding function and will not be further discussed here.

Because molecules are naturally represented by small graphs of *variable* size, it is also useful to develop methods for deriving more flexible deep architectures that can be applied directly to molecules and more generally to structured data of variable size, such as sequences, trees, graphs, and 3D structures. This can be achieved using the recursive approach described in.[44] However, the standard recursive approach relies on data represented by directed acyclic graphs (DAGs), whereas molecules are usually represented by undirected graphs (UGs). Thus, we first briefly review the general recursive approach for building deep learning architectures from DAGs and then show how the approach can be adapted to molecular UGs.

RECURSIVE DEEP LEARNING ARCHITECTURES

Directed Acyclic Graph Recursive Neural Networks (DAG-RNN). The starting point in this approach is a directed acyclic graph (DAG) associated with the data. Very often the DAG corresponds to a probabilistic graphical model (Bayesian network) of the data,[46] although it does not have to be so. The directed edges typically correspond to causal or temporal relationships between the variables. For instance, in the case of sequence data, such as text data or biological sequence data, the graphs are often based on linear chains associated with Markov models (Figure 1), such as hidden Markov models (HMMS),

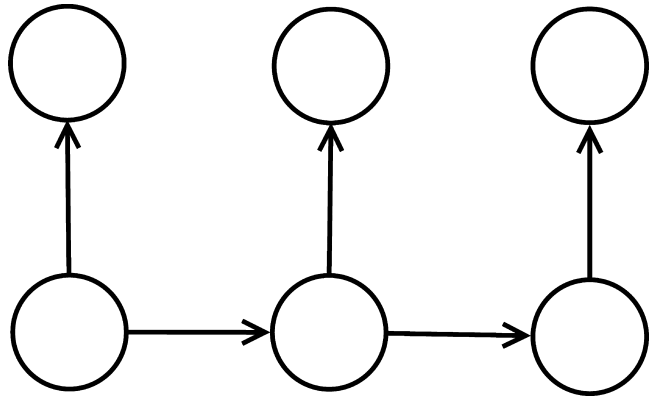


Figure 1. Directed acyclic graph (DAG). This is the graphical model (Bayesian Network) representation of a first-order hidden Markov model for sequence data. The HMM is defined by a finite set of hidden states, an alphabet of symbols, and two stochastic matrices: one for the state transitions and one for the symbol emissions from a given state. Horizontal edges correspond to transitions between hidden (nonvisible states). Vertical edges correspond to emissions of symbols.

input−output hidden Markov models, and other variants.[47] With two-dimensional data such as images, protein contact maps, or board games (e.g., GO), the graphs are typically based on two-dimensional lattices with acyclic orientations. Other kinds of structured data may be associated with oriented trees. In all these cases, the DAG-RNN approach associates vector variables with the nodes of the DAG and places a neural network (or any other kind of parametrized function) on the edges of the DAG to parametrize the relationship between the corresponding vector variables. While a different network can be placed on each edge, when the DAG has a regular structure, it is natural to share the weights of the neural networks associated with similar edges. For instance, in the example of Figures 1 and 2, the DAG associated with a hidden Markov model of the data can be converted to a deep neural network by using two basic neural network building blocks. One neural network for the transitions from state to state and one neural network for the emission of symbols from each state. These building blocks are shared or repeated at each position in time. When the architecture is unfolded in time or space, it yields a deep neural network with many layers and shared weights

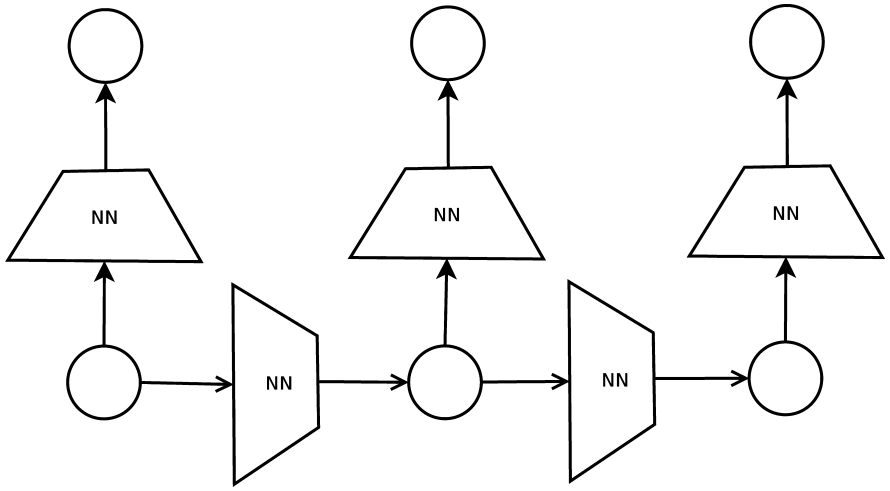


Figure 2. Directed acyclic graphs recursive neural network (DAG-RNN). A feed-forward neural network (or any other parametrized deterministic function) is assigned to each edge of the DAG. The neural network implements a deterministic function between corresponding input vectors and output vectors. The architecture and parameters of the neural networks can be shared across similar edges. In this case, this yields a model associated with two neural networks only: one to compute the next state vector given the current state vector and one to compute the emitted symbol given the current state vector.

which can be trained by gradient descent (backpropagation) and other algorithms.

Undirected Graph Recursive Neural Networks (UG-RNN). The DAG-RNN approach has been applied successfully to several problems, ranging for instance from protein secondary structure[43] to protein contact map[44] prediction to the game of GO.[45] However, it raises the obvious question of how it can be extended to domains where the graphs associated with the data are undirected graphs (UG) and possibly cyclic, which is obviously the case for small-molecule data. One possible approach is to try to convert the UG into a DAG in some canonical fashion. For small molecules, one could use an approach similar to what is used to produce canonical SMILES strings to derive a canonical numbering of the vertices and thus a canonical orientation of the edges. However, the resulting canonical orientation is likely to be quite arbitrary among all possible orientations, and hence unsatisfactory. Here instead, we take an approach that finesses this problem essentially by taking all possible acyclic orientations into consideration and using them as an ensemble. This is possible here because small molecules have a relatively small number of nodes and edges and thus considering all possible acyclic orientations is computationally feasible. The process is schematically illustrated in Figures 3 and 4. Starting from an undirected graph, we cycle through all the vertices. When a given vertex is selected as the root, a DAG is generated by orienting all the edges toward the root along the shortest possible paths. Each DAG has the same vertices as the original UG and essentially the same edges except that the edges are single oriented edges. In some cases, some of the original undirected edges can be oriented in either direction while leaving the overall derived graph acyclic. One possibility is to include all such possible orientations. Another possibility is to choose one orientation at random. Here, to keep things more manageable, we simply delete the corresponding ambiguous edges from the corresponding DAG. In short, if a molecular graph has N vertices associated with N atoms, this procedure yields N DAGs. We can then apply the DAG-RNN approach to each of the resulting DAGs and combine the outputs of all the DAG-RNN models to obtain an ensemble and derive a final prediction.

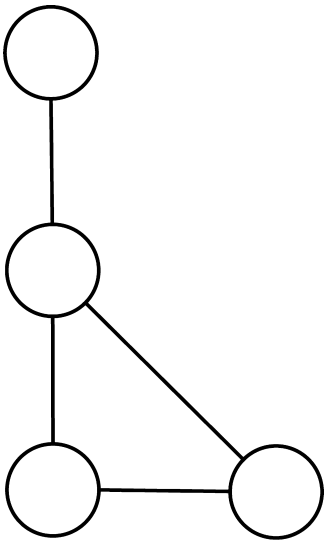


Figure 3. Undirected graph.

More precisely, consider a molecular graphs with N nodes $v_1,\ldots,v_N$, and N associated DAGs derived by the process above. A "contextual" vector $G_{v,k}$ is associated with each node v in each DAG indexed by k. This vector is a function of the local properties of the node v and of the contextual vectors associated with its parent nodes in the form

$$G_{v,k} = M^G(i_v, G_{pa^1_{[v,k]}}, \ldots, G_{pa^n_{[v,k]}}) \quad (2)$$

where $i_v \in \mathbb{R}^l$ is the input vector associated with the properties of vertex v (e.g., information about the corresponding atom) and $pa^1_{[v,k]},\ldots, pa^n_{[v,k]}$ are the parents of vertex v in the k^{th} DAG. Notice how this representation is recursive—each context vector is computed as a function of other context vectors. The function M^G which implements this recursion by "crawling" the molecular graph is implemented by a parametrized neural network, although other classes of parametrized functions can also be used. The neural network is taken to be the same for all molecules, all DAGs, and all vertices. While this is not strictly

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

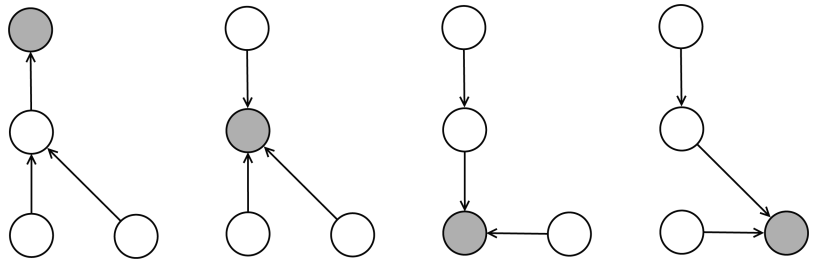


Figure 4. Directed acyclic graphs.

necessary, for instance, different NNs could be used for different classes of molecules, this strong weight sharing approach allows us to keep the number of free parameters in the model small. Notice how in order for this representation to be possible, there must be an upper bound n to the number of parents a node can have (in the application presented in this work, typically $n = 4$). If a node has m parent nodes with $m < n$, blank vectors (all zeroes) are passed to the function M^G as its last $n-m$ arguments. Likewise, for a source node with no parents, only the input vector is nonzero: all the other components of the contextual vector are set to 0.

In this approach, one is free to choose the nature of the local information vector i_v associated with each node v. This information could go well beyond the atom type and include, for instance, additional information about the local connectivity or properties of the molecule (e.g., aromaticity, topological indices, information about local paths or trees). To demonstrate the power of the UG-RNN approach and the underlying hypothesis that such information is extracted automatically by the crawling process, here we use only the atom type of the node and the bond type (single, double, triple) associated with the edges connecting the node to its parents $pa^1_{[v,k]},\ldots, pa^n_{[v,k]}$. This information is encoded as a binary vector with one-hot encoding (e.g., with three atom types only, the atom type is encoded as C = (1,0,0), N = (0,1,0), and O = (0,0,1) and similarly for the bond types).

As there is a path from each atom in a DAG to the root of the DAG, the recursion above ends up producing a final contextual vector in the root node of each DAG that receives, directly or indirectly, some contribution from all the other contextual vectors in the DAG. Intuitively, this vector can be viewed as the final product or summary of the crawling process in the corresponding DAG—it is a "view of the molecule" as seen from the corresponding root node. The N different views can be combined in different ways. Here, we simply add the corresponding vectors. Thus, the overall description of the molecule is obtained as the sum of descriptions of the molecule "as seen" from each of its nodes/atoms. More formally, $G_{\text{structure}}$ is defined as

$$G_{\text{structure}} = \sum_{k=1}^{N} G_{r_k,k} = (D_1, \ldots, D_K) \tag{3}$$

where here r_k denotes the root of the k-th DAG. Thus, $G_{\text{structure}}$ can be viewed as a feature vector with K learned features. The final prediction is produced by the output function M^O

$$p = M^O(G_{\text{structure}}) \tag{4}$$

where p is a class probability in classification problems or a continuous value in regression problem. Just like the encoding function M^G, the output function M^O can be implemented by a feed-forward neural network. An example of an alternative approach for combining the different views would be to apply a predictor to the contextual vector of each root node, and then take the average prediction of the ensemble. In either case, the resulting overall model is a deep feed-forward neural network, and therefore, it can be trained by gradient descent.[48−51] The feed-forward nature is a direct consequence of the use of the DAGs in the encoding step. Given a set of training examples, the parameters of the M^O and M^G networks can be trained by gradient descent to minimize the error (e.g., squared error in the case of regression, relative entropy in the case of classification) between predicted and true values. Thus the features used to encode molecules are learned by the system in a fully automated and task-specific manner. That is, if training is successful, the $G_{\text{structure}}$ vector provides an encoding of the molecular graph that is optimal in terms of minimizing the prediction error.

Example: UG-RNN Model of Acetic Acid. In this section, we illustrate the UG-RNN approach in detail using the acetic acid molecule (Figures 5 and 6) as a concrete example. For illustration purposes, we include hydrogen atoms, but in practice, these are implicit and can be omitted, which has the advantage of leading to more compact architectures and faster training. In this case, the graph for acetic acid has eight nodes. The first step consists in generating the corresponding eight DAGs, each one with edges directed toward a different root node (Figure 7, root atoms highlighted). The second step

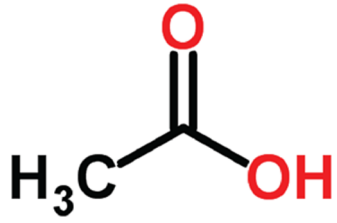


Figure 5. Acetic acid.

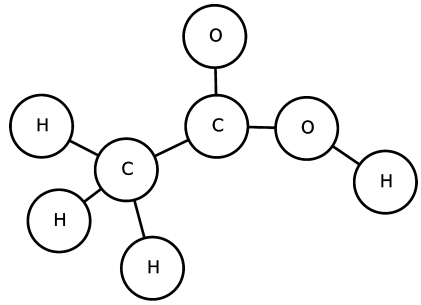


Figure 6. Acetic acid undirected graph.

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

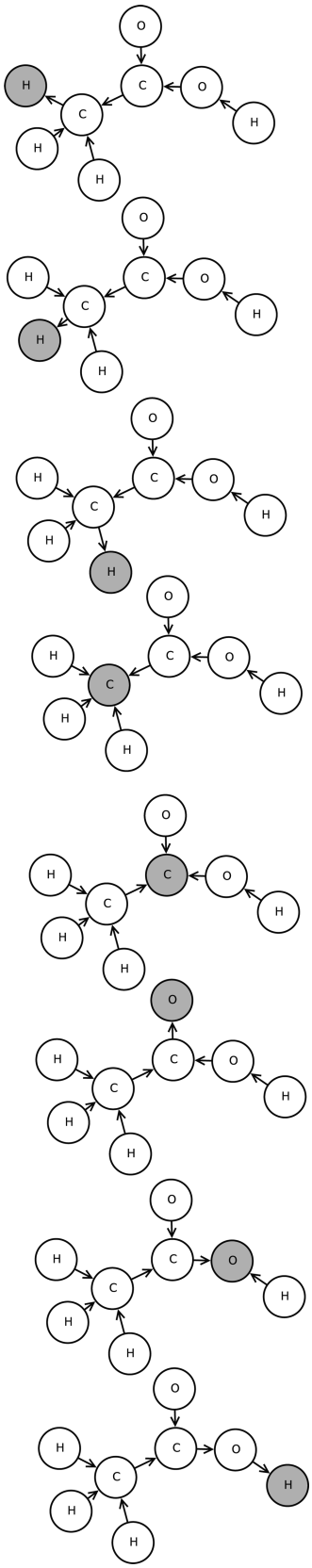


Figure 7. Acetic acid DAGs.

consists in initializing the contextual vector for each source node and each DAG and then propagating the information along the DAG edges, using the neural network M^G to calculate the contextual vector for all the internal nodes up to the root node. Specifically, using the top DAG in Figure 7 with root node v_8 as the example, one must initialize the contextual vector for four source nodes: v_1,v_2,v_3 associated with hydrogen atoms and v_7 associated with an oxygen atom (Figure 8). For these

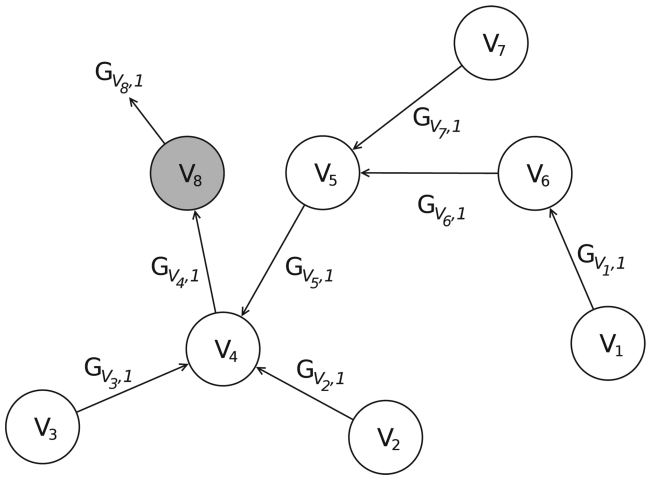


Figure 8. Application of M^G to the first DAG.

four boundary nodes, the contextual vector of the parents is set to 0, and only the input vector associated with the vertex is set to a nonzero value, corresponding to the atom type. Information is then propagated along the DAG structure using eq 2 and the current parameter values for the function M^G to compute the contextual vector of each internal node, and ultimately of the root (sink node) v_8, resulting in the vector $G_{v8,1}$ (Figure 8). The same procedure is applied to the other seven DAGs finally producing eight vectors, each one describing the molecular structure "as seen" from the root of each DAG. The third step consists in generating the vector $G_{\text{structure}}$ describing the whole molecular graph by computing the sum of the eight contextual vectors associated with the eight roots of the eight DAGs (Figure 9). The fourth and final step consists in mapping the vector $G_{\text{structure}}$ using the output function M^O into the property of interest, in this case aqueous solubility. During training, the error between the predicted and true value is computed for each training examples and the parameters of M^G and M^O are then adjusted by gradient descent.

UG-RNN with Contracted Rings. It is well known[52,53] that backpropagation can run into problems of gradient diffusion or exploding or vanishing gradients in deep architectures, including recursive architectures and that generally speaking the severity of these problems tend to increase with the depth of the architectures. Thus here, we consider also an approach which aims at reducing the depth of the recursive architectures essentially by contracting rings in molecular UGs to single points. Cyclic and polycyclic molecules with one or more rings are of course very common.[54] An example of molecule with a single ring is aspirin (Figure 10), and an example of polycyclic molecule is amoxicillin (Figure 11).

If one computes the smallest set of smallest rings[55,56] of the molecular graph G and contracts each ring to a single node, one obtains a new undirected graph Gc with a smaller diameter than G. A new node representing a ring R is assigned a new label R_n where n is the length of the ring capped to the length of the largest ring in the data. The new node is connected to all the nonring nodes that were originally connected to the ring R,

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

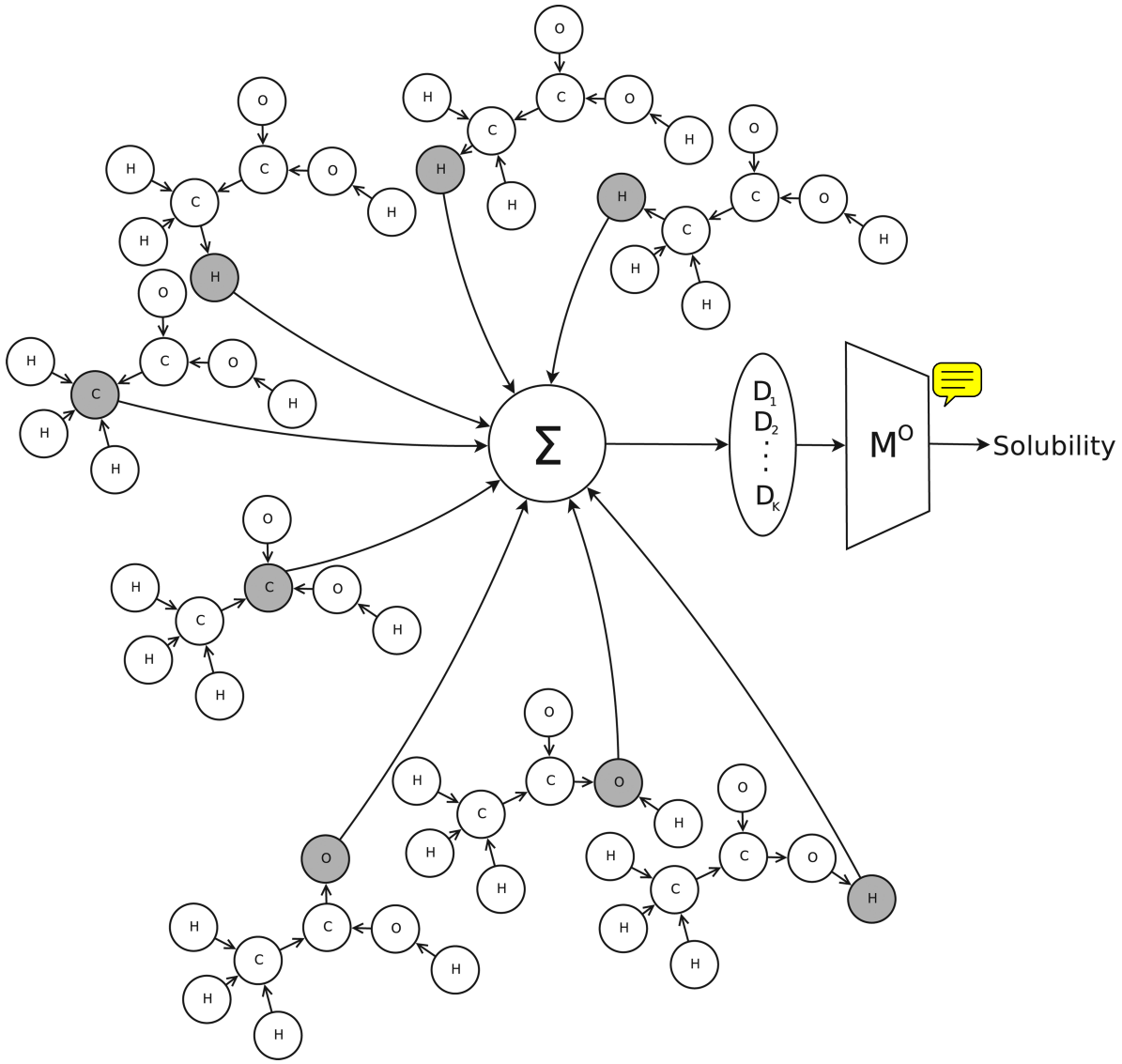


Figure 9. Sum of eight G vectors to produce the vector $G_{\text{structure}} = (D_1,...,D_K)$ corresponding to K descriptors learned from the data. The output function M^O produces the final prediction.

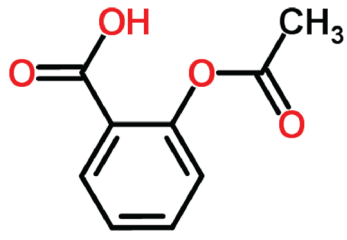


Figure 10. Aspirin.

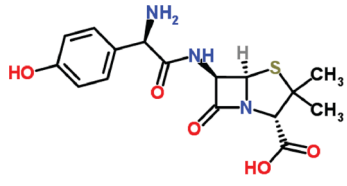


Figure 11. Amoxicillin.

with edge type labels equal to those of the original bonds. In addition, for polycyclic molecules, a new node associated with a ring R can also be connected to other newly created nodes that are associated with rings sharing at least one vertex with R. Although we experimented with various edge labeling rules, in the end using the same label as for single-bond edges for all new edges arising between such newly created nodes provides the most simple and effective solution.

Applying this procedure to the graphs representing aspirin and amoxicillin yields the graphs in Figures 12 and 13, respectively. Thus, in the experiments, we explore also an alternative approach in which the graph representing a

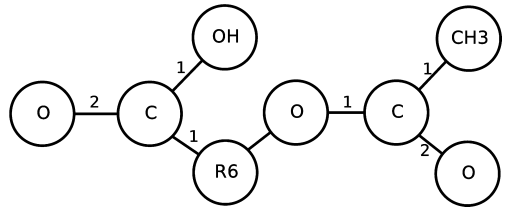


Figure 12. Contracted graph of aspirin with bond types.

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

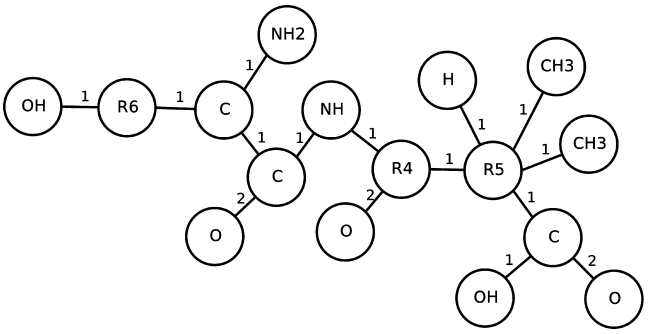


Figure 13. Contracted graph of amoxicillin with bond types.

molecule is first reduced by contracting its rings using the procedure above, and then the UG-RNN approach is applied. We call this approach UG-RNN with contracted rings (UG-RNN-CR).

DATA

To train and test the approach, we use four publicly available benchmark data sets widely used in the solubility prediction literature.

Small Delaney Data Set. This data set[15] originally contained 2874 molecules together with their measured aqueous solubility (log mol/L at 25 °C). This data set is particularly interesting because it can be used as a benchmark for comparisons against the GSE method.[23] As described in Delaney,[15] the GSE was obtained from a set of molecules similar to the ones contained in the "small" Delaney data set. Furthermore, various kernel methods[57] have also been trained and tested on this data set with better results than GSE.

Huuskonen. This data set contains 1026 organic molecules selected by Jarmo Huuskonen[58] from the AQUASOL database[59] and the PHYSPROP database.[60] Molecules are listed together with their aqueous solubility values, expressed in log mol/L at 20−25°. For instance, Frohlich et al.[61] report a squared correlation coefficient of 0.90 for an 8-fold cross-validation, using support vector machines with a RBF (radial basis function) kernel.

Intrinsic Solubility Data Set. This data set contains 74 molecules together with their intrinsic solubility values reported by Bergstrom[62,63] and others.[23,64,65] The data set has been selected by Louis et al.[25] to test a wide range of predictive methods, including linear models, SVMs, and shallow neural networks. Because of its small size, this set is useful for assessing the performances of the UG-RNN method when its training set contains few input/output examples.

Solubility Challenge Data Set. This data set contains 125 molecules selected by Linas et al.[66] using the criteria that molecules ought to contain a ionizable group (pK_a between 1 and 13) and be commercially available. The set is divided into two subsets: a training set containing 97 molecules and a test set containing 28 molecules. The challenge consisted in predicting the solubility of the molecules in the test set using the solubility values of the molecules in the training set.

In general, we use 10-fold cross validation methods to assess the performance of the various UG-RNN predictors. Thus, we divide the data randomly into 10 subsets of equal size and use nine of them for training and the remaining one for testing in all 10 possible combinations. We then compute the average and standard deviation of the performance across the 10 data splits. In addition, each training set consisting of 90% of the original data is randomly split into a proper training set, a small validation set using an 80/20 proportion for the data sets containing more than 1000 examples (Delaney and Huuskonen), and an 85/15 proportion for the smaller data sets (intrinsic solubility and solubility challenge data sets). The validation set is used to fit the hyperparameters of the models (see below). The different proportions are used to match what is reported in the literature for comparison purposes. The details of all the data set splits are given in the Supporting Information.

ADDITIONAL METHODOLOGICAL ASPECTS

As previously mentioned, both the encoding function M^G and the output mapping function M^O are implemented using a neural network. In both cases, we use a standard three-layer neural network architecture with one hidden layer. All neurons use a sigmoid transfer function (tan h) and weights are randomly initialized. In order to reduce the residual generalization error,[67] we use an ensemble of 20 models with a different number of hidden units and features (i.e., the outputs units of M^G), as described in Table 1. The optimal value of the learning rate η is determined by varying it from 10^{-1} to 10^{-4} and keeping the value that gives the lowest RMSE (root mean square error) (see metrics). To facilitate learning, we slightly modify the gradient descent procedure as in refs 50,51. Specifically, the gradient of the error with respect to a weight dw is used to modify the weight according to the simple gradient descent rule $\Delta w = -\eta dw$ only if $|dw| \in [0.1,1]$. Outside this range, to avoid exploding or vanishing gradients, the learning rule is clipped: $\Delta w = -\eta \mathrm{sign}(dw)$ if $|dw|>1$, or $\Delta w = -\eta 0.1 \mathrm{sign}(dw)$ if $|dw| < 0.1$. Each UG-RNN model is trained for 5000 epochs, and the outputs of the best 10 networks, selected by their root mean square error (RMSE) on the validation set, are averaged as an ensemble to compute the prediction on the test set,during each fold of the 10-fold cross validation procedure.

Table 1. Architecture of 20 Encoding Neural Networks M^G and Output Neural Networks M^O

neural network	M^G hidden units	M^G output units	M^O hidden units
model_1	7	3	5
model_2	7	4	5
model_3	7	5	5
model_4	7	6	5
model_5	7	7	5
model_6	7	8	5
model_7	7	9	5
model_8	7	10	5
model_9	7	11	5
model_10	7	12	5
model_11	3	3	5
model_12	4	3	5
model_13	5	3	5
model_14	6	3	5
model_15	7	3	5
model_16	8	3	5
model_17	9	3	5
model_18	10	3	5
model_19	11	3	5
model_20	12	3	5

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

Because of the importance given to the octanol–water partition coefficient in the aqueous solubility literature, we also assess the performances of both the UG-RNN and UG-RNN-CR models using two different inputs for the output network M^O. In addition to the case described above where the input to M^O is the vector $G_{structure}$ alone, we also consider the case where the input consists of $G_{structure}$ plus the log $P_{octanol}$ (calculated using Marvin Beans[68]). In this way, we can partially assess how log $P_{octanol}$ affects the generalization capability of the UG-RNN and UG-RNN-CR models and better understand the kind of information contained in the vector $G_{structure}$.

RESULTS

Metrics. In order to assess the performance of the UG-RNN predictors and compare them with other methods, we use three standard metrics: root mean square error (RMSE), average absolute error (AAE), and Pearson correlation coefficient (R) defined by

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{i}(t_i - p_i)^2} \quad (5)$$

$$\mathrm{AAE} = \frac{1}{n}\sum_{i=1}^{n}|t_i - p_i| \quad (6)$$

$$R = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(p_i - \bar{p})}{\sqrt{\sum_{i=1}^{n}(t_i - \bar{t})^2}\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}} \quad (7)$$

Here, p_i is the predicted value, and t_i is the target value (experimentally observed) for molecule i. In some cases, we use R^2 instead of R as our error metric in order to compare our results with other published results. In the tables, for clarity purposes the best results are marked in bold.

Small Delaney Data Set. Results obtained by 10-fold cross validation on the small Delaney data set are shown in Table 2.

Table 2. Prediction Performances and Standard Deviations Using 10-Fold Cross Validation on the Small Delaney Data Set (1144 molecules)

models	R^2	std R^2	RMSE	std RMSE	AAE	std AAE
UG-RNN	**0.92**	0.02	**0.58**	0.07	**0.43**	0.04
UG-RNN-CR	0.86	0.03	0.79	0.09	0.57	0.06
UG-RNN + log P	0.91	0.02	0.61	0.07	0.46	0.05
UG-RNN-CR + log P	0.91	0.02	0.63	0.05	0.47	0.03
GSE[23]	–	–	–	–	0.47	–
2D kernel (param $d = 2$)[57]	0.91	–	0.61	–	0.44	–

The UG-RNN approach gives the best results for every metric, and the UG-RNN-CR approach does not perform as well. Including the log P information leads to significant improvements for the simplified UG-RNN-CR approach with contracted rings but not for the full UG-RNN approach with no ring contractions. The best UG-RNN models match or surpass the perfomances of the GSE and 2D kernel methods, although generally the differences are within one standard deviation.

Huuskonen Data Set. Results obtained by 10-fold cross validation on the Huuskonen Data set are shown in Table 3 and are consistent with those observed on the small Delaney data set. The UG-RNN method achieves the best results for all the metrics. Adding the log P information does not improve its performance but significantly improves the performance of the restricted UG-RNN-CR method with contracted rings. Published performances obtained with kernel methods are similar to the performance of the UG-RNN models.

Table 3. Prediction Performances and Standard Deviations Using 10-Fold Cross Validation on the Huuskonen Data Set (1026 molecules)

models	R^2	std R^2	RMSE	std RMSE	AAE	sdt AAE
UG-RNN	**0.91**	0.01	**0.60**	0.06	**0.46**	0.04
UG-RNN-CR	0.80	0.04	0.92	0.07	0.65	0.05
UGR-NN + log P	**0.91**	0.01	0.61	0.06	0.47	0.04
UG-RNN-CR + log P	0.89	0.02	0.68	0.06	0.52	0.04
RBF kernel[61]	0.90	–	–	–	–	–

Intrinsic Solubility Data Set. Results obtained by 10-fold cross validation on the intrinsic solubility data set are shown in Table 4. On this small data set, the best performing model is UG-RNN-CR + log P across all three metrics used. The UG-RNN model does not seem to be able to generalize well on this data set, probably because of its small size (only 60 training examples). UG-RNN-CR does not perform well, but it benefits substantially from the inclusion of log P in the input label. In fact, UG-RNN-CR + log P is the only UG-RNN-based method to outperform the best method by Louis et al.[25] The latter is based on a feed-forward neural network whose input layer consists of a set of four descriptors: log P and three topological descriptors, $^1\chi^V$, $\Delta^2\chi$, and ^{2}IC. Given that in the UG-RNN-CR + log P the mapping function is also implemented by a feed-forward neural network with log P as one of the inputs, the only substantive difference between the two approaches is the choice of the other input features. These are chosen in advance and "hand-crafted" in the previously published approach, whereas they are automatically learned from the data in the UG-RNN approach.

Table 4. Prediction Performances and Standard Deviations Using 10-Fold Cross Validation on the Intrinsic Solubility Data Set (74 molecules)

models	R	std R	RMSE	sdt RMSE	AAE	std AAE
UG-RNN	0.64	0.04	0.96	0.01	0.80	0.11
UG-RNN-CR	0.55	0.09	1.05	0.12	0.88	0.12
UG-RNN + log P	0.67	0.02	0.93	0.07	0.77	0.06
UG-RNN-CR + log P	**0.81**	0.01	**0.72**	0.04	**0.51**	0.03
Louis et al.[25]	0.74	–	0.73	–	0.53	–

Solubility Challenge Data Set. Results obtained by 10-fold cross validation on the solublity challenge data set are shown in Table 5. On this data set, we provide a comparison with the relevant work of Hewitt et al.[11] assessing the predictive performances of three different approaches: linear regression, artificial neural networks, and category formation. The input for their models was a vector consisting of 426 molecular descriptor values computed using the Dragon Professional software (version 5.3).[69] For validation purposes, the molecules in the training set were ordered according to their solubility values, and every fifth molecule was taken as a validation molecule. Furthermore, a genetic algorithm approach was used to select the molecular descriptors providing the best predictive

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563–1575

Table 5. Prediction Performance and Standard Deviations Using 10-Fold Cross Validation on the Solubility Challenge Data Set (125 molecules)

models	R^2	std R^2	RMSE	std RMSE	AAE	std AAE
UG-RNN	0.32	0.03	1.41	0.12	1.08	0.10
UG-RNN-log *P*	0.45	0.04	1.27	0.13	1.03	0.11
UG-RNN-CR-log *P*	0.44	0.09	1.28	0.18	1.03	0.16
UG-RNN-Huusk	0.43	0.02	1.16	0.03	0.93	0.03
UG-RNN-Huusk-sub	0.48	0.02	1.11	0.03	0.84	0.01
UG-RNN-log *P*-Huusk	0.54	0.02	1.00	0.03	0.82	0.03
UG-RNN-log *P*-Huusk-sub	0.60	0.02	0.94	0.02	0.71	0.02
UG-RNN-CR-log *P*-Huusk	0.62	0.03	0.96	0.06	0.83	0.06
UG-RNN-CR-log *P*-Huusk-sub	0.67	0.03	0.90	0.06	0.74	0.05
NN-Sol-Chal[11]	0.40	–	1.51	–	–	–
MLR-Sol-Chal[11]	0.51	–	0.95	–	0.77	–
new in silico consesus[11]	0.60	–	0.90	–	0.68	–

Table 6. Differences in Solubility Values between the Solubility Challenge and Huuskonen Data Sets

SMILES	Sol Chal	Huusk
CCN(CC)CCNC(C1=C(C=CC=C2)C2=NC(OCCCC)=C1)=O	−4.39	−3.70
CC/C(C1=CC=C(O)C=C1)=C(C2=CC=C(O)C=C2)/C	−4.43	−4.35
O=S1(C2=CC(S(N)(=O)=O)=C(Cl)C=C2NCN1)=O	−3.68	−2.62
CN(C)CCCN1C2=C(C=CC=C2)CCC3=C1C=CC=C3	−4.11	−4.19
COC1=CC=C2C(C(CC(O)=O)=C(C)N2C(C3=CC=C(Cl)C=C3)=O)=C1	−2.94	−4.62
CCN(CC)CC(NC1=C(C)C=CC=C1C)=O	−1.87	−1.76
NC1=CC=C(S(NC2=NC(C)=CC=N2)(=O)=O)C=C1	−3.12	−2.85
CC1=CC=C(S(NC(NCCCC)=O)(=O)=O)C=C1	−3.46	−3.39

performance on the validation set. For completeness, they also applied several commercially available solubility predictors to the solubility challenge test set. Perhaps surprisingly, they observed that a simple multiple linear regression method (MLR-Sol-Chal) obtained better results than a more complex approach based on neural networks (NN-Sol-Chal); commercially available tools, trained on bigger data sets, obtained better results than MLR-Sol-Chal but not as good as expected.

The UG-RNN models are trained using the solubility challenge training set, and we observe that the best performing models (UGRNN-log *P* and UGRNN-CR-log *P*) obtain worse results than MLR-Sol-Chal but better results than the neural network-based (NN-Sol-Chal). The fact that the UG-RNN-based models outperform the NN-Sol-Chal model provides evidence that the automated feature selection of the UG-RNN method can capture molecular properties that are more informative for the task at hand than precomputed molecular descriptors. We also observe large standard deviations on each metric, indicating that there is considerable variability in the results across different folds. Such variability is a sign of overfitting, and as already suggested by Hewitt at al., this is a clear sign that the training set is too small and does not contain enough information to address the solubility problem with any kind of generality. Moreover, the average Tanimoto similarity of each molecule in the test set to its 10 closest neighbors in the training set is only 0.53, which also suggests that at least some of the molecules in the test set are likely to be outside the applicability domain of the trained models.

Thus, one may suspect that larger training sets could lead to significant performance improvements. To test this hypothesis, we also train several UG-RNN-based models on an expanded training set that includes the molecules from both the solubility challenge data set and the Huuskonen data set. Because the training set should not contain any molecule that is present in the test set, we must remove all shared molecules. Therefore, before starting the training procedure, we compute all the pairwise Tanimoto similarities between the molecules in the Huuskonen data set and the solubility challenge test set. We find that the two data sets share 10 molecules, and some of the shared molecules are assigned significantly different log solubility values (Table 6). Perhaps the most striking and controversial discrepancy is provided by indomethacin, which is known to be practically insoluble in water.[70] It is reported to have a solubility value of −2.94 log units in the solubility challenge data set to be contrasted with a value of −4.62 in the Huuskonen Data set, a difference of more than 1.5 log units. We also noticed differences in excess of 0.5 log units for hydrochlorothiazide and dibucaine. The method adopted to determine experimental solubility values (chasing equilibrium) for the solubility challenge is supposed to ensure a log error of 0.05 units;[66] however, the significant discrepancies with some of the values reported in the literature casts some doubts.[11] In any case, in Table 5, we also report the performance of the UG-RNN-based models when the solubility values of the solubility challenge test set are replaced with the solubility values of the Huuskonen data set (only for the shared molecules). As expected, expansion of the training set results in significant performance improvement for all the metrics, as well as shrinking of the standard deviations. In particular, the UG-RNN-CR-log *P*-Huusk model obtains an average squared correlation R^2 = 0.67 that is even better than the one reported in Hewitt et al. (R^2 = 0.62) obtained with an ensemble of commercial solubility tools (the new in silico consensus). Furthermore, it is interesting to note that the substitution of the Huuskonen solubility values further improves the overall performance of the UG-RNN-based models, with UG-RNN-CR-LogP-Huusk performing the best with an average squared correlation R^2 = 0.67.

dx.doi.org/10.1021/ci400187y | J. Chem. Inf. Model. 2013, 53, 1563−1575

In summary, the results obtained across all four data sets are quite consistent in demonstrating the general effectiveness and competitiveness of the UG-RNN approach in its different forms while also exposing the fundamental problems arising from training sets that are too small or too noisy.

Domain of Applicability. Estimating the domain of applicability (DOA) of a QSAR model is essential to obtain reliable predictions. Schröter et al,[71] for instance, state that predictions of aqueous solubility for molecules whose structure falls outside the DOA are generally poor. In the literature, there are several methods to estimate the DOA of a QSAR model. It is possible to sort them into three categories: range-based methods,[72−74] distance-based methods,[72−75] and probability−distribution-based methods.[72] Here, we employ a distance-based method using Euclidean distance to estimate the DOA of the UG-RNN trained on the small Delaney data set.[15] For completeness, we also use the Tanimoto similarity measure.

During training, as described in the Data section, this data set is randomly partitioned into 10 folds, yielding for each round of cross validation a test set containing 10% of the entire data set (i.e., 115 molecules) and a training set containing the remaining 90%. In addition, 20% of each training set is set aside for validation purposes, leaving in effect 10 training sets containing 78% of the entire data set (i.e., 823 molecules). To estimate the DOA for each of the 10 selected models, we first compute the Euclidean distance between each molecule i in a test set and each molecule j in the corresponding training set by

$$d_{ij} = \sqrt[2]{(D_{i1} - D_{j1})^2 + ... + (D_{iK} - D_{jK})^2} \quad (8)$$

where K is the length of the encoding vectors produced by the first stage of the UG-RNN method. We then compute the average distance D_i for each molecule i in a test set to the corresponding training set by

$$d_i = \frac{1}{T}\sum_{j=i}^{T} d_{ij} \quad (9)$$

where here T denotes the number of molecules in the corresponding training set. When this calculation is applied to each fold, one ends up with an average distance d_i for each molecule in the entire data set and for each one of the 10 selected models. A similar procedure is carried also with other metrics, such as the Tanimoto similarity measure, computed for each molecule by taking the average similarity to its 10 most similar neighbors in the corresponding training set using the FP4 Open Babel fingerprint format (http://openbabel.org/wiki/Tutorial:Fingerprints).

The next step consists in binning the data according to d_i into bins of equal size and calculating the average absolute error AAE_b for each bin b

$$AAE_b = \frac{1}{B}\sum_{i=1}^{B} |t_i^{(b)} - p_i^{(b)}| \quad (10)$$

where B is the number of molecules in each bin. The results are shown in Figure 14 with corresponding Table 7 for the Euclidean distance and in Figure 15 with the corresponding Table 8 for the Tanimoto similarity. In both cases, we use four bins of equal size. One can clearly observe that on average prediction errors increase with distance or dissimilarity, albeit not linearly. Therefore, making predictions only for molecules that have an average distance or dissimilarity from the training set below a certain threshold (e.g., 2.45 ± 0.15 in Euclidean distance) could be a way to improve the reliability of the predictor in a screening pipeline in the absence of larger and more diverse training sets.

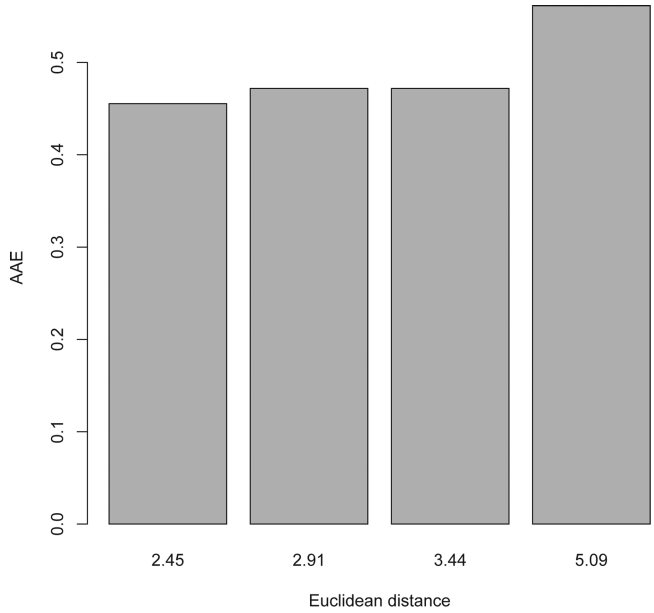


Figure 14. Relationship between Euclidean distances and average absolute error using four bins.

Table 7. AAE (average Euclidean distance) and Standard Deviation[a]

bin number	1	2	3	4
AAE	0.45	0.47	0.48	0.56
distance (av)	2.45	2.91	3.44	5.09
distance (st dev)	0.15	0.14	0.18	1.67

[a]Each bin contains 2860 distances.

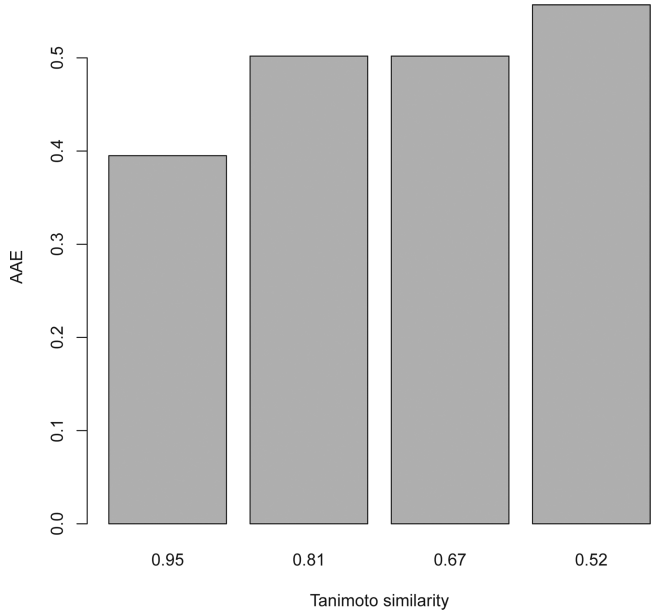


Figure 15. Relationship between Tanimoto similarity values and average absolute error using four bins.

Training Time. The training time for the UG-RNN models scales roughly linearly with the size of the training set and with

Table 8. AAE (average Tanimoto similarit) and Standard Deviation[a]

bin number	1	2	3	4
AAE	0.40	0.50	0.51	0.56
similarity (av)	0.95	0.81	0.67	0.52
similarity (st dev)	0.05	0.04	0.04	0.08

[a]Each bin contains 2860 similarity measures.

the diameter of the molecules being processed. Detailed training times are given in the Supporting Information, but for the data considered here, the average time per molecule and per epoch is 7.8 ms on a good workstation. Thus, the training time is quite reasonable, and much larger data sets could be accommodated by this method. The approach could be further accelerated by using GPUs or parallel distributed implementations if necessary.

Internal Representations. Figure 16 shows the 3D space arrangement of the UG-RNN feature vectors $G_{structure}$ resulting

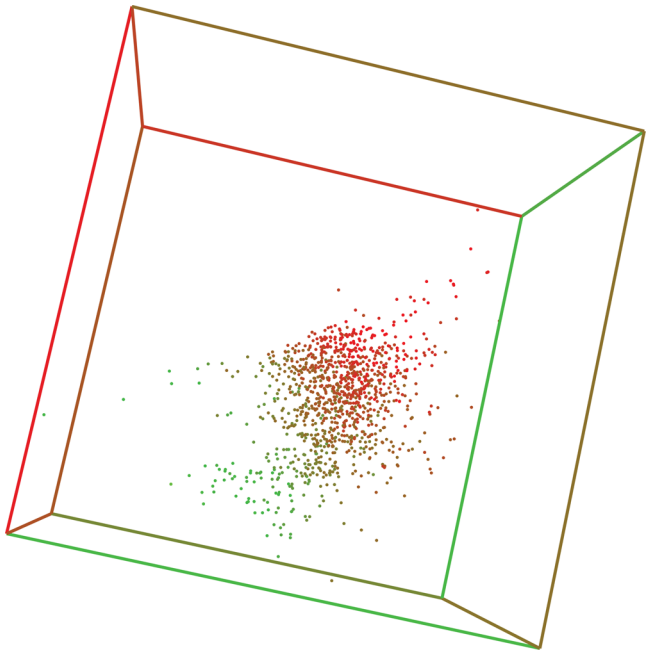


Figure 16. Scatter plot of learned feature vectors for molecules in the small Delaney data set.

from a training process on the small Delaney data set. In this case, we use a small network with $K = 3$ in order to enforce low-dimensional easy-to-visualize feature vectors. Each point correspond to the image of a molecular structure in the space of features. The color of the point, scaled from green to red, represents the aqueous solubility of the corresponding molecule. One can notice immediately that points with the same color tend to be colocated in the 3D plot. This illustrates how the internal representation of molecular structures learned by the UG-RNN models can be correlated with the task at hand, providing further evidence of their capability to automatically extract from the molecular graphs features that are relevant for the prediction of a given property.

CONCLUSION

Experimental results show that the UG-RNN approach can be used to build aqueous solubility predictors that match and sometimes outperform current state-of-the-art methods. One important difference between UG-RNN-based approaches with respect to other methods is the ability to automatically extract internal representations from the molecular graphs that are well suited for the specific tasks. This aspect is an important advantage for a problem like aqueous solubility prediction, where the optimal feature set is not known and may even vary from one data set to the other. It also saves time and avoids other costs and limitations associated with the use of human expertise to select features.

The UG-RNN standard model performs well on both thesmall Delaney data set and the Huuskonen data set, while its results on the intrinsic solubility data set are weaker. A likely explanation for this observation is the particularly small size of the intrinsic solubility data set that leads to overfitting and poor generalization. Moreover, the UG-RNN model does not seem to benefit from the addition of log P information to its set of features, a sign that the log P information is already implicitly contained in the learned features.

The UG-RNN-CR model with contracted rings has weaker predictive capabilities on all the data sets used in this study. This suggests that the loss of information about ring structures negatively affects the generalization capability of the UG-RNN in spite of the decrease in the depth of the recursions that facilitates gradient propagation during learning. On the other hand, the model seems to be rescued when the log P information is added to its features (UG-RNN-CR + log P) providing further evidence that this information is implicitly extracted by the UG-RNN approach.

Finally, a UG-RNN-based Web server for aqueous solubility prediction called AquaSol is available through the ChemDB chemoinformatics portal (cdb.ics.uci.edu) together with downloadable code and other information. However, the domain of applicability of predictors trained on a few hundreds or a few thousands of molecules is bound to be limited in chemical space, and thus, annotating and gathering larger data sets on aqueous solubility or other properties is important for future applications. This is why the main contribution of this paper is not the development of a specific predictor but rather the development of a general deep learning methodology for chemoinformatics problems. The approach uses recursive neural networks adapted to undirected graphs representing molecular structures. A similar deep learning approach could be used also for other molecular representations. For instance, for 1D fingerprint representations, convolutional architectures could be used. For 2D representations based on adjacency matrices, the same 2D-RNN approaches[51] used for protein contact map prediction could be used, as well as their more recent descendants.[34,76] For 3D representations based on atom coordinates, local coordinate information, such as bond lengths and bond angles, could be included in the inputs of the neural networks used for this study. Similar ideas could also be applied to problems involving more than one molecule, for instance, to predict molecular interactions or reaction grammars.[6,7] It is precisely with growing amounts of freely available data and computing power that one can expect deep learning methods to become useful in different areas of chemoinformatics.

ASSOCIATED CONTENT

Supporting Information

Information as mentioned in the text. This material is available free of charge via the Internet at http://pubs.acs.org.

AUTHOR INFORMATION

Corresponding Author

*E-mail: alessandro.lusci@ucdconnect.ie (A.L.); pfbaldi@uci.edu (P.B.).

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

A.L. is funded through a GREP Ph.D. scholarship from the Irish Research Council for Science, Engineering, and Technology. P.B.'s research is supported by the following grants: NSF IIS-0513376, NIH LM010235, and NIH NLM T15 LM07443. We acknowledge OpenEye Scientific Software and ChemAxon for academic software licenses and Jordan Hayes and Yuzo Kanomata for computing support.

REFERENCES

(1) Scholkopf, B.; Smola, A. J. *Learning with Kernels*; MIT Press: Cambridge, MA, 2002.

(2) Ralaivola, L.; Swamidass, S. J.; Saigo, H.; Baldi, P. *Neural Networks* **2005**, *18*, 1093−1110 Special Issue on Neural Networks and Kernel Methods for Structured Domains..

(3) Azencott, C.; Ksikes, A.; Swamidass, S. J.; Chen, J.; Ralaivola, L.; Baldi, P. *J. Chem. Inf. Model.* **2007**, *47*, 965−974.

(4) Ceroni, A.; Costa, F.; Frasconi, P. *Bioinformatics* **2007**, *23*, 2038−2045.

(5) Mahé, P.; Vert, J.-P. *Mach. Learn.* **2009**, *75*, 3−35.

(6) Kayala, M.; Azencott, C.; Chen, J.; Baldi, P. *J. Chem. Inf. Model.* **2011**, *51*, 2209−2222.

(7) Kayala, M.; Baldi, P. *J. Chem. Inf. Model.* **2012**, *52*, 2526−2540.

(8) Waterbeemd, H. V. D.; Gifford, E. *Nat. Rev.* **2003**, *2*, 192−204.

(9) Starita, A.; Micheli, A.; Sperduti, A. *J. Chem. Inf. Comput. Sci.* **2000**, *41*, 202−218.

(10) Fühner, H. *Ber. Dtsch. Chem. Ges.* **1924**, *57B*, 510−515.

(11) Hewitt, M.; Cronin, M. T. D.; Enoch, S. J.; Madden, J. C.; Roberts, D. W.; Dearden, J. C. *J. Chem. Inf. Model.* **2009**, *49*, 2572−2587.

(12) Reynolds, J.; Gilbert, D. B.; Tanford, C. *Proc. Natl. Acad. Sci. U.S.A.* **1974**, *71*, 2925−2927.

(13) Hansch, C.; Quinlan, J. E.; Lawrence, G. L. *J. Org. Chem.* **1968**, *33*, 347−350.

(14) Faller, B.; Ertl, P. *Adv. Drug Delivery Rev.* **2007**, *59*, 533−545.

(15) Delaney, J. S. *J. Chem. Inf. Comput. Sci.* **2003**, *44*, 1000−1005.

(16) Yalkowsky, S. H.; Valvani, S. C. *J. Pharm. Sci.* **1980**, *69*, 912−922.

(17) Kamlet, M. J.; Doherty, R. M.; Abboud, J.-L. M.; Abraham, M. H.; Taft, R. W. *J. Pharm. Sci.* **1986**, *75*, 338−348.

(18) Randic, M. *J. Am. Chem. Soc.* **1975**, *97*, 6609−6615.

(19) Kier, L. B.; Hall, L. H. *Molecular Connectivity in Chemistry and Drug Design*; Academic Press: New York, 1976.

(20) Kier, L. B.; Hall, L. H. *Molecular Connectivity in Structure−Activity Analysis*; John Wiley & Sons: New York, 1986.

(21) Leo, A.; Hansch, C.; Elkins, D. *Chem. Rev.* **1971**, *71*, 525−616.

(22) Leo, A. *Chem. Rev.* **1993**, 1281−1306.

(23) Jain, N.; Yalkowsky, S. *J. Pharm. Sci.* **2001**, *90*, 234−252.

(24) Timmerman, H.; Todeschini, R.; Consonni, V.; Mannhold, R.; Kubiny, H. *Handbook of Molecular Descriptors*; Wiley-VCH: Weinheim, Germany, 2002.

(25) Louis, B.; Agrawal, V. K.; Khadikar, P. V. *Eur. J. Med. Chem.* **2010**, *45*, 4018−4025.

(26) Dearden, J. *Expert Opinion in Drug Discovery* **2006**, *1*, 31−52.

(27) Dannenfelser, R. M.; Paric, M.; White, M.; Yalkowsky, S. *Chemosphere* **1991**, *23* (2), 141−165.

(28) Jorgensen, W.; Duffy, E. *Adv. Drug Delivery Rev.* **2002**, *54* (30), 355−366.

(29) Hinton, G.; Osindero, S.; Teh, Y. *Neural Comput.* **2006**, *18*, 1527−1554.

(30) Bengio, Y.; LeCun, Y. Scaling Learning Algorithms towards AI. In *Large Scale Kernel Machines*; Bottou, L., Chapelle, O., DeCosta, D., Weston, J., Eds.; MIT Press: Cambridge, MA, 2007.

(31) Lee, H.; Grosse, R.; Ranganath, R.; Ng, A. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning*, New York, 2009; pp 609−616.

(32) Lee, H.; Pham, P.; Largman, Y.; Ng, A. Unsupervised Feature Learning for Audio Classification Using Convolutional Deep Belief Networks. In *Advances in Neural Information Processing Systems 22*; Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., Culotta, A., Eds.; NIPS Foundation: La Jolla, CA, 2009; pp 1096−1104.

(33) Hinton, G.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. R. Improving Neural Networks by Preventing Co-Adaptation of Feature Detectors, 2012. http://arxiv.org/abs/1207.0580 (accessed July 1, 2013).

(34) Di Lena, P.; Nagata, K.; Baldi, P. *Bioinformatics* **2012**, *28*, 2449−2457, DOI: 10.1093/bioinformatics/bts475.

(35) Krizhevsky, A.; Sutskever, I.; Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; MIT Press: Cambridge, MA, 2012.

(36) Socher, R.; Pennington, J.; Huang, E. H.; Ng, A. Y.; Manning, C. D. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP '11 Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Stroudsburg, PA, 2011; pp 151_161.

(37) Hinton, G.; Salakhutdinov, R. *Science* **2006**, *313*, 504.

(38) Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H.; Montreal, U. In *Advances in Neural Information Processing Systems 19*; MIT Press: Cambridge, MA, 2007; p 153.

(39) Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.-A.; Vincent, P.; Bengio, S. *J. Mach. Learn. Res.* **2010**, *11*, 625−660.

(40) Baldi, P. *Designs, Codes, Cryptogr.* **2012**, *65*, 383−403.

(41) LeCun, Y.; Matan, O.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D.; Baird, H. S. Handwritten zip code recognition with multilayer networks. *Proc. IEEE* **1990**, *2*, 35−40.

(42) LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. *Proc. IEEE* **1998**, *86*, 2278−2324.

(43) Baldi, P.; Brunak, S.; Frasconi, P.; Pollastri, G.; Soda, G. *Bioinformatics* **1999**, *15*, 937−946.

(44) Baldi, P.; Pollastri, G. *J. Mach. Learn. Res.* **2003**, *4*, 575−602.

(45) Wu, L.; Baldi, P. *Neural Networks* **2008**, *21*, 1392−1400.

(46) Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, 2009.

(47) Baldi, P.; Brunak, S. *Bioinformatics: The Machine Learning Approach*, 2nd ed.; MIT Press: Cambridge, MA, 2001.

(48) Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. *Nature* **1986**, *323*, 533−536.

(49) Baldi, P. *IEEE Trans. Neural Networks* **1995**, *6*, 182−195.

(50) Pollastri, G.; Baldi, P. *Bioinformatics* **2002**, *18*, 62−70.

(51) Baldi, P.; Pollastri, G. *J. Mach. Learn. Res.* **2003**, *4*, 575−602.

(52) Bengio, Y.; Simard, P.; Frasconi, P. *IEEE Trans. Neural Networks* **1994**, *5* (2), 157−166.

(53) Larochelle, H.; Bengio, Y.; Louradour, J.; Lamblin, P. *J. Mach. Learn. Res.* **2009**, *10*, 1−40.

(54) March, J. *Advanced Organic Chemistry: Reactions, Mechanisms, and Structure*, 3rd ed.; New York: Wiley, 1985.

(55) Zamora, A. *J. Chem. Inf. Comput. Sci.* **1976**, *16*, 40−43.

(56) Fan, B. T.; Panaye, A.; Doucet, J. P.; Barbu, A. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 657−662.

(57) Azencott, C.-A.; Ksikes, A.; Swamidass, S. J.; Chen, J. H.; Ralaivola, L.; Baldi, P. *J. Chem. Inf. Comput. Sci.* **2007**, *47*, 965−974.

(58) Huuskonen, J. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 773−777.

(59) Yalkowsky, S. H.; M., D. R. *The Arizona Database of Aqueous Solubility*; College of Pharmacy, University of Arizona: Tucson, AZ, 1990.

(60) Physical/Chemical Property Database(PHYSOPROP). SRC Environmental Science Center: Syracuse, NY, 1994.
(61) Fröhlich, H.; Wegner, J. K.; Zell, A. *QSAR Comb. Sci.* **2004**, *23*, 311−318.
(62) Bergstroem, C.; Strafford, M.; Lazorova, L.; Avdeef, A.; Luthman, K.; Artursson, P. *J. Med. Chem.* **2003**, *46*, 558−570.
(63) Wassvik, C.; Holmen, A.; Bergstrom, C.; Zamora, I.; Artursson, P. *Eur. J. Pharm. Sci.* **2006**, *29*, 294−305.
(64) Faller, B.; Ertl, P. *Adv. Drug Delivery Rev.* **2007**, *59*, 533−545.
(65) Glomme, A.; Maerz, J.; Dressman, J. *J. Pharm. Sci.* **2005**, *94*, 1−16.
(66) Linas, A.; Glen, R.; Goodman, J. *J. Chem. Inf. Model.* **2008**, *48*, 1289−1303.
(67) Hanses, L.; Salamon, L. *IEEE Trans.* **1990**, *12*, 993−1001.
(68) Marvin Beans. ChemAxon. http://chemaxon.com (accessed July 1, 2013).
(69) Dragon Professional Software for Windows. Milano Chemometrics and QSAR Research Group. http://michem.disat.unimib.it/chm/ (accessed July 1, 2013).
(70) O'Neil, M. J. *The Merck Index*, 13th ed.; Merck & Co. Inc.: Whitehouse Station, NJ, 2001.
(71) Schröeter, T. S.; Schwaighofer, A.; Mika, S.; Laak, A. T.; Suelze, D.; Ganzer, U.; Heinrich, N.; Müller, K.-R. *Estimating the Domain of Applicability for Machine Learning Qsar Models: A Study on Acqueous Solubility of Drug Discovery Molecules*; Springer Science+Business Media B.V.: Dordrecht, The Netherlands, 2007.
(72) Netzeva, T. I.; et al. *ATLA, Altern. Lab. Anim.* **2005**, *33* (2), 1−19.
(73) Tetko, I. V.; Bruneau, P.; Mewes, H.-W.; Rohrer, D. C.; Poda, G. I. *Drug Discovery Today* **2006**, *11* (15/16), 700−707.
(74) Tropsha, A. Variable Selection QSAR Modeling, Model Validation, and Virtual Screening. In *Annual Reports in Computational Chemistry*; Spellmeyer, D. C., Ed.; Elsevier: Amsterdam, The Netherlands, 2006; Volume 2, Chapter 7, pp 113−126.
(75) Bruneau, P.; McElroy, N. R. *J. Chem. Inf. Model.* **2006**, *46*, 1379−1387.
(76) Tegge, A. N.; Wang, Z.; Eickholt, J.; Cheng, J. *Nucleic Acids Res.* **2009**, *37*, W515−W518.