*Send Orders for Reprints to reprints@benthamscience.ae*



# Using Deep Learning for Compound Selectivity Prediction

Ruisheng Zhang*, Juan Li, Jingjing Lu, Rongjing Hu, Yongna Yuan and Zhili Zhao

*School of Information Science & Engineering, Lanzhou University, Lanzhou, Gansu 730000, China*

Ruisheng Zhang

**Abstract:** Compound selectivity prediction plays an important role in identifying potential compounds that bind to the target of interest with high affinity. However, there is still short of efficient and accurate computational approaches to analyze and predict compound selectivity. In this paper, we propose two methods to improve the compound selectivity prediction. We employ an improved multitask learning method in Neural Networks (NNs), which not only incorporates both activity and selectivity for other targets, but also uses a probabilistic classifier with a logistic regression. We further improve the compound selectivity prediction by using the multitask learning method in Deep Belief Networks (DBNs) which can build a distributed representation model and improve the generalization of the shared tasks. In addition, we assign different weights to the auxiliary tasks that are related to the primary selectivity prediction task. In contrast to other related work, our methods greatly improve the accuracy of the compound selectivity prediction, in particular, using the multitask learning in DBNs with modified weights obtains the best performance.

**Keywords:** Deep belief networks, compound selectivity, neural network, multitask learning.

## 1. INTRODUCTION

The identification of potential compounds is a time-consuming and costly process and plays an initial and critical role in drug discovery. As a successful lead compound, it not only has to bind to the protein (also known as the target) with high affinity, but also should be selective and does not cause undesirable side effects [1]. The goal of compound selectivity prediction is to identify compounds that only bind to the target of interest with high affinity, but no reaction with other targets so as to minimize the likelihood of side effects. More and more evidence suggests that the majority of drugs and other biologically active compounds are likely to act on more than one target protein, and often many [2, 3]. For instance, Clozapine has a high affinity for a number of serotonin (5-HT2A, 5-HT2C, 5-HT6, 5-HT7), dopamine (D4), muscarinic (M1, M2, M3, M4, M5), adrenergic (1-and 2-subtypes) and other biogenic amine receptors [4]. This means one candidate compound acting on the target protein of interest may also cause side effects to other targets. Thus, the selectivity has been considered as a stringent requirement of a compound to become drug candidates. However, the experimental determination of compound selectivity often takes place later in the drug discovery process, *e.g.*, during binding assays or clinical trials [5]. If the assay or trial fails, all efforts related to drug discovery did in vain. Therefore, an efficient and accurate computational approach to analyze and predict compound selectivity at earlier stages is desirable.

Using computational methods to predict the physical, chemical, or biological properties of molecules has a long history in Cheminformatics and Bioinformatics. For example, Hansch *et al.* [6] developed computational methods to predict Structure-Activity Relationship (SAR) in 1962. In recent years, many researchers have started to develop some machine learning approaches to analyze and predict Structure-Selectivity Relationship (SSR). Ning *et al.* [1] proposed a cascaded learning method and a multitask neural networks learning method to predict the SSR. Moreover, there are other methods, including similarity-search-based approach [7], Bayesian method [8] and SVM [9, 10]. The multitask method developed by Ning *et al.* [1] which gets the state-of-the-art prediction treats their four tasks as the same, and does not outstand the primary task that predicts the selectivity for the target of interest. And there are many practical constraints on classical SSR prediction. SSR data sets may involve a large number of descriptors that are sparse with strong correlations, but the classical methods always cannot handle the large compound descriptors. Due to this, descriptor selection or extraction methods, such as Principal Component Analysis (PCA) or other hand-engineered approaches, have to be applied to reduce the effective number of descriptors from thousands to hundreds or even tens, thus valuable prediction information was lost. Another limitation of classical methods is that there may not be enough training information to build a representative model compared to chemical space. For example, some confirmatory assays only verify less than 20 compounds, which may not allow a learning algorithm to obtain sufficient knowledge. But these existing methods need to maintain many models on different targets, this may overfit the training data. Thus once unsupervised learning or semi-supervised learning well explored, we potentially get additional useful information for better model learning [11].

In this paper, we develop an improved multitask learning method in NNs and also use the multitask learning in DBNs to build SSR models. The first one is built on previously developed techniques and uses a probabilistic classifier with a logistic regression for binary classification of multiple tasks. The second method builds a depth of architecture using DBN learning. On the top layer of DBN fine-turning

*Address correspondence to this author at the School of Information Science & Engineering, Lanzhou University, Lanzhou, Gansu 730000, China; Tel: +86-931-8914000 ext.8421; E-mail: zhangrs@lzu.edu.cn

1875-6697/16 $58.00+.00 © 2016 Bentham Science Publishers

phrase, a logistic regression layer is integrated, too. We employ information from multiple targets to build one multitask SSR model, which includes two SAR tasks (activity for the target of interest and the challenge target) and two SSR tasks (selectivity for the target of interest and the challenge target). On this layer, the selectivity for the target of interest is the primary task, others are auxiliary tasks. The SAR and SSR tasks are learned simultaneously implicitly transferred across one another with multitask learning. Our DBNs can be routinely applied to data sets which contain thousands of compound descriptors without data reduction. Moreover, in the pre-training phrase our method build one model for different targets with unsupervised learning. In addition, multitask learning in fine-tuning phrase contributes to preventing overfitting. Thus our DBNs make better prospective prediction. Although training DBNs is still computationally intensive, using Graphical Processing Units (GPU) can make this issue manageable.

The evaluation results show that the proposed methods outperform the methods developed previously. More precisely, the approach based on multitask learning in DBNs with different weights performs the best in proposed approaches.

This paper is organized as follows. In Section 2, we briefly review the deep learning and multitask learning technologies. Related work with machine learning app-roaches to predict compound selectivity in recent years is provided in Section 3. Our methods for SSR prediction are presented in Section 4. In Section 5, the data sets and evaluation metrics used in our research are presented. Our results for SSR prediction are presented in Section 6. Finally, the conclusions are given in Section 7.

## 2. BACKGROUND AND NOTATION

### 2.1. Deep Learning

Compared to shallow architectures, such as Support Vector Machines (SVMs) and NNs, deep learning [12] is essential to build deep architectures for extracting multiple levels of distributed features of the input automatically. In general terms, deep architectures are composed of multiple layers of parameterized non-linear modules. There are several ways of generating deep architectures, such as Convolutional Neural Networks (CNNs) [13, 14], Stacked Autoencoders (SAs) [15, 16], Recursive Neural Networks (RNNs) [17-19] and DBNs [20, 21].

DBNs are based on Restricted Boltzmann Machines (RBMs), which are particular energy-based models. The RBM is a particular type of random neural network model which has two-layer architecture, symmetric connections and no self-feedback. The energy function $\boldsymbol{E(v,h)}$ of an RBM model is defined as follows:

$$E(v,h) = -b'v - c'h - h'Wv \quad (1)$$

where $\boldsymbol{W}$ represents the weights connecting hidden and visible units and $\boldsymbol{b}$, $\boldsymbol{c}$ are the offsets of the visible and hidden layers, respectively.

Because visible and hidden units of RBMs are conditionally independent given one-another, so

$$p(h|v) = \prod_i p(h_i|v) \quad (2)$$

$$p(v|h) = \prod_j p(v_j|h) \quad (3)$$

The RBMs with binary units (where $\boldsymbol{v_j}$ and $\boldsymbol{h_i} \in \{0,1\}$) get a probabilistic version of the usual neuron activation function:

$$p(h_i = 1|v) = \sigma(c_i + W_i v) \quad (4)$$

$$p\left(v_j = 1\middle|h\right) = \sigma(b_j + W_j' h) \quad (5)$$

And the free energy of an RBM with binary units is:

$$F(v) = -b'v - \sum_i log\ (1 + e^{(c_i + W_i v)}) \quad (6)$$

Contrastive Divergence (CD) is an approximation of the log-likelihood gradient that has been found to be a successful update rule for training RBMs. The chain is initialized with a training example. Samples are obtained after only k-steps of Gibbs sampling. In practice, the good results can be obtained even when $\boldsymbol{k} = 1$.

The update of parameters ($\boldsymbol{W, c_i, b_j}$) according to CD is given as follows:

$$\Delta W = \varepsilon(< vh >_{data} - < vh >_{recon}) \quad (7)$$

$$\Delta b = \varepsilon(< v >_{data} - < v >_{recon}) \quad (8)$$

$$\Delta c = \varepsilon(< h >_{data} - < h >_{recon}) \quad (9)$$

where $\boldsymbol{\varepsilon}$ represents the learning rate, and $<\cdot>_{\boldsymbol{recon}}$ is the distribution of reconstruction from the input.

After the greedy layer-wise unsupervised training of each layer above, which is called pre-training, a supervised training can be used to add extra learning machinery to convert the learned representation into supervised predictions.

In these years, many sophisticated deep learning methods have emerged, including Stacked Denoising Autoencoders (SDAs) [22], Sparse Autoencoder [23], Regularized Auto-encoder [24], Contractive Autoencoder [25], Deep Neural Network [26], Convolutional Deep Belief Networks [27] *etc*. Deep learning has improved the state-of-the-art in almost every field from computer vision to speech recognition to natural language processing to bioinformatics [28-31]. Various studies have reported promising results with the use of deep learning in cheminformatics areas [18, 32-35]. Thus it is promising to apply deep learning methods to predict selectivity properties.

### 2.2. Multitask Learning (MTL)

Multitask learning [36] is a transfer learning that improves the generalization performance by using the domain information contained in the training signals of related tasks. The related tasks learn in parallel while using a common representation and shared hidden layers so as to improve the learning performance. The idea is that the common information related to prediction can be shared among these tasks, and learning them together can generate better performance than learning each task separately.

Many multitask learning approaches have been developed in the last few years, including kernel methods [37], Bayesian models [38], Deep Neural Networks [30, 39] *etc*. There are also researchers that have shown that using

MTL can get promising results in cheminformatics [1, 11, 34, 40].

### 2.3. Notation

This paper follows the definitions and notations given by Ning *et al.* [1]. The protein targets and compounds are denoted by $\boldsymbol{t}$ and $\boldsymbol{c}$, respectively. The sets of targets or compounds are denoted by $\boldsymbol{T}$ and $\boldsymbol{C}$, respectively. For each target $\boldsymbol{t_i}$, its sets of active and inactive compounds are denoted by $\boldsymbol{C_i^+}$ and $\boldsymbol{C_i^-}$, respectively, and the union of the two sets is denoted by $\boldsymbol{C_i}$. We regard the target of interest and a set of challenge targets as $\boldsymbol{t_i}$ and $\boldsymbol{T_i}$, respectively. A compound is always unselective for challenge targets against the target of interest $\boldsymbol{t_i}$. Given a target $\boldsymbol{t_i}$ and a challenge set $\boldsymbol{T_i}$, $\boldsymbol{t_i}$'s selective compounds against $\boldsymbol{T_i}$ are denoted by $\mathbf{S_i^+}$, and the remaining nonselective active compounds are denoted by $\boldsymbol{S_i^-}$. All different SSR classification models can be learned using positive and negative training instances, *i.e.*, $\mathbf{S_i^+}$ and $\boldsymbol{S_i^-} \cup \boldsymbol{C_i^-}$ respectively. Here, we treat both the inactive and nonselective active compounds as negative training instances. However, the compounds in $\boldsymbol{S_i^-} \cup \boldsymbol{C_i^-}$ are usually much more than $\boldsymbol{S_i^+}$. In order to get a reasonable SSR model, the compounds in $\boldsymbol{S_i^-}$ and $\boldsymbol{C_i^-}$ are randomly selected to make sure the same number of positive and negative training compounds.

## 3. RELATED WORK

In recent years, machine learning approaches, such as neural networks, SVMs and Bayesian method have been applied to analyze and predict compound selectivity with some success in cheminformatics. Vogt *et al.* [7] predicted the compound selectivity based on checking if they are similar to the known selective compounds. Stumpfe *et al.* [8] used both k-nearest-neighbor and Bayesian methods to build models to identify selective compounds. Wessermann *et al.* [9, 10] built SSR models based on SVMs. Peltason *et al.* [41] analyzed the compound similarity and selectivity data based on Network-like Similarity Graphs (NSGs), which organize molecular networks in terms of similarity relationships and SAR index values. Ning *et al.* [1] developed neural networks to build both cascaded model and multitask model. The cascaded method decomposes the selectivity prediction into two steps, one model for each step. The multitask method incorporates activity/selectivity models into one multitask model. Ning *et al.* showed that their models had $F_1$ score 0.759 and performed much better than many other conventional selectivity prediction methods.

## 4. PROPOSED METHODS

The methods that we propose for building SSR model are based on NNs and deep learning with multitask learning. Specifically, we employ NNs and RBMs as the underlying machine learning mechanism and determine the selectivity of a compound by building different types of binary classification models. On top layer of NNs and DBNs, a logistic regression layer is employed. Moreover, we incorporate information from multiple tasks to build SSR model on this logistic regression layer. The key insight is that both the compound activity and selectivity for other targets are used to build an SSR model compared to the traditional SSR models that only take into account the labels for the target of interest. In our multitask learning, we have tasks of predicting activity/selectivity both for the target of interest and other challenge targets. The selectivity prediction for the target of interest is the primary task, while others are auxiliary tasks referred to the primary task. If a compound is selective for one target in $\boldsymbol{T_i}$, then this compound is nonselective for $\boldsymbol{t_i}$. Note that our four labels for each training instance are not independent. But we do not describe such dependencies explicitly; only rely on NNs, DBNs and the learning process to implicitly incorporate such constraints from training instances.

### 4.1. SSR Models with MTL in NNs

Given a target $\boldsymbol{t_i}$ and a challenge set $\boldsymbol{T_i}$, the goal of our SSR model is to predict whether a compound is selective for $\boldsymbol{t_i}$ but against all targets in $\boldsymbol{T_i}$ at the same time. The multitask SSR models developed by Ning *et al.* [1] is a multitask SSR model with artificial neural network. On one hand, if the NNs is a binary classification which has only one output. Conventionally, when a prediction score is higher than 0.5, it is considered a positive prediction (0.5 by default serves as a threshold to determine whether a prediction is positive or not). However, different thresholds generate different outputs. For example, if a selectivity output is 0.45 then it will be unselective when using default 0.5 as the threshold, while its actual label is a selective compound for target of interest $\boldsymbol{t_i}$ against challenge targets $\boldsymbol{T_i}$. However, it can be judged to a positive instance if modify the threshold to 0.4. To improve the prediction, Ning *et al.* adopted different thresholds and involved manual operations in setting the sigmoid function. However, such approach wastes much time to search the best threshold and it may not be the best. To deal with this problem, we use a probabilistic classifier as the output, and the label of the maximum probability will be the prediction result, thus to reduce adjusting a threshold parameter. Logistic regression used on the output layer is parameterized by a weight matrix $\boldsymbol{W}$ and a bias vector $\boldsymbol{b}$. Mathematically, which can be written as follows:

$$P(Y = i \mid x, W, b) = softmax_i(Wx + b) = \frac{e^{W_i x + b_i}}{\sum_j e^{W_j x + b_j}} (i \in \{0,1\}) \quad (10)$$

The prediction is then done by taking the max of the vector whose $\boldsymbol{i}$th element is $\boldsymbol{P(Y = i|x)}$ for our primary task:

$$y_{pred} = max_i P(Y = i|x, W, b) \quad (i \in \{0,1\}) \quad (11)$$

In the case of logistic regression, it is very common to use the negative loglikelihood as the loss function. This is equivalent to maximize the likelihood of the data set $D$ under the model parameterized by $\theta$, simulation results using this error function show a better network performance, that is defined as Formula 12.

where $\boldsymbol{D}$ is the set of training data, and $\boldsymbol{w_j'}$ is the weight of multitask, which is 0.25 in our multitask neural networks. ***L2_reg*** is the weight of $L2$ regularization which penalizes certain parameter configurations. Fig. (**1**) shows our SSR model implemented by multilayer perceptron with logistic

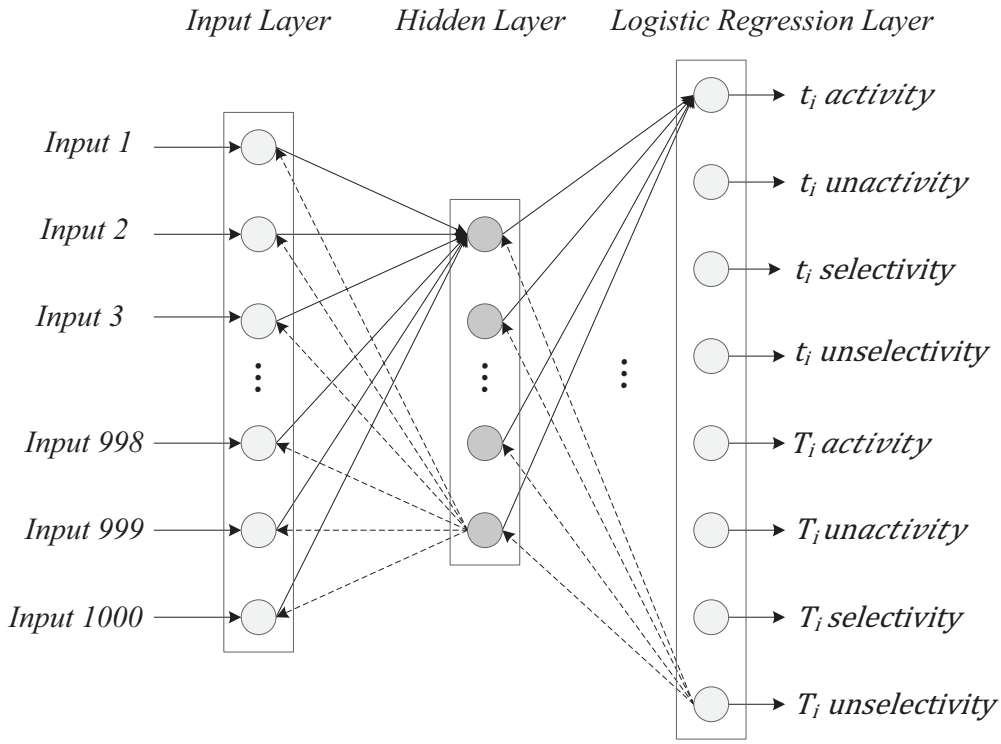


**Fig. (1). Multitask neural networks for target $t_i$ and challenge set $T_i$ with logistic regression layer.** The first two outputs are the first task, the 3th and 4th outputs are our primary task to predict selectivity for $t_i$, the 5th and 6th outputs are the third task, and the 7th and 8th outputs are our last task.

regression layer. The inputs of our neural networks are the 1000 dimension features. We refer to this SSR model as $\boldsymbol{SSR_{m1000-lr}}$.

$$L(\theta = \{W, b\}, D) = -\sum_{i=0}^{D}\sum_{j=0}^{3} w_j' * log\left(P\left(Y = y^i \middle| x^i, W, b\right)\right) + L2\_reg * \sqrt{\sum_W |W|^2} \quad (12)$$

On the other hand, Ning *et al.* [1] applied PCA (which finds the directions of greatest variance in the data set while retaining most of the information) to reduce the 2048 bit binary Chemaxon compound descriptors which describe the chemical structures of the compounds to 1000 dimensions. Such solution decreases the requirements of capacity and memory and increases the efficiency in a smaller dimensions space of inputs. However, although it runs fast, it loses some implicit information between invariance in the training data. In some sense, more descriptors can potentially lead to better selectivity prediction because there is more implicit information between the binary descriptor of compounds. To address this problem, we adopt 2048 bit binary compound descriptors as the inputs to our neural network, which also uses logistic regression as the output layer. And all our code is in Python using Theano and can be accelerated by using of GPU. We refer to this model as $\boldsymbol{SSR_{m2048-lr}}$.

### 4.2. SSR Models with MTL in DBNs

Both the existing SSR methods and $\boldsymbol{SSR_{m1000-lr}}$ models are shallow machine learning for the compounds selectivity prediction. For example, the NNs with only one hidden layer or the SVMs with a linear kernel. Moreover, the feature selection of such models is a completely empirical process which often requires careful engineering and considerable domain expertise; it is independent with prediction task and may lose some key information that can potentially lead to better prediction. In addition, compared to the entire chemical space, we do not have a rich set of training samples to build a representative model, because there are usually few compounds that can selectively bind to a target. In such situation, unsupervised learning approaches can be an attractive alternative to source labeled training data.

To address these problems, we further propose a DBNs architecture with multitask learning to predict compound selectivity, as shown in Fig. (**2**). The DBNs architecture has two phrases: pre-training and fine-tuning.

The pre-training phrase consists of learning a stack of restricted Boltzmann machines (RBMs) adopting CD to pre-train the DBNs, which is a greedy layer-wise unsupervised training of each layer and could extract multiple level of distributed representation of the input compounds. The pre-training phrase has been proposed to initialize the parameters prior to Back Propagation (BP). The loss function here is reconstruction cross-entropy that is defined as:

$$L(x, z, D) = -\sum_{k=1}^{D}[x_k log z_k + (1 - x_k) log\,(1 - z_k)] \quad (13)$$

where $\boldsymbol{D}$ is the set of pre-training data, $\boldsymbol{x}$ is an input, $\boldsymbol{z}$ is a reconstruction of same shape as $\boldsymbol{x}$ through these transformations below:

$$y = \sigma(Wx + b) \quad (14)$$

$$z = \sigma(W'y + b') \quad (15)$$

The fine-tuning phrase is composed by MLP which shares all forward weights with RBMs. Similarly, logistic regression is stacked as the output layer, multitask is applied on logistic regression layer. The selectivity for the target of interest is the primary task, others are auxiliary tasks referred to the primary task. Then it is fine-tuned using BP of error derivatives to build our classification model that directly predicts whether a compound is selective for the target of interest. The loss function is the same with that in multitask neural networks without L2 regularization, and the weight of multitask is also 0.25. Thus the fine-turning loss function is:

$$L(\theta = \{W, b\}, D) = \sum_{i=0}^{D} \sum_{j=0}^{3} w_j' * log\left(P(Y = y^i | x^i, W, b)\right)$$

$$w_j' = 0.25 \tag{16}$$

The additional parameters in the networks associated with auxiliary tasks are used only to aid in the training of the network. After training is completed, the portion associated with the auxiliary tasks is discarded, and the classification is performed identically to a conventional single task classifier. Thus, the SAR and SSR tasks are learned jointly implicitly transferred across one another with multitask learning. Moreover, deep learning can improve the generalization of the shared tasks. We refer this model as $\boldsymbol{SSR_{mDBN}}$. Meanwhile, we refer to another model, which is implemented by DBNs but there is a single compound selectivity task without other auxiliary tasks in the training stage, as $\boldsymbol{SSR_{DBN}}$.

### 4.3. Differentiating Primary Task and Auxiliary Tasks

As we known, the multitask method developed by Ning *et al.* [1] regards their four tasks as the same, and does not outstand the primary task that predicts the selectivity for the target of interest. If the weights of tasks are assigned appropriately, what is learned for other tasks can help the primary task learn better. Motivated by this observation, we further differ the weights in multitask learning with DBNs. Our multitask method treats all these task as four different but related tasks in the training stage, and the primary task would have high weight than others. We refer to this SSR model as $\boldsymbol{SSR_{mDBNw}}$.

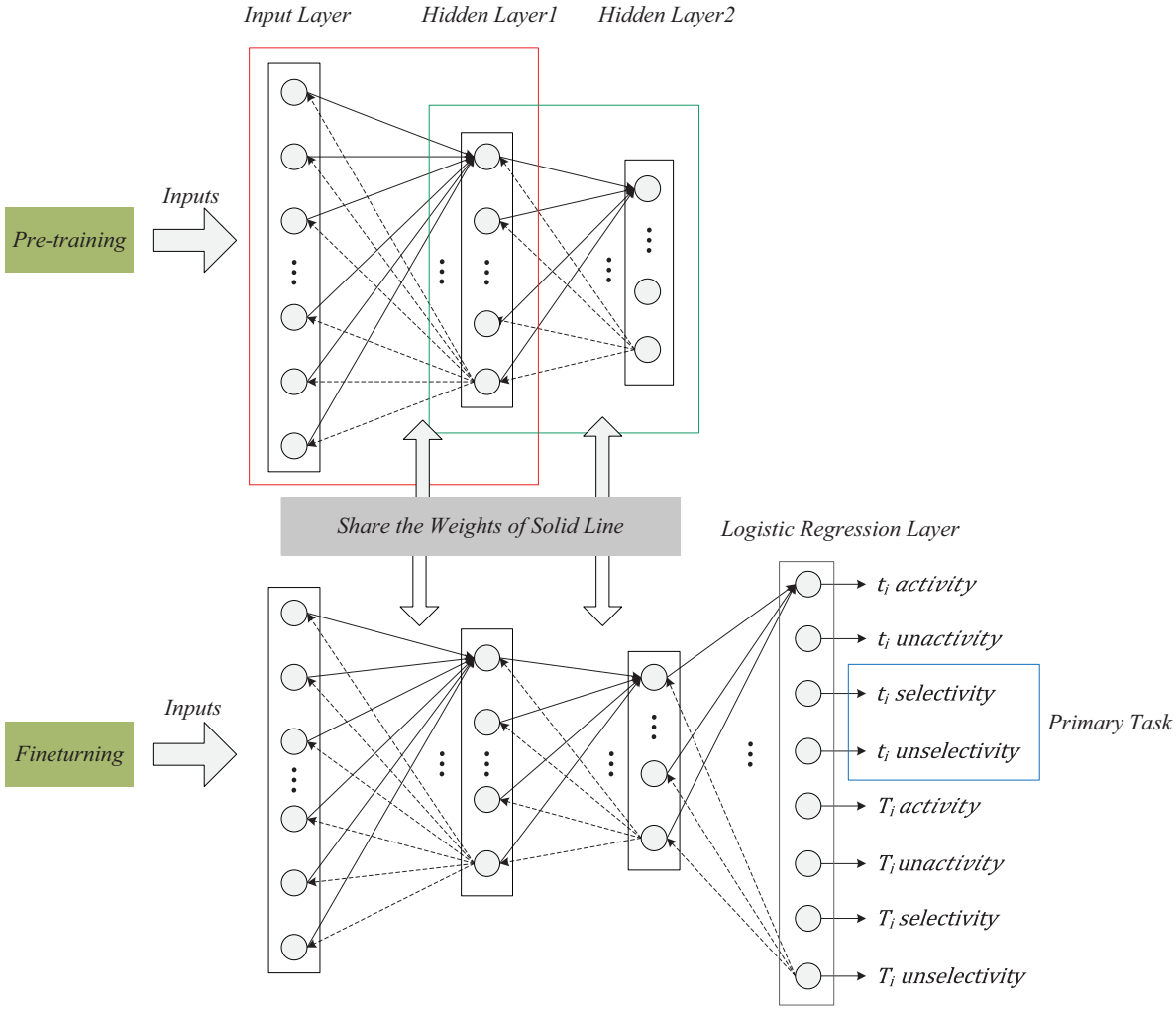


**Fig. (2). Multitask in DBNs for target $t_i$ and challenge set $T_i$ with logistic regression layer.** The pre-training initializes the parameters of the fine-turning phrase, pre-training consists of a stack of RBMs with layer-wise unsupervised training, and fine-turning consists of a stack of multi-hidden layer MLP with logistic regression as output layer with BP training. Primary task represents the selectivity for the target of interest.

## 5. EVALUATION

### 5.1. Data Sets

The performance of the various SSR models is evaluated on set of protein targets and their ligands that compiled from the literature by Ning *et al.* [1]. There are two data sets for experiment test, the first data set DS1 contains 116 individual SSR prediction tasks involving a single target $\boldsymbol{t_i}$ as the target of interest and another single target $\boldsymbol{t_j}$ as its challenge set. In these 116 SSR prediction tasks, the average numbers of active and selective compounds for the target of interest are 172 and 26, respectively. DS1 maximizes the number of interested targets to test for any statistically significant conclusions. Note that there is another data set in Ning *et al.*, however, we do not evaluate our model on it due to its unavailability.

### 5.2. Training Termination Conditions

In the following experiments, we follow the termination condition given by Ning *et al*, and use 0.005 as learning rate, 10000 as the maximum number of epochs for neural networks training and the fine-turning training in DBNs; in addition to the maximum number of training iterations, we apply early-stopping to combat overfitting the training data.

### 5.3. Evaluation Metrics

The performances of the different methods are evaluated *via* a five-fold cross-validation in which the corresponding active compounds and inactive compounds of each target are randomly split into five folds, four folds for model learning and the other fold for testing, and each fold has the same number of selectively active compounds.

The quality of the SSR models is measured using both $F_1$ and $Accuracy$.

(1) $F_1$ is the harmonic mean of Precision and Recall and is defined as:

$$F_1 = \frac{2(Precision)(Recall)}{Precision + Recall} \tag{17}$$

in which $Precision$ is the fraction of the selective compounds classified correctly (*i.e.*, true positive) over all compounds that are classified as selective (*i.e.*, true positive and false positive) by SSR models. $Precision$ is defined as:

$$Precision = \frac{Truepositive}{Truepositive + Falsepositive} \tag{18}$$

$Recall$ is the selective compounds classified correctly (*i.e.*, true positive) over all selective compounds in the

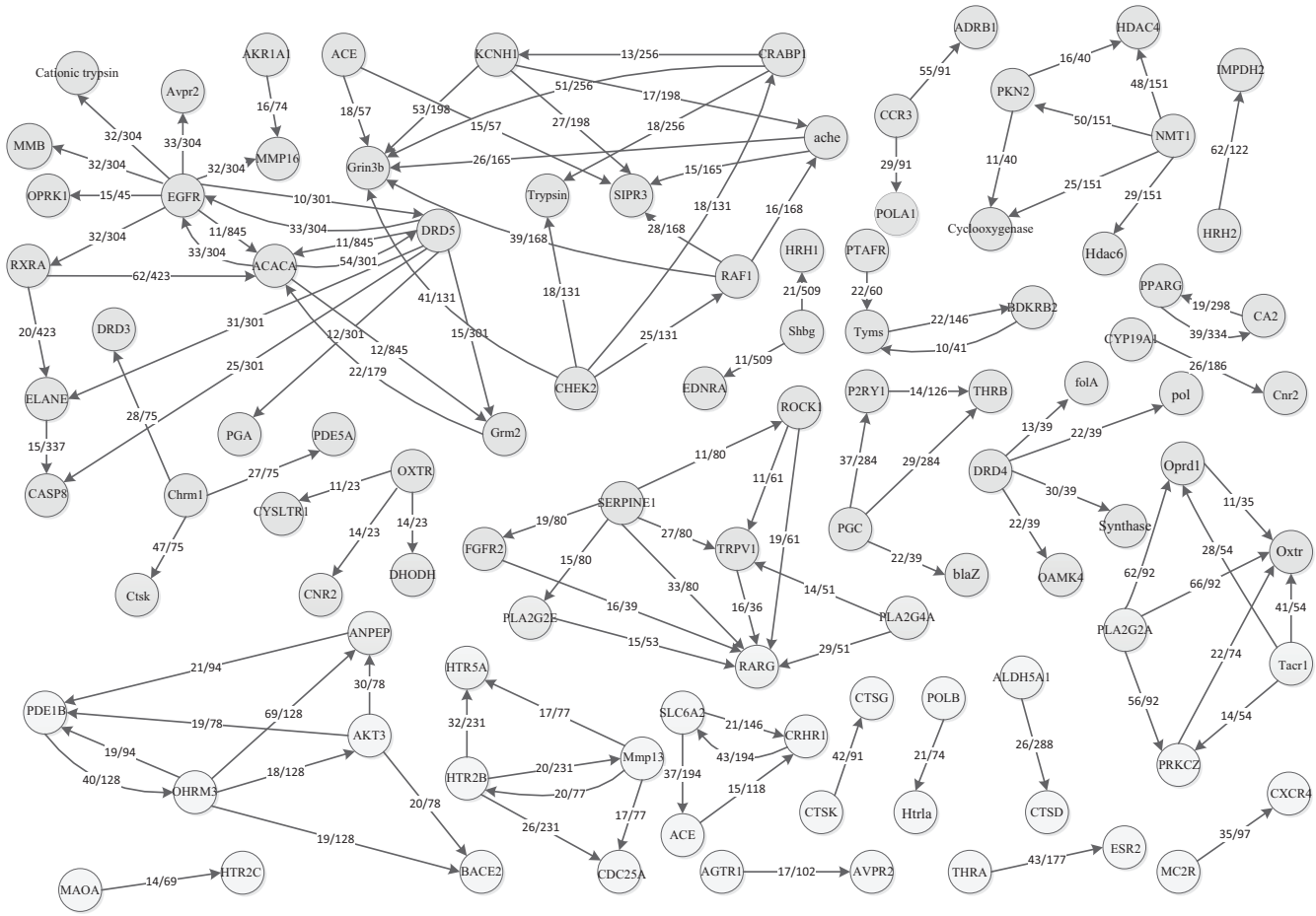


**Fig. (3). Data set (DS1):** The nodes in the graph represent the targets. The directed edge from target A to target B with a label $x / y$ represents that target A has $y$ active compounds, and $x(x <= y)$ compounds are selective for target A against target B [1].

testing data (*i.e.*, true positive and false negative) by SSR models. $Recall$ is defined as:

$$Recall = \frac{Truepositive}{Truepositive + Falsenegative} \quad (19)$$

Intuitively, one of them is very high, $F_1$ will be high. $F_1$ depends on both $Precision$ and $Recall$, and the higher this ratio, the better is our algorithms at predicting which compounds are selective.

(2) The other measure used to test the model is the number of correctly classified examples.

$$Accuracy = \frac{Truepositive + Truenegative}{Truepositive + Falsepositive + Truenegative + Falsenegative}$$

$$= \frac{\sum_{d \in D_1} (t_d = o_d)}{D_1} \quad (20)$$

where $\boldsymbol{D1}$ is the set of test data, $\boldsymbol{t_d}$ is the target label of training instance $\boldsymbol{d}$, and $\boldsymbol{o_d}$ is the output of our model for instance $\boldsymbol{d}$.

Thus, $Accuracy$ does not distinguish the number of correct labels of two classes by showing the probability of the true value of the class labels. In other words, it assesses the overall effectiveness of the model. While $F_1$ is used to calculate the positive class. During the experimental evaluation, we report the average $F_1$ and $Accuracy$ across five folds.

## 6. RESULTS AND DISCUSSION

In this section, we will evaluate different SSR models on DS1, in which each challenge set has only one target and each target may have multiple challenge sets. That is, in DS1, $\boldsymbol{T_j} = \{\boldsymbol{t_j}\}$.

### 6.1. SSR Models with MTL in NNs

We investigate the range of the optimal number of hidden layers and optimal numbers of hidden neurons by performing a grid search on such numbers using SSR MTL models with 64, 128, 256, 512 hidden neurons. The results demonstrated that the model with 256 hidden neurons suffices to learn a good model, and the experiments with additional hidden layer cannot lead to any improvement. Therefore, all considered SSR models reported below use neural networks with 256 hidden neurons, others are not reported. Moreover, we use the BP algorithm for NNs training. BP requires a set of learning parameters, and in the following experiments, we specify such learning parameters as follows: learning rate 0.005, maximum number of epochs 10000 and *L2_reg* 0.0001.

Table **1** shows the performance achieved by the improved multitask SSR models in NNs on DS1, $\boldsymbol{SSR_{mt}}$ is denoted the multitask SSR model developed by Ning *et al.*, and its best average performance is 0.759. The performance achieved by the $\boldsymbol{SSR_{m1000-lr}}$ model is 0.8765. These results show that $\boldsymbol{SSR_{m1000-lr}}$ is capable of computing nonlinear transformations of the input, and its improvement over $\boldsymbol{SSR_{mt}}$ for DS1 is 13.3%. The best average performance of $\boldsymbol{SSR_{m2048-lr}}$ is 0.894 with 15.6% improvement over $\boldsymbol{SSR_{mt}}$. The higher $F_1$ score results from high values of the underlying precision and recall measures, and indicate that taking into account more features of compounds leads to better SSR prediction result. That is because the number of dimensions increases, the accuracy of the SSR model improves. However, it takes more time when no dimensionality reduction is performed if our Python codes not run on GPU.

**Table 1. SSRs Average $F_1$ Score. The second column is the neurons of multiple layers in NNs; the third column is the weights of four tasks, each labeled "0.25−0.25−0.25−0.25" indicating that all tasks are regarded as same.**

| SSR Model | Neurons of Layers | Weights of Tasks | $F_1$ Score |
|---|---|---|---|
| $\mathbf{SSR_{mt}}$ | 1000×256×4 | 0.25−0.25−0.25−0.25 | 0.759 |
| $\mathbf{SSR_{m1000-lr}}$ | 1000×256×8 | 0.25−0.25−0.25−0.25 | 0.876 |
| $\mathbf{SSR_{m2048-lr}}$ | 2048×256×8 | 0.25−0.25−0.25−0.25 | 0.894 |

A finer grained picture of the performance of two different multitask SSR models in MLP involved in DS1 is shown in Fig. (**4**). The histogram shows the $F_1$ scores achieved by each model for the 116 SSR tasks of DS1. For each SSR model, the results are presented in an increasing order. It shows that $\boldsymbol{SSR_{m2048-lr}}$ leads improvements for most individual SSR prediction tasks in contrast to $\boldsymbol{SSR_{m1000-lr}}$. This indicates that PCA loses some information that can potentially lead to overall better selectivity prediction, which should be taken advantage of.

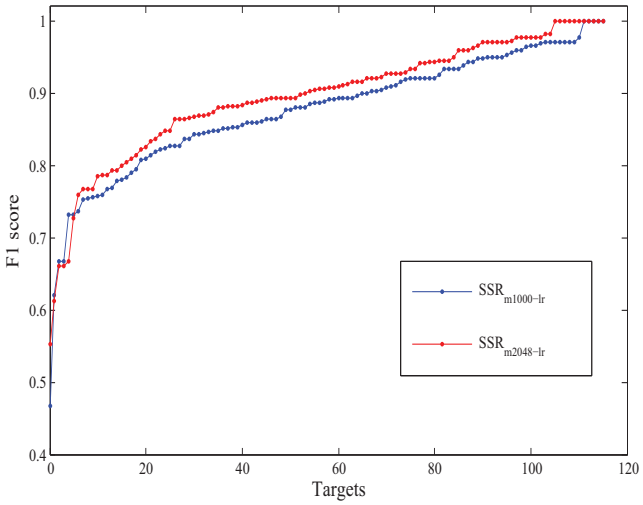


**Fig. (4).** The Comparison of $\boldsymbol{SSR_{m1000-lr}}$ and $\boldsymbol{SSR_{m2048-lr}}$.

### 6.2. SSR Models with MTL in DBNs

We also research on the range of the optimal number of hidden layers and optimal numbers of hidden neurons by performing a grid search on such numbers using SSR MTL models with 2, 3 and 4 hidden layers and 128,256,512 and 1000 hidden neurons on each layer. The results demonstrate that model with 2 hidden layers (1000×256) suffices to learn a better model, and our experiments with additional hidden layer cannot lead to any improvement. In the pre-training

stage, we use the CD algorithm for DBNs pre-training. In the following experiments, we specify learning parameters as follows: pre-training learning rate is 0.018, maximum number of epochs is 10000 and $k$ is 1. While the learning parameters of fine-turning stage are specified as follow: learning rate 0.005 and maximum number of epochs 10000.

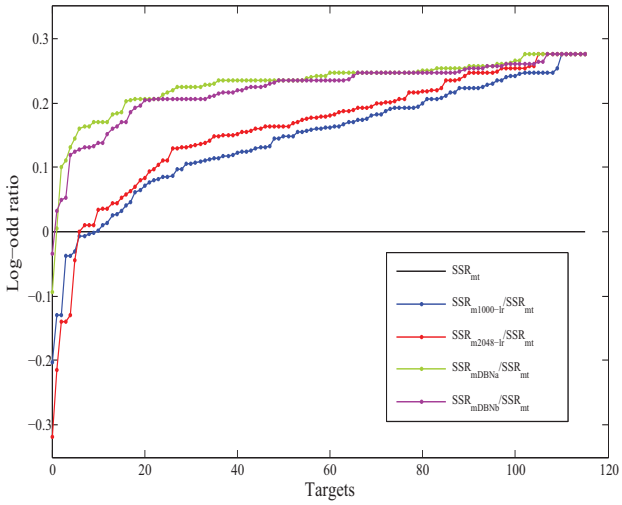


**Fig. (5).** Pairwise improvement.

Table **2** shows the performance achieved by $\boldsymbol{SSR_{mDBN}}$ models. The performances achieved by $\boldsymbol{SSR_{mDBNa}}$ and $\boldsymbol{SSR_{mDBNb}}$ models are 0.955 and 0.947, respectively. It shows that $\boldsymbol{SSR_{mDBN}}$ outperforms $\boldsymbol{SSR_{mt}}$, $\boldsymbol{SSR_{m1000-lr}}$ and $\boldsymbol{SSR_{m2048-lr}}$ significantly, their improvements over $\boldsymbol{SSR_{mt}}$ for DS1 are 20.5% and 19.8%, respectively. The results indicate that the deep architecture is capable of leading to significantly better generalization by utilizing the greedy layer-wise unsupervised training when the number of labelled examples is small. In addition, the results show that training a network with additional hidden layer is harder. While *Accuracy* of $\boldsymbol{SSR_{mDBNa}}$ and $\boldsymbol{SSR_{mDBNb}}$ models are 0.952 and 0.946, respectively. This means that the accuracy on overall class is slightly smaller than $F_1$ of the positive class, thus the model gets more correct labels on the selective compounds. It also shows that the identification of small potential compounds that selectively bind to the target interest is a challenge, but our deep learning discovers selective compounds slightly easier. All our code is in Python using Theano and can be accelerated by using of GPU.

Specially, compared to $\boldsymbol{SSR_{m1000-lr}}$ model, the numbers of hidden neurons of $\boldsymbol{SSR_{mDBNa}}$ model are 1000 and 256, respectively. The input binary compound descriptor is processed by several layers of feature extraction. The feature extraction gets dependence with prediction task through multi-RBM layers. The nonlinear dimensionality reduction with deep learning is more effective than the PCA, since deep learning can build a distributed representation model for compounds and improve the generalization of the shared tasks.

A finer grained picture of the performance of the different SSR prediction tasks involved in DS1 is shown in the plots of Fig. (**5**). These plots show the log ratios of the $F_1$ scores achieved by each model over that achieved by Ning *et al.* for the 116 SSR tasks of DS1. For each SSR model, the results in Fig. (**5**) are presented in an increasing order. All of the plots confirm our expections. Fig. (**5**) shows that our methods including both SSR models with MTL in NNs ($\boldsymbol{SSR_{m1000-lr}}$, $\boldsymbol{SSR_{m2048-lr}}$) and SSR models with MTL in DBNs ( $\boldsymbol{SSR_{mDBNa}}$ , $\boldsymbol{SSR_{mDBNb}}$ ) lead to significant improvements for most individual SSR prediction tasks than $\boldsymbol{SSR_{mt}}$, while SSR models with MTL in DBNs ($\boldsymbol{SSR_{mDBNa}}$, $\boldsymbol{SSR_{mDBNb}}$) perform better than SSR models with MTL in NNs ($\boldsymbol{SSR_{m1000-lr}}$,$\boldsymbol{SSR_{m2048-lr}}$) and the former is more stable for most individual SSR prediction tasks. $\boldsymbol{SSR_{mDBNa}}$ performs slightly better than $\boldsymbol{SSR_{mDBNb}}$ for more individual SSR prediction tasks, and this indicates that adding more hidden layers cannot lead to any improvements and training DBNs with more hidden layers is harder. $\boldsymbol{SSR_{m2048-lr}}$ has slightly higher performance than $\boldsymbol{SSR_{m1000-lr}}$ for most individual SSR prediction tasks.

### 6.3. Differentiating Primary Task and Auxiliary Tasks

In this section, we study how the auxiliary tasks affect the primary task. Table **3** contains the results obtained with multitask learning and single task learning in DBNs.

Table **3** shows the performance achieved by the $\boldsymbol{SSR_{mDBNw}}$ models which is different from the weights of tasks between $\boldsymbol{SSR_{mDBNa}}$ and $\boldsymbol{SSR_{mDBNb}}$. Many of the results reported above have small variations, indicating that the performance of the various $\boldsymbol{SSR_{mDBNw}}$ models is stable over a wide range of weight values in both $\boldsymbol{SSR_{mDBNa}}$ and $\boldsymbol{SSR_{mDBNb}}$. We believe that this is a nice property of our method. However, for the data set, $\boldsymbol{SSR_{mDBN0.25}}$, $\boldsymbol{SSR_{mDBN0.4}}$ and $\boldsymbol{SSR_{mDBN0.6}}$ multitask models outperform the single task $\boldsymbol{SSR_{DBN}}$ on DS1. This indicates that by incorporating additional information (*i.e.* compound activity and selectivity properties against challenge sets, *etc*.) rather than focusing on the selectivity property alone improves selectivity prediction performance. Against all other schemes, $\boldsymbol{SSR_{mDBN0.4}}$ model achieves the best SSR prediction result. This indicates that larger weight for the primary task can get

**Table 2. SSRs Average $F_1$ Score and *Accuracy*. $SSR_{mDBN}$ contains multiple hidden layers. The second column is the neurons of multiple layers in DBNs; the third column is the weights of four tasks, each labeled "0.25−0.25−0.25−0.25" indicating that all tasks are regarded as same. $SSR_{mDBNa}$ and $SSR_{mDBNb}$ represent SSR modes with different number of hidden layers.**

| SSR Model | Neurons of Layers | Weights of Tasks | $F_1$ Score | Accuracy |
|---|---|---|---|---|
| $\boldsymbol{SSR_{mt}}$ | 1000×256×4 | 0.25−0.25−0.25−0.25 | 0.759 | - |
| $\boldsymbol{SSR_{mDBNa}}$ | 2048×1000×256×8 | 0.25−0.25−0.25−0.25 | 0.955 | 0.952 |
| $\boldsymbol{SSR_{mDBNb}}$ | 2048×1000×512×256×8 | 0.25−0.25−0.25−0.25 | 0.947 | 0.946 |

**Table 3. SSRs Average $F_1$ Score. $SSR_{mDBN}$ contains multiple hidden layers. The second column is the neurons of multiple layers in DBNs; the third column is the weights of four tasks, each labeled "0.2−0.4−0.2−0.2" indicating that the weight of four tasks is not regarded as the same role. And the $w$ of $SSR_{mDBNw}$ represents the weight of primary task. $SSR_{mDBNa}$ and $SSR_{mDBNb}$ represent SSR modes with different number of hidden layers. The bold numbers indicate the best average performance over $SSR_{mDBNa}$ and $SSR_{mDBNb}$, respectively.**

| SSR Model | | Neurons of Layers | Weight of Tasks | $F_1$ Score |
|---|---|---|---|---|
| $SSR_{mDBNa}$ | $SSR_{DBN}$ | 2048×1000×256×8 | 0−1−0−0 | 0.937 |
| | $SSR_{mDBN0.25}$ | 2048×1000×256×8 | 0.25−0.25−0.25−0.25 | 0.955 |
| | $SSR_{mDBN0.4}$ | 2048×1000×256×8 | 0.2−0.4−0.2−0.2 | **0.958** |
| | $SSR_{mDBN0.6}$ | 2048×1000×256×8 | 0.2−0.6−0.1−0.1 | 0.950 |
| $SSR_{mDBNb}$ | $SSR_{DBN}$ | 2048×1000×512×256×8 | 0−1−0−0 | 0.941 |
| | $SSR_{mDBN0.25}$ | 2048×1000×512×256×8 | 0.25−0.25−0.25−0.25 | 0.947 |
| | $SSR_{mDBN0.4}$ | 2048×1000×512×256×8 | 0.2−0.4−0.2−0.2 | **0.948** |
| | $SSR_{mDBN0.6}$ | 2048×1000×512×256×8 | 0.2−0.6−0.1−0.1 | 0.943 |

better performance, but the model couldn't full use of other tasks if the weight for the primary task is too large.

## CONCLUSION

In this paper, we implemented an improved multitask learning in NNs and a multitask learning in DBNs for building SSR models and compared them with the SSR model proposed by Ning *et al.* (*i.e.* $\boldsymbol{SSR_{mt}}$). And the improved multitask learning in NNs used a probabilistic classifier with a logistic regression to predict compound selectivity, which outperformed $\boldsymbol{SSR_{mt}}$. The multitask learning in DBNs achieved the best result and outperformed previous methods on a large number of different SSR prediction tasks. This model can not only build a distributed representation for the inputs with unsupervised learning which likely do with the properties of substructure descriptors better than classical descriptor extraction methods, but also improve the generalization of the shared tasks which prevent overfitting. This suggests that semi-supervised learning like deep learning benefits selectivity prediction. At last, the approach based on multitask learning in DBNs with different weights did better than the other approaches, which indicated that the auxiliary multitask in DBNs have a different role against the primary task. Thus our multitask learning method in DBNs is considered to be the state-of-the-art for predicting compound selectivity in drug discovery research.

## CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

## ACKNOWLEDGEMENTS

This work is supported by the Fundamental Research Funds for the Central Universities (Project No. lzujbky-2015-105) and the National Natural Science Foundation of China (Grant No. 21503101).

## REFERENCES

[1] Ning, X., Walters, M.; Karypisxr, G. Improved machine learning models for predicting selective compounds. *J. Chem. Inf. Model.*, **2012**, *52*(1), 38-50.

[2] Hopkins, A.L. Network pharmacology: The next paradigm in drug discovery. *Nat. Chem. Biol*., **2008**, *4*(11), 682-690.

[3] Paolini, G.V.; Shapland, R.H.; van Hoorn, W.P.; Mason, J.S.; Hopkins, A.L. Global mapping of pharmacological space. *Nat. Biotechnol*., **2006**, *24*(7), 805-815.

[4] Roth, B.L.; Sheffler, D.J.; Kroeze, W.K. Magic shotguns *versus* magic bullets: selectively non-selective drugs for mood disorders and schizophrenia. *Nat. Rev. Drug. Discov.*, **2004**, *3*(4), 353-359.

[5] Karaman, M.W.; Herrgard, S.; Treiber, D.K.; Gallant, P.; Atteridge, C.E.; Campbell, B.T.; Chan, K.W.; Ciceri, P.; Davis, M.I.; Edeen, P.T.; Faraoni, R.; Floyd, M.; Hunt, J.P.; Lockhart, D.J.; Milanov, Z.V.; Morrison, M.J.; Pallares, G.; Patel, H.K.; Pritchard, S.; Wodicka, L.M.; Zarrinkar, P.P. A quantitative analysis of kinase inhibitor selectivity. *Nat. Biotechnol.,* **2008**, *26*(1), 127-132.

[6] Hansch, C.; Maloney, P.P.; Fujita, T.; Muir, R.M. Correlation of biological activity of phenoxyacetic acids with hammett substituent constants and partition coefficients. *Nature*, **1962**, *194*(4824), 178-180.

[7] Vogt, I.; Stumpfe, D.; Ahmed, H.E.; Bajorath, J. Methods for computer-aided chemical biology. Part 2: Evaluation of compound selectivity using 2D molecular fingerprints. *Chem. Biol. Drug. Des.,* **2007**, *70*(3), 195-205.

[8] Stumpfe, D.; Geppert, H.; Bajorath, J. Methods for computer-aided chemical biology. Part 3: Analysis of structure-selectivity relationships through single- or dual-step selectivity searching and bayesian classification. *Chem. Biol. Drug. Des.*, **2008**, *71*(6), 518-528.

[9] Wassermann, A.M.; Geppert, H.; Bajorath, J. Searching for target-selective compounds using different combinations of multiclass support vector machine ranking methods, kernel functions, and fingerprint descriptors. *J. Chem. Inf. Model.*, **2009**, *49*(3), 582-592.

[10] Wassermann, A.M.; Geppert, H.; Bajorath, J. Application of support vector machine-based ranking strategies to search for target-selective compounds. *Methods. Mol. Biol.*, 2011, 672, 517-530.

[11] Ning, X.; Rangwala, H.; Karypis, G. Multi-assay-based struct-ure−activity relationship models: improving structure−activity relationship models by incorporating activity information from related targets. *J. Chem. Inf. Model.,* **2009**, *49*(11), 2444-2456.

[12] Hinton, G.E.; Salakhutdinov, R.R. Reducing the dimensionality of data with neural networks. *Science*, **2006**, *313*(5786), 504-507.

[13] Cires¸an, D.C.; Meier, U.; Masci, J.; Gambardella, L.M.; Schmidhuber, J. In: *Flexible, high performance convolutional neural networks for image classification, Proceedings of the twenty-second international joint conference on artificial intelligence,* Manno-Lugano, Switzerland, **2011**; pp. 1237-1242.

[14] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. NIPS Proceedings, **2012**.

[15] Baldi, P. Autoencoders, unsupervised learning, and deep architectures. JMLR: Workshop and Conference Proceedings 27:37-50, **2012**.

[16] Rifai, Salah.; Bengio, Yoshua.; Courville, Aaron.; Vincent, Pascal.; Mirza, M. Disentangling factors of variation for facial expression recognition. Computer Vision - ECCV, **2012**, *7577*, 808-822.

[17] Graves, A.; Fern´andez, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, Proceeding ICML '06 Proceedings of the 23rd international conference on Machine learning, **2006**, pp. 369-376.

[18] Lusci, A.; Pollastri, G.; Baldi, P. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *J. Chem. Inf. Model.*, **2013**, *53*(7), 1563-1575.

[19] Sutskever, I. Training recurrent neural networks. **2013**.

[20] Mohamed, A.-R.; Dahl, G.E.; Hinton, G. Acoustic Modeling Using Deep Belief Networks. *IEEE. Trans. Audio. Speech. Lang. Processing.,* **2012**, *20*(1), 14 - 22.

[21] Hinton, G.E., Osindero, S.; Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural. Comput.,* **2006**, *18*(7), 1527-1554.

[22] Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, **2010**, *11*, 3371-3408.

[23] Goodfellow, I.J.; Aaron, A.C.; Bengio, Y. Large-Scale Feature Learning With Spike-and-Slab Sparse Coding. the 29th International Conference on Machine Learning (ICML), **2012**.

[24] Alain, Guillaume.; Bengio, Y. What regularized auto-encoders learn from the data generating distribution. **2012**. Available at: www.arxiv.org/abs/1211.4246

[25] Rifai, S.; Vincent, P.; Muller, X.; Glorot, X.; Bengio, Y. *Contractive Auto-Encoders: Explicit Invariance During Feature Extraction,* Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, **2011**.

[26] Ciresan, D.C.; Meier, U.; Schmidhuber, J. Transfer learning for Latin and Chinese characters with deep neural networks. The 2012 International Joint Conference on Neural Networks (IJCNN), **2012**.

[27] Lee, H.; Grosse, R.; Ranganath, R.; Ng, A.Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, Proceeding ICML '09 Proceedings of the 26th Annual International Conference on Machine Learning, **2009**, 609-616

[28] Bengio, Y.; LeCun, Y. In: *Large-Scale Kernel Machines*; Léon Bottou, Olivier Chapelle, Dennis DeCoste, Jason Weston, Ed.; MIT Press, **2007**; Vol. *34*. Available at: http://www.iro.umontreal.ca/

[29] Lee, H.; Largman, Y; Pham, P.; Ng, A.Y. Unsupervised feature learning for audio classification using convolutional deep belief networks. NIPS Proceedings, **2009**.

[30] Collobert, R. and J. Weston, A unified architecture for natural language processing: deep neural networks with multitask learning, Proceedings of the 25th international conference on Machine learning, **2008**; ACM: Helsinki, Finland; pp. 160-167.

[31] Lena, Di.; Nagata, P.K.; Baldi, P. Deep architectures for protein contact map prediction. *Bioinformatics*, **2012**, *28*(19), 2449-2457.

[32] Merck Molecular Activity Challenge: Help develop safe and effective medicines by predicting molecular, **2012**. Available from: www.kaggle.com/c/MerckActivity.

[33] Eickholt, J.; Cheng, J. Predicting protein residue-residue contacts using deep networks and boosting. Bioinformatics, **2012**, *28*(23), 3066-3072.

[34] Montavon, G.; Rupp, M.; Gobre, V.; Vazquez-Mayagoitia, A.; Hansen, K.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. Anatole. Machine learning of molecular electronic properties in chemical compound space. *New. J. Phys.*, **2013**, *15*.

[35] Eickholt, J. Cheng, J. DNdisorder: Predicting protein disorder using boosting and deep networks. *BMC. Bioinformatics,* **2013**, 14.

[36] Caruana, R. Multitask Learning. *Mach. Learn.*, **1997**, *28*(1), 41-75.

[37] Evgeniou, T.; Micchelli, C.A.; Pontil, M. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, **2005**, 615-637.

[38] Ni, K.; Carin, L.; Dunson, D. Multi-task learning for sequential data *via* iHMMs and the nested Dirichlet process, Proceedings of the 24th international conference on Machine learning, **2007**, ACM: Corvalis, Oregon, pp. 689-696.

[39] Seltzer, M.L.; Droppo, J. Multi-task learning in deep neural networks for improved phoneme recognition. Proceedings in Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference, Vancouver, BC, USA, May 26-31, **2013**; pp. 6965 - 6969

[40] Jacob, L.; Hoffmann, B.; Stoven, V.; Vert, J.P. Virtual screening of GPCRs: an in silico chemogenomics approach. BMC. Bioinformatics., **2008**, *9*, 363.

[41] Peltason, L.; Hu, Y.; Bajorath, J. From structure-activity to structure-selectivity relationships: Quantitative assessment, selectivity cliffs, and key compounds. ChemMedChem., **2009**, *4*(11), 1864-1873.

**Received: July 3, 2015** **Revised: January 12, 2016** **Accepted: February 17, 2016**