Methods 110 (2016) 64–72

Contents lists available at ScienceDirect

Methods

journal homepage: www.elsevier.com/locate/ymeth

Boosting compound-protein interaction prediction by deep learning

Kai Tian [a,1], Mingyu Shao [a,1], Yang Wang [c], Jihong Guan [b], Shuigeng Zhou [a,*]

[a] Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai 200433, China
[b] Department of Computer Science and Technology, Tongji University, Shanghai 201804, China
[c] School of Software, Jiangxi Normal University, Nanchang 330022, China

ARTICLE INFO

Article history:
Received 12 April 2016
Received in revised form 20 June 2016
Accepted 28 June 2016
Available online 1 July 2016

Keywords:
Compound-protein interaction
Deep learning
Deep neural network (DNN)

ABSTRACT

The identification of interactions between compounds and proteins plays an important role in network pharmacology and drug discovery. However, experimentally identifying compound-protein interactions (CPIs) is generally expensive and time-consuming, computational approaches are thus introduced. Among these, machine-learning based methods have achieved a considerable success. However, due to the nonlinear and imbalanced nature of biological data, many machine learning approaches have their own limitations. Recently, deep learning techniques show advantages over many state-of-the-art machine learning methods in some applications. In this study, we aim at improving the performance of CPI prediction based on deep learning, and propose a method called DL-CPI (the abbreviation of *D*eep *L*earning for *C*ompound-*P*rotein *I*nteractions prediction), which employs *deep neural network* (DNN) to effectively learn the representations of compound-protein pairs. Extensive experiments show that DL-CPI can learn useful features of compound-protein pairs by a layerwise abstraction, and thus achieves better prediction performance than existing methods on both balanced and imbalanced datasets.

1. Introduction

In network pharmacology [1], the assumption of "one drug for one target for one disease" (on which traditional drug discovery is based) is challenged and the relationships between drugs and targets become complicated. One drug may act on multiple targets while there are also proteins that are targeted by two or more compounds. Therefore, identification of the interactions between chemical compounds and proteins plays a critical role in network pharmacology, drug discovery, drug target identification, elucidation of protein functions, and drug repositioning [1,2].

Since experimentally identifying compound-protein interactions (CPIs) is generally expensive and time-consuming [3,4] and has undesirable coverage and throughput [5], various *in silico* approaches have been developed to speed up the experimental process, and meanwhile to cut down the cost.

Up to now, most computational approaches for prediction CPIs are based on the structures of compounds and proteins and/or the interactions among them. For example, Cheng et al. [6] developed multitarget-quantitative structure-activity relationships (QSAR) and chemogenomic methods for CPI prediction, in which substructure patterns and sequence descriptors are calculated for each molecule and protein respectively. Li et al. [7] proposed to predict CPIs by molecular docking, which docks a molecule into the possible binding sites of proteins and ranks the interactions by calculating their interaction energy. Liu et al. [8] detected potential CPIs using pharmacophore mapping approach. In this approach, a given molecule is mapped onto each pharmacophore model (a set of spatial arrangement features essential for a molecule to interact with proteins) of proteins, and the fit value between the molecule and the pharmacophore is calculated. The top ranked hits are then considered CPI candidates. Cobanoglu et al. [9] presented a probabilistic matrix factorization (PMF) method to predict drug-target interactions, where the connectivity matrix of the bipartite graph is decomposed into two matrices of latent variables. Cheng et al. [10,2] developed a network-based inference method that uses the known CPI bipartite network topology similarity to predict novel CPIs, which employs a mass diffusion-like process across the CPI network.

In addition, machine-learning based methods, which have been successfully applied to various prediction problems in biology [11,12], have the potential to effectively learn the relationships among compounds and target proteins to predict new drug-target

* Corresponding author.
E-mail addresses: ktian14@fudan.edu.cn (K. Tian), shaomy@fudan.edu.cn (M. Shao), yang1995t@163.com (Y. Wang), jhguan@tongji.edu.cn (J. Guan), sgzhou@fudan.edu.cn (S. Zhou).
[1] These two authors contributed equally to this work.

http://dx.doi.org/10.1016/j.ymeth.2016.06.024
1046-2023/

interactions [5] from the viewpoint of chemogenomics [13]. Actually, a number of machine learning methods have been proposed to predict CPIs. For instance, in Wang et al.'s [14] work, the substructure descriptors of ligands and sequence descriptors of proteins are extracted and concatenated to form an ligand-protein interaction (LPI) vector and support vector machine (SVM) is used to predict LPIs. Cheng et al. [6] adopted feature selection techniques to reduce the high dimensionality of the chemogenomics space before training SVM and achieved a high AUC (the area under the receiver operating characteristics), while Tabei et al. [15] enhanced the prediction performance of linear SVM by applying an improved minwise hashing algorithm to construct new compact fingerprints for compound-protein pairs. Kim et al. [16] applied both SVM and logistic regression to CPI prediction and found that drug-drug interaction is a promising feature for drug target interaction prediction. Yu et al. [17] employed random forests (RF) by integrating chemical, genomic, and pharmacological information to predict CPIs, and obtained comparable performance to SVM with approximately half time cost.

Though SVM and logistic regression generally do not perform bad, they cannot capture nonlinear relationships among features, which prevents them from performing perfectly [18,19]. Furthermore, the imbalanced nature of data ubiquitously existing in many bioinformatics problems also degrades the performance of many existing predictors, such as RFs [20]. In the era of big biological data, more effective models are urgently needed to do better predictions with the rapidly amassing data.

Recently, deep learning (DL) techniques have been proved advantageous over traditional state-of-the-art machine learning methods in some applications [21]. In bioinformatics, deep learning has also successfully applied to several problems. For instance, Spencer et al. [22] applied a deep network to *ab initio* protein secondary structure prediction where the position-specific scoring matrix (PSSM), the amino acid residues (RES), and the Atchley factors (FAC) were used as features. Lena et al. [23] introduced a deep spatio-temporal architecture that consists of multidimensional stack of learning modules for contact prediction. Leung et al. [24] developed a deep neural network (DNN) model that can jointly predict the splicing patterns in individual tissues and the differences in splicing patterns across tissues. Moreover, Fakoor et al. [25] applied unsupervised feature learning and deep learning methods to handle cancer diagnosis problems by training a more generalized version of cancer classifier. Chicco et al. [26] proposed a deep AutoEncoder model that achieves better performance than other standard machine learning methods on gene annotation prediction. Wang et al. [27] modeled drug target interaction (DTI) relationships with a two-layer graphical model that is known as restricted Boltzmann machine (RBM). They constructed RBMs for all targets with the same parameters. Unterthiner et al. [28] proposed to use a deep neural network with multiple output units to predict DTIs, they formulated DTI prediction as a multi-task learning problem. Recently, Hamanaka et al. [29] trained a deep belief network to predict compound protein interactions (CPIS) and achieved better performance than SVM. However, training a deep belief network is very time consuming as it is a generative model that is trained by layer-wise pre-training of RBMs.

Among the different DL techniques, deep neural network (DNN) is a feedforward, artificial neural network with multiple hidden layers between inputs and outputs. It can automatically learn complex functions that map inputs to outputs, without hand-crafted features or rules [30,31]. Using techniques such as dropout and momentum training to speed up the training procedure, DNN is shown to be potentially suitable for big data including "omics" datasets [24,18].

In this work, we aim at improving the performance of CPI prediction by deep learning. Concretely, we propose a method DL-CPI (*D*eep Learning for Compound-*P*rotein *I*nteractions prediction), to predict new CPIs by constructing a deep neural network (DNN) model and extracting chemical and protein features from compound-protein pairs as input. By appropriately optimizing the hyperparameters of the model, experimental results show that the DL-CPI method outperforms six existing prediction models on both balanced and imbalanced data. The good performance of our method validates the applicability of the DNN model to the CPI prediction problem.

2. Materials and methods

2.1. The DL-CPI pipeline

Fig. 1 shows the pipeline of our DL-CPI method. In this study, we propose a deep learning approach for predicting compound-protein interactions (DL-CPI, Deep Learning for Compound-Protein Interactions). We first retrieve CPIs from public databases as positive samples and generate negative samples by randomly pairing compounds and proteins and keeping those not appearing in the positive set. Then, we extract the chemical fingerprint of each compound and the domain features for each protein from public databases, respectively. For each example (CPI or compound-protein pair), we concatenate the features of the corresponding compound and protein as the feature vector of the example. Next, we input the feature vectors of both positive and negative examples to the DNN model. After hyperparameter adjustment, we train the DNN model and get the DNN predictor. Finally, we evaluate the prediction performance of the DNN predictor using a set of performance metrics and compare our method with existing prediction approaches. In what follows, we describe the major steps of the pipeline in detail.

2.2. Datasets

2.2.1. Compound-protein interactions

We retrieved CPIs of human from the STITCH database (Version 4.0) [32], a comprehensive resource for both known and predicted interactions of compounds and proteins as positive examples. Eventually, we obtained 612,214 interactions between 51,444 unique proteins and 258,936 unique compounds in total.

2.2.2. Compound data

For each compound, we used its basic substructures as features, and constructed a fingerprint (a binary vector where "1" indicates the presence of a certain feature) of features to represent the compound. The fingerprints were obtained from the PubChem database [33], and each compound is represented as a 881-dimension binary vector.

2.2.3. Protein data

We extracted 5523 domains from the Pfam database [34], and represented each protein as a 5523-dimension vector with binary elements (1 or 0). For each element in the domain feature vector, a value of "1" denotes the presence and "0" denotes the absence of the domain, respectively.

2.2.4. Negative samples

The negative samples were generated by random pairing. We first generated 612,214 negative samples (the same number of positive examples). After removing the negative examples with too few domain features, we got 606,469 negative samples in all. We then built both balanced and imbalanced datasets using randomly paired negative samples. We randomly chose positive examples from the total 612,214 positive examples. The number

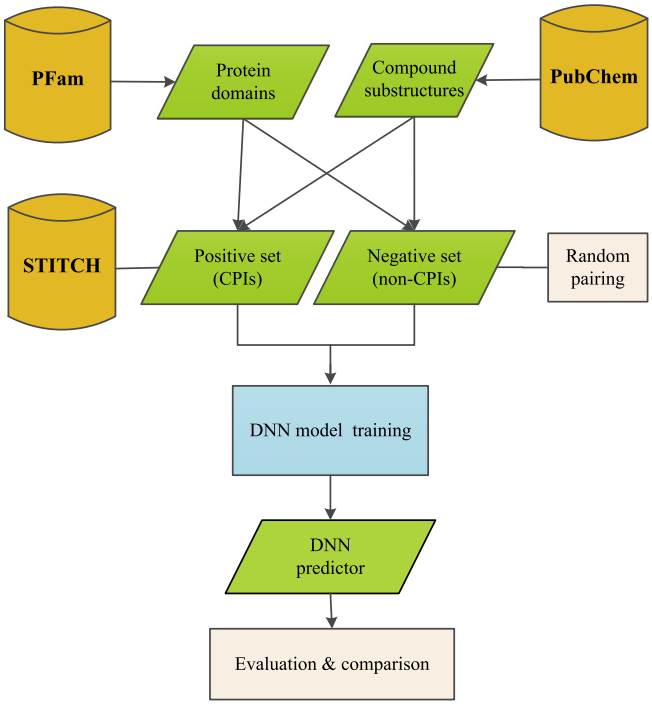

Fig. 1. The pipeline of the DL-CPI method.

Table 1
Statistic of the datasets used in our study. The second row is the ratio of positives over negatives in the datasets, and the following rows give the number of positive samples, the number of negative samples, the number of unique proteins, the number of unique compounds, the number of compound features (substructures), and the number of protein features (domains), respectively.

Dataset	Datasetl	Dataset 2	Dataset3
Ratio	1:1	1:3	1:5
#Positives	612,214	202,156	121,293
#Negatives	606,469	606,469	606,469
#Proteins	7389	7389	7389
#Compounds	258,936	258,936	258,936
#Substructures	881	881	881
#Domains	5523	5523	5523

of positive examples is determined by the given ratio between positive and negative examples, which is set to 1:1, 1:3 and 1:5, respectively. That is, the number of positive examples is about 606,469, 202,156 and 121,293, respectively.

Table 1 presents the statistics of all datasets used in our study.

2.3. The deep neural network (DNN) model

In our study, we formulated the problem of CPI prediction as a binary classification problem. We extracted useful features from compounds and proteins respectively and used them to train the deep neural network (DNN) model to predict new CPIs. The DNN model is comprised of three components: the input layer, the hidden layers, and the output layer, which are similar to general artificial neural networks but differ from them in the number of hidden layers and the training procedure [35]. Fig. 2 illustrates the architecture of the DNN model and its training procedure.

In this paper, the input layer of the model has 6404 features that consist of chemical substructures and protein domains of CPIs or compound-protein pairs. Each hidden layer extracts more discriminative features than the previous layer does. Mathematically, let x denote the input data, $z^{(l)}$ denote the input of the l-th layer, and $a^{(l)}$ denote the activation of the l-th layer, we have the following formulation:

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)} \tag{1}$$

where $a^{(1)} = x$, $W^{(l)}$ is the connection weight matrix between the l-th layer and the $(l+1)$-th layer, $b^{(l)}$ is the bias term in the $(l+1)$-th layer and $a^{(l+1)} = f(z^{(l+1)})$. Here $f(\cdot)$ means the activation function, and the most widely-used activation function is the sigmoid function.

$$\sigma(z^{(l)}) = \frac{1}{(1+e^{(-z^{(l)})})}. \tag{2}$$

However, a recent study shows that the activation function named rectified linear unit (*ReLU*) is more effective and efficient than the sigmoid function [36]. Hence, besides the sigmoid function, we also adopted ReLU as the activation function in the DNN model.

The output layer of the DNN model is a logistic regression classifier that can be formulated as:

$$t = \sigma(a^{(L)}) \tag{3}$$

where L is the total number of layers in the model. Since we formulated the CPI prediction problem as a binary classification task in this study, the output size is 2. Each output value can be explained as the predicted probability of the testing example belonging to a class (positive or negative).

The goal of the DNN model is to minimize the discrepancy between the predicted classes and the real classes of the training examples. That is to minimize the loss function through a certain optimization algorithm.

Specifically, in our study, we employed cross entropy [37] as the loss function,

$$J(W,b) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{2}\left[y_j^{(i)} ln\left(t_j^{(i)}\right) + \left(1-y_j^{(i)}\right) ln\left(1-t_j^{(i)}\right)\right] \tag{4}$$

where n is the number of the training examples, $t^{(i)}$ is the predicted class of the i-th example, and $y^{(i)}$ is the real class of this sample.

We used the weight penalty coefficient under L2-norm to regularize the weight matrix:

$$J_{weightpenalty}(W,b) = J(W,b) + \lambda\sum_{l=1}^{L-1}\|W^{(l)}\| \tag{5}$$

where λ is the weight penalty coefficient. The sparsity constraint is imposed on the sigmoid function,

$$\rho_j^{(l)} = \frac{1}{n}\sum_{i=1}^{n}\left[a_j^{(l)}(x^{(i)})\right], \tag{6}$$

here $\rho_j^{(l)}$ is the average activation value of the j-th hidden unit in the l-th hidden layer, and $\rho_j^{(l)} = \rho$ is defined as the sparsity that is used to control the sparsity of the hidden units. The sparsity is minimized by calculating the KL divergence,

$$\sum_{j=1}^{s_l} KL\left(\rho\|\rho_j^{(l)}\right) \tag{7}$$

where s_l is the number of hidden units in the l-th hidden layer.

Finally, the loss function can be written as:

$$J_{final} = J(W,b) + \lambda\sum_{l=1}^{L-1}\|W^{(i)}\| + \beta\sum_{l=2}^{r}\sum_{j=1}^{s_l} KL\left(\rho\|\rho_j^{(l)}\right) \tag{8}$$

where r denotes the number of layers that use sparsity constraint, and β is the weight that controls the sparsity penalty factor.

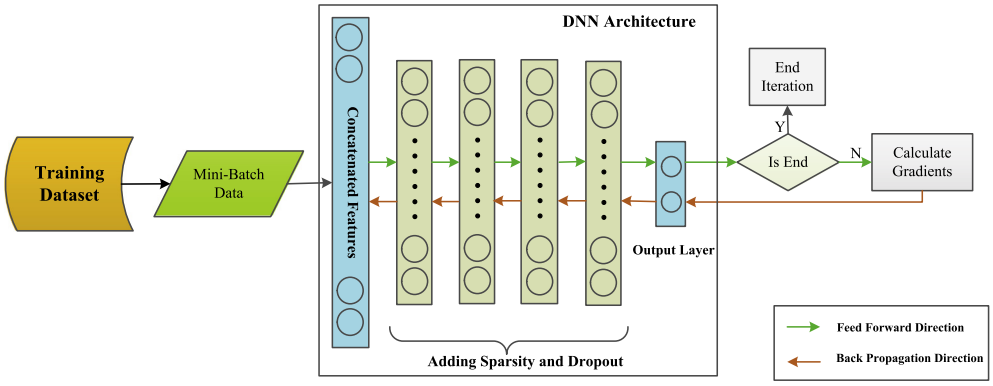

Fig. 2. The architecture and training process of the DNN model. In our study, the DNN model first takes the chemical substructures and the protein domains involved in a CPI or compound-protein pair as the input feature. Then, the DNN model projects the input features into a new space by extracting discriminative information layer-by-layer. The model is trained in an iterative manner, till a certain criterion is met, such as a preset number of iterations (30 in this work) is reached or the error or loss is under a preset threshold. In each iteration, the DNN model is first trained in the feed forward direction and then tuned by supervised back-propagation to minimize the loss function.

We employed Stochastic Gradient Descent (SGD) [36] as the optimization method to train the DNN model. To speed up the training procedure, we also employed the mini-batch method [38] with the batch size equal to 200.

To improve the performance of the DNN model, we further adopted the dropout technique [39]. In this paper, we found that the dropout rate $p = 0.6$ works best. This is a hyper-parameter and the default value in most open-source software packages is 0.5.

Since the hyperparameter adjustment is the trickiest part of training the DNN model, and the performance of a DNN model heavily relies on an appropriate setting of the hyperparameters, we give the details of hyperparameter adjustment in the following subsection.

2.4. Hyperparameter adjustment

The hyperparameters in the DNN model include the number of hidden layers, the number of hidden units in each hidden layer, the weight penalty coefficient and the sparsity coefficient, the learning rate and the dropout rate. In particular, the number of hidden layers and the number of hidden units in each hidden layer define the structure of the DNN model, the weight penalty coefficient and the sparsity coefficient are both regularization terms that are used to prevent the model from overfitting. The learning rate and the dropout rate are mainly used to accelerate the training process while preventing the model from overfitting.

In order to adjust the hyperparameters more efficiently, we sampled a subset from the training dataset to train the DNN model, and applied the tuned hyperparameters to the whole data set.

Therefore, we randomly selected 100,000 samples from the training dataset and adjusted the above-mentioned hyperparameters for this subset. Those samples were used only for tuning hyperparameters. All of the models were trained from scratch with datasets of much larger size. We first adjusted the hyperparameters roughly by grid search and obtained their candidate values or intervals, then we fine-tuned these parameters around the rough values or in the intervals obtained in the grid search stage to achieve the best performance. The selection of the initial values and intervals of these hyperparameters is based on the suggestions in [38,40,41].

For each possible set of parameters, we trained a DNN model separately (the results of hyperparameter adjustment can be found in Section 3).

After the hyperparameter adjustment, we constructed a four-hidden-layer DNN model with 2000 hidden units in each hidden layer, which was used for subsequent prediction.

2.5. Performance measures

To evaluate the DL-CPI method, we employed the following performance metrics: *accuracy* (ACC), *sensitivity* (SN), *specificity* (SP), *Fl-measure* (F1), *area under the receiver-operating characteristic curve* (AUC), and *area under precision recall curve* (AUPR).

Let P and N be the numbers of positives and negatives in the test dataset, TP, FN, TN and FP denote the numbers of true positives, false negatives, true negatives and false positives in the predictions respectively, these performance metrics are defined as follows:

$$ACC = \frac{TP + TN}{P + N} \tag{9}$$

$$SN = \frac{TP}{TP + FN} \tag{10}$$

$$SP = \frac{TN}{TN + FP} \tag{11}$$

$$F1 = 2 * \frac{Rec * Pre}{Rec + Pre} \tag{12}$$

where Pre and Rec are *precision* and *recall*. *precision* is defined as follows:

$$Pre = \frac{TP}{TP + FP}, \tag{13}$$

recall is another name of *sensitivity*, which is generally used in information retrial area. F1 is the harmonic mean of *precision* and *recall*.

AUC and AUPR were computed by utilizing the open source code provided in [42].

3. Results

3.1. Feature visualization

One merit of the DNN model is that it can capture more discriminative features from the raw input data through the hidden layers and thus improves the performance of classification. So in this subsection we show this merit of the DNN model by visualizing the features learned from the data. The features are visualized using t-SNE [43], a technique for high-dimensional data visualization. We visualized the features learned in each hidden layer, which are shown in Fig. 3.

From the feature visualization results, we can see that after the first hidden layer (see Fig. 3(a)), there are still some dots of

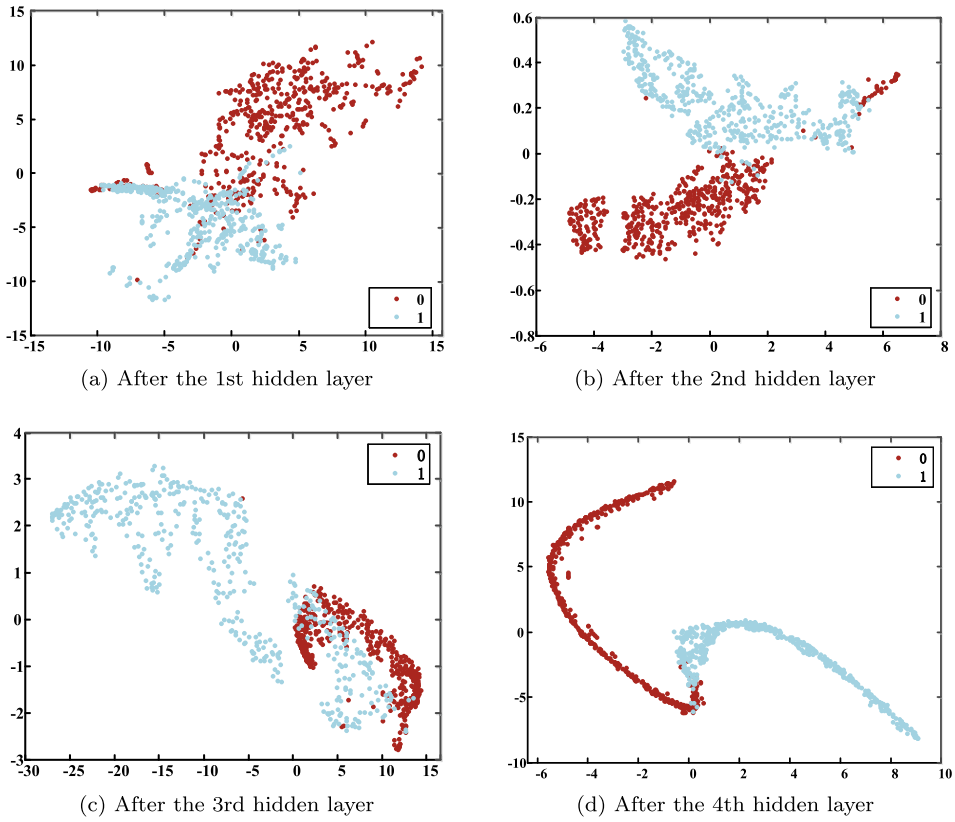

Fig. 3. The features learned with from 1 to 4 hidden layers. From (a) to (d), each panel represents the result after one more hidden layer is added, and the two classes are marked by blue (positive) and red (negative) dots, respectively.

different colors mixing together. With more hidden layers involving in the training, the features become more and more discriminative. After the fourth hidden layer is added (see Fig. 3(d)), the two classes are clearly split. This process shows how the DNN model learns better with more hidden layers being added. That is, the DNN model can extract useful features for prediction by a layer-wise abstraction.

3.2. The results of hyperparameter adjustment

As mentioned in Section 2, the hyperparameters have a significant impact on the performance of the DNN model. For example, the activation function determines how the DNN model projects the raw features into a new feature space, the number of hidden layers decides the degree to which the features are abstracted, and the learning rate influences the speed of the learning process. Here, we aim to obtain a good set of hyperparameters by checking how different hyperparameters affect the prediction performance of the DNN model. Fig. 4 illustrates how the performance of the DNN model changes with respect to the values of different hyperparameters. The results were obtained based on the methods mentioned in Section 2.4. We calculated ACC, AUC and AUPR to demonstrate the performance.

From Fig. 4(a), we can see that the prediction performance is improved significantly in terms of ACC, AUC and AUPR by introducing the ReLU activation function. Here, ReLU*n* indicates that the first *n* hidden layers use ReLU as the activation function, and the other hidden layer(s) use(s) Sigmoid as the activation function. We can also observe that with more hidden layers employing ReLU as activation function, the prediction performance turns better and better steadily.

From Fig. 4(b), we can see that when the number of hidden layers is small (2 or 3 layers), the prediction performance is not quite consistent. This may be because the hidden layers have not yet extracted enough discriminative features before doing prediction, as illustrated in Fig. 3. The performance of the DNN model with four hidden layers reaches the best and then decreases slightly with more hidden layers being added. Since hyperparameter adjustment is carried out on a small subset of data, too many hidden layers may incur overfitting instead. However, for a larger dataset, the number of hidden layers for the best performance may be raised appropriately.

For the rest of hyperparameters, we determined the values based on the results shown in Fig. 4(c)–(e) respectively. From Fig. 4(c), it seems that too many hidden units in each layer may incur overfitting and thus degrades prediction performance.

After hyperparameter adjustment, we trained a DNN model with four hidden layers and 2000 hidden units in each hidden layer. We adopted the ReLU as the activation function for the first three hidden layers and the sigmoid function as the activation function for the last hidden layer. The sparsity coefficient is set to 0.0001 and the weight penalty coefficient is set to 0.00001.

3.3. Comparison with existing prediction methods

Here, we compare our method with several existing CPI prediction approaches, including logistic regression (LR, with L1 regularization and L2 regularization, denoted as LR-L1 and LR-L2, respectively), three variants of support vector machine (SVM) (these models are different in regularization method and loss function, including L2-regularized L2-loss (solving dual) SVM, L2-regularized L2-loss (solving primal) SVM, and L1-regularized L2-loss SVM. For simplicity, they are denoted as SVM-1, SVM-2

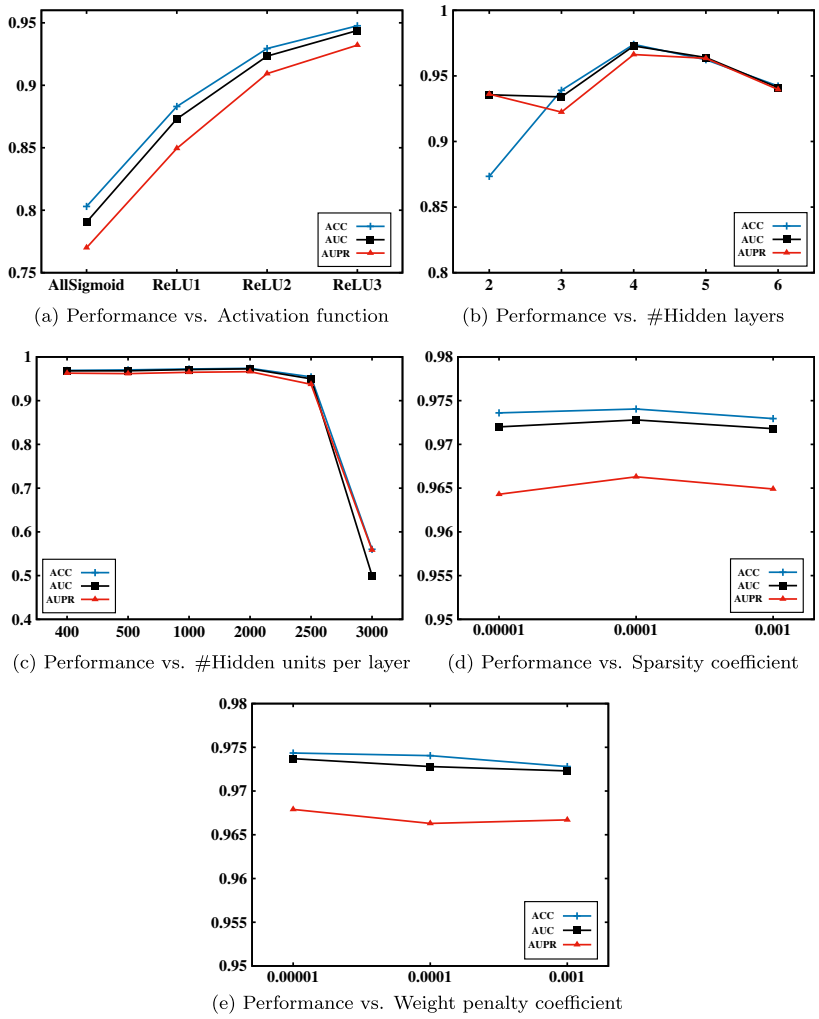

Fig. 4. The performance of the DNN model under different hyperparameters. The horizontal axis of panel (a) denotes the activation function used. Specifically, AllSigmoid means that all the layers employ the sigmoid function as the activation function, ReLU*n* indicates that the first *n* hidden layers use ReLU as the activation function and the rest layer/layers uses/use the sigmoid function as the activation function. The horizontal axis of panel (b) and (c) denotes the number of hidden layers and the number of hidden units in each layer, respectively. The horizontal axis of panel (d) and (e) means the value of the sparsity coefficient and the value of weight penalty coefficient, respectively. The vertical axis of all panels indicates the values of ACC, AUC and AUPR.

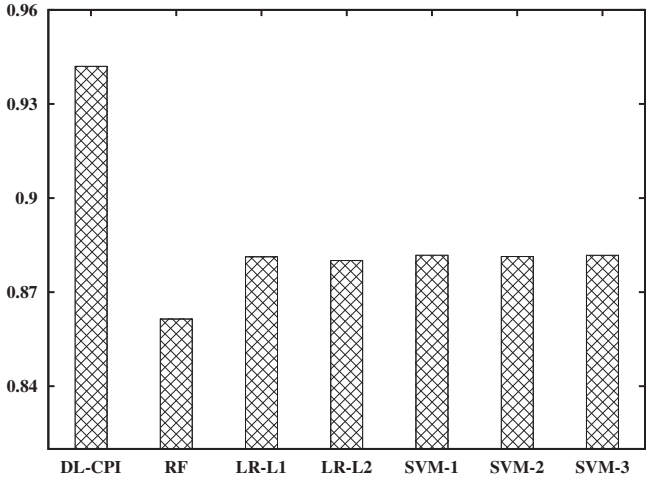

Fig. 5. The accuracy values of different prediction models. Here, LR-L1 means L1-regularized logistic regression, LR-L2 is L2-regularized logistic regression, SVM-1 means L2-regularized L2-loss SVM (dual), SVM-2 indicates L2-regularized L2-loss SVM (primal), and SVM-3 is L1-regularized L2-loss SVM.

and SVM-3, respectively) and random forest (RF). For each dataset, we used 5-fold cross validation to evaluate these methods and computed all performance measures.

Since the accuracy metric considers both positive and negative examples predicted correctly, we first compare the accuracy values of these prediction approaches and present the results in Fig. 5. From Fig. 5, it is obvious that our method DL-CPI outperforms all the other prediction models. The LR and SVM based models have roughly similar accuracy while RF has the lowest accuracy.

In Fig. 6, we present the sensitivity (SN), specificity (SP), and the *F*1-measure (F1) values of our method and the compared methods. Sensitivity and specificity describe how well a model predicts the positive and negative samples respectively, and F1-measure takes both the precision and recall into account. We can see that DL-CPI outperforms all the other models significantly in the three performance metrics. RF has the second highest sensitivity but the lowest specificity, while the LR/SVM-based models have roughly similar and moderate performance in terms of the three measures.

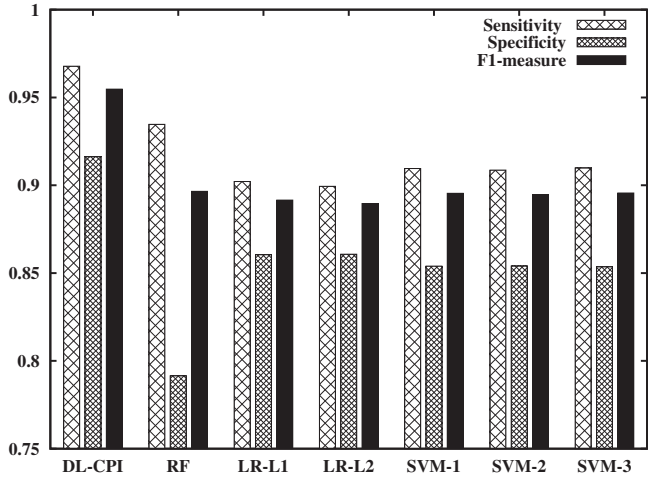

Fig. 6. The sensitivity, specificity, and F1-measure values of different prediction models. Here, LR-L1 means L1-regularized logistic regression, LR-L2 is L2-regularized logistic regression, SVM-1 means L2-regularized L2-loss SVM (dual), SVM-2 indicates L2-regularized L2-loss SVM (primal), and SVM-3 is L1-regularized L2-loss SVM.

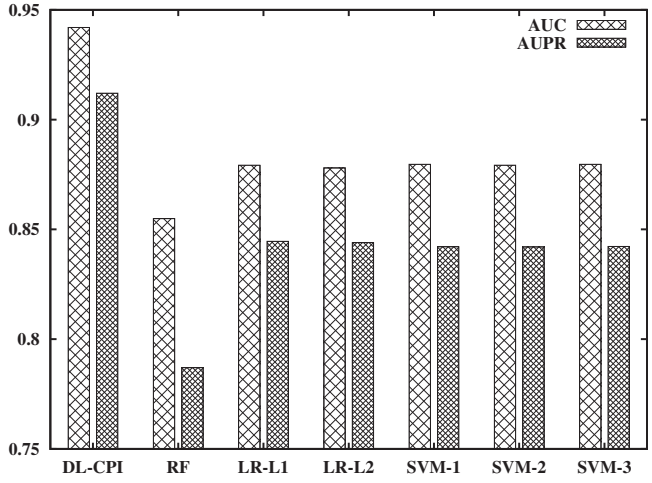

Fig. 7. The AUC and AUPR values of different prediction models. Here, LR-L1 means L1-regularized logistic regression, LR-L2 is L2-regularized logistic regression, SVM-1 means L2-regularized L2-loss SVM (dual), SVM-2 indicates L2-regularized L2-loss SVM (primal), and SVM-3 is L1-regularized L2-loss SVM.

Next, we present the AUC and AUPR values of each prediction method in Fig. 7. In comparison with the other performance measures above, AUC and AUPR give a more comprehensive evaluation of a prediction model. From Fig. 7 we can see that DL-CPI still outperforms all the other models in terms of these two measures. RF ranks the worst, while there is a little difference among the LR/SVM-based models in AUC and AUPR. This result further shows the advantage of our method.

So far, all the experimental results were obtained from the balanced dataset *Datasetl,* that is, the number of positive samples is roughly similar to the number of the negative samples. Since imbalance is one of the major problems that most machine learning algorithms need to handle [20], to evaluate the performance of our method more comprehensively, we also carried out experiments on imbalanced datasets: *Dataset2* and *Dataset3,* where the ratio of negative samples over positive samples is 3:1 and 5:1, respectively. We also conducted 5-fold cross validation on these two datasets with all the evaluated methods, the results are shown in Table 2.

Table 2
Comparison of *ACC*, *SN*, *SP*, *F1*, *AUC*, and *AUPR* on imbalanced datasets. The highest value of each performance metric on each dataset is highlighted in bold.

Datasets	Models	AGG	SN	SP	F1	AUG	AUPR
Dataset2	DL-GPI	**0.932**	0.893	**0.946**	**0.912**	**0.919**	0.8
	RF	0.839	**0.962**	0.5	0.896	0.724	**0.819**
	LR-L1	0.883	0.74	0.93	0.804	0.83	0.68
	LR-L2	0.883	0.747	0.929	0.809	0.835	0.682
	SVM-1	0.887	0.758	0.93	0.817	0.841	0.691
	SVM-2	0.883	0.741	0.931	0.805	0.833	0.683
	SVM-3	0.887	0.757	0.93	0.816	0.841	0.69
Dataset3	DL-CPI	**0.938**	0.826	**0.96**	0.878	**0.893**	0.718
	RF	0.866	**0.965**	0.435	**0.913**	0.687	**0.857**
	LR-L1	0.898	0.642	0.949	0.748	0.792	0.563
	LR-L2	0.899	0.649	0.949	0.753	0.796	0.568
	SVM-1	0.903	0.683	0.946	0.777	0.812	0.584
	SVM-2	0.9	0.653	0.95	0.756	0.799	0.572
	SVM-3	0.903	0.682	0.947	0.776	0.812	0.585

From Table 2, we can see that our method outperforms both the LR-based methods and the SVM-based methods in terms of all performance measures, and also outperforms RF in terms of *ACC*, *SP*, *F1*, and *AUC* on *Dataset2,* and *ACC*, *SP*, and *AUC* on *Dataset3.* RF performs marginally better than DL-CPI in terms of *SN* and *AUPR* on both imbalanced datasets, and outperforms DL-CPI in terms of F1 on *Dataset3.* This is possibly due to its high sensitivity (or recall).

However, it ranks the worst in terms of ACC, *SP* and AUC, which indicates that RF may be not good at identifying true negatives from the testing data. On the contrary, DL-CPI performs fairly good with respect to all performance measures, which implies that DL-CPI can predict CPIs better as a whole. For the other models, the SVM-based models are slightly better than the LR-based models, and there is little difference among different regularization mechanisms.

3.4. Efficiency

We also recorded the training time and testing time of our method and the seven existing methods. The results are presented in Tables 3 and 4. We can see that the logistic regression-based methods and the SVM methods consume substantially less time in training, usually a few minutes, while RF and DL-CPI method cost much more time, usually a few hours. For testing, RF and DL-CPI method still cost more time than the other methods, but the difference is not so significant as in training. In summary, our method DL-CPI has no advantage in efficiency. This is actually the common drawback of deep learning, which need to be overcome in the future.

Note that in this paper, we used liblinear-1.96 package (in C language) for SVM-based models and LR-based models, and Weka-3-6-12 package (in Java language) was used to build the RF model, while our DL-CPI method was implemented in Maltab.

Table 3
Comparison of training time on different datasets.

Methods	Datasets		
	Datasetl	Dataset2	Dataset3
DL-CPI	13 h56 m45 s	11 h41 ml8 s	6 h49 m37 s
RF	5 h56 m33 s	3 h43 m2 s	3 h43 m45 s
LR-L1	**2 m17 s**	**38 s**	**37 s**
LR-L2	4 m15 s	2 m20 s	1 m16 s
SVM-1	13 m26 s	8 ml5 s	7 m11 s
SVM-2	3 m20 s	1 m45 s	1 m53 s
SVM-3	4 m48 s	4 m30 s	6 m38 s

Note that 1) h is short for hours, m is short for minutes and s is short for seconds. 2) The shortest training time on each dataset is in bold text.

Table 4
Comparison of testing time on different datasets.

Methods	Datasets		
	Datasetl	Dataset2	Dataset3
DL-GPI	3 m39 s	2 m26 s	2 m11 s
RF	3 m40 s	2 m34 s	2 m3 s
LR-L1	4 s	10 s	2 s
LR-L2	4 s	10 s	3 s
SVM-1	4 s	10 s	2 s
SVM-2	4 s	10 s	2 s
SVM-3	4 s	10 s	3 s

This is the minor reason for DL-CPI's inefficiency, because Matlab is an interpreted language that is slower than compiled languages.

4. Conclusion

In this work, to improve CPI prediction we proposed the DL-CPI method that employs the DNN model to extract discriminative features from compounds and proteins. Concretely, we first trained a DNN model on a small subset of data to obtain a good set of hyperparameters, then we applied the obtained set of hyperparameters to training better DNN models on larger datasets for predicting CPIs. We evaluated our method by employing multiple performance measures and comparing our method with several existing prediction approaches. Experimental results on both balanced and imbalanced datasets show that our method DL-CPI generally performs better than the compared approaches.

One factor that contributes to the good performance of DL-CPI is the tuning of hyperparameters. For example, after adopting ReLU as the activation function, the performance is boosted remarkably. The performance of DL-CPI can be further improved by employing more advanced regularization techniques such as Dropconnect [44]. Another factor is the large amount of data available. The DNN model can extract more discriminative information from massive data. This merit makes it promising in the era of big biological data.

Acknowledgments

A short version of this paper was originally included in the Proceedings of BIBM 2015. This work was supported by National Natural Science Foundation of China (NSFC) under Grant No. 61272730. Jihong Guan was partially supported by the Program of Shanghai Subject Chief Scientist under Grant No. 15XD1503600.

References

[1] A.L. Hopkins, Network pharmacology, Nat. Biotechnol. 25 (10) (2007) 1110–1111.
[2] F. Cheng, Y. Zhou, W. Li, G. Liu, Y. Tang, Prediction of chemical-protein interactions network with weighted network-based inference method, PLoS ONE 7 (7) (2012) e41064.
[3] J.A. DiMasi, R.W. Hansen, H.G. Grabowski, The price of innovation: new estimates of drug development costs, J. Health Econ. 22 (2) (2003) 151–185.
[4] R.L. Schilsky, J. Allen, J. Benner, E. Sigal, M. McClellan, Commentary: tackling the challenges of developing targeted therapies for cancer, Oncologist 15 (5) (2010) 484–487.
[5] T. Pahikkala, A. Airola, S. Pietila, S. Shakyawar, A. Szwajda, J. Tang, T. Aittokallio, Toward more realistic drug-target interaction predictions, Briefings Bioinf. 16 (2) (2015) 325–337.
[6] F. Cheng, Y. Zhou, J. Li, W. Li, G. Liu, Y. Tang, Prediction of chemical-protein interactions: multitarget-QSAR versus computational chemogenomic methods, Mol. BioSyst. 8 (9) (2012) 23732384.
[7] H. Li, Z. Gao, L. Kang, H. Zhang, K. Yang, K. Yu, X. Luo, W. Zhu, K. Chen, J. Shen, X. Wang, H. Jiang, TarFisDock: a web server for identifying drug targets with docking approach, Nucl. Acids Res. 34 (2) (2006) 219–224.
[8] X. Liu, S. Ouyang, B. Yu, Y. Liu, K. Huang, J. Gong, S. Zheng, Z. Li, H. Li, H. Jiang, Pharmmapper server: a web server for potential drug target identification using pharmacophore mapping approach, Nucl. Acids Res. 38 (suppl 2) (2010) W609–W614.
[9] M.C. Cobanoglu, C. Liu, F. Hu, Z.N. Oltvai, I. Bahar, Predicting drug-target interactions using probabilistic matrix factorization, J. Chem. Inf. Model. 53 (12) (2013) 3399–3409.
[10] F. Cheng, C. Liu, J. Jiang, W. Lu, W. Li, G. Liu, W. Zhou, J. Huang, Y. Tang, Prediction of drug-target interactions and drug repositioning via network-based inference, PLoS Comput. Biol. 8 (5) (2012) e1002503.
[11] A.L. Swan, A. Mobasheri, D. Allaway, S. Liddell, J. Bacardit, Application of machine learning to proteomics data: classification and biomarker identification in postgenomics biology, OMICS 17 (12) (2013) 595–610.
[12] A. Lavecchia, Machine-learning approaches in drug discovery: methods and applications, Drug Discovery Today 20 (3) (2015) 318–331.
[13] S. Jaroch, H. Weinmann, Chemical Genomics: Small Molecule Probes to Study Cellular Function, vol. 58, Springer Science & Business Media, 2007.
[14] F. Wang, D. Liu, H. Wang, C. Luo, M. Zheng, H. Liu, W. Zhu, X. Luo, J. Zhang, H. Jiang, Computational screening for active compounds targeting protein sequences: methodology and experimental validation, J. Chem. Inf. Model. 51 (11) (2011) 28212828.
[15] Y. Tabei, Y. Yamanishi, Scalable prediction of compound-protein interactions using minwise hashing, BMC Syst. Biol. 7 (6) (2013) 1.
[16] S. Kim, D. Jin, H. Lee, Predicting drug-target interactions using drug-drug interactions, PLoS ONE 8 (11) (2013) e80129.
[17] H. Yu, J. Chen, X. Xu, Y. Li, H. Zhao, Y. Fang, X. Li, W. Zhou, W. Wang, Y. Wang, A systematic prediction of multiple drug-target interactions from chemical, genomic, and pharmacological data, PLoS ONE 7 (5) (2012) e37608.
[18] D. Quang, Y. Chen, X. Xie, DANN: a deep learning approach for annotating the pathogenicity of genetic variants, Bioinformatics 31 (5) (2015) 761–763.
[19] D.A. Freedman, Statistical Models: Theory and Practice, Cambridge University Press, 2009.
[20] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[21] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[22] M. Spencer, J. Eickholt, J. Cheng, A deep learning network approach to ab initio protein secondary structure prediction, IEEE/ACM Trans. Comput. Biol. Bioinf. (TCBB) 12 (1) (2015) 103–112.
[23] P.D. Lena, K. Nagata, P.F. Baldi, Deep spatio-temporal architectures and learning for protein structure prediction, Adv. Neural Inf. Process. Syst. (2012) 512–520.
[24] M.K. Leung, H.Y. Xiong, L.J. Lee, B.J. Frey, Deep learning of the tissue-regulated splicing code, Bioinformatics 30 (12) (2014) i121–i129.
[25] R. Fakoor, F. Ladhak, A. Nazi, M. Huber, Using deep learning to enhance cancer diagnosis and classification, in: Proceedings of the ICML Workshop on the Role of Machine Learning in Transforming Healthcare, JMLR: W&CP, Atlanta, Georgia, 2013.
[26] D. Chicco, P. Sadowski, P. Baldi, Deep autoencoder neural networks for gene ontology annotation predictions, in: Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, ACM, 2014, pp. 533–540.
[27] Y. Wang, J. Zeng, Predicting drug-target interactions using restricted boltzmann machines, Bioinformatics 29 (13) (2013) i126–i134.
[28] T. Unterthiner, A. Mayr, G. Klambauer, M. Steijaert, J.K. Wegner, H. Ceulemans, S. Hochreiter, Deep learning for drug target prediction, in: Proceedings of the NIPS Workshop on Representation and Learning Methods for Complex Outputs, Montreal, Canada, 2014.
[29] M. Hamanaka, K. Taneishi, H. Iwata, Y. Okuno, Prediction of compound-protein interactions based on deep learning methods, Proc. Symp. Chemoinf. 2015 (2015) 46–49.
[30] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[31] Y. Bengio, Learning deep architectures for AI, Found. Trends@ Mach. Learning 2 (1) (2009) 1–127.
[32] M. Kuhn, D. Szklarczyk, S. Pletscher-Frankild, T.H. Blicher, C. von Mer-ing, L.J. Jensen, P. Bork, STITCH 4: integration of protein-chemical interactions with user data, Nucl. Acids Res. 42 (Database issue) (2014) D401–D407.
[33] Y. Wang, J. Xiao, T.O. Suzek, J. Zhang, J. Wang, S.H. Bryant, Pub-Chem: a public information system for analyzing bioactivities of small molecules, Nucl. Acids Res. 37 (Web Server issue) (2009) 623–633.
[34] R.D. Finn, A. Bateman, J. Clements, P. Coggill, R.Y. Eberhardt, S.R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E.L. Sonnhammer, J. Tate, M. Punta, Pfam: the protein families database, Nucl. Acids Res. 42 (Database issue) (2014) D222–D230.
[35] M. Spencer, J. Eickholt, J. Cheng, A deep learning network approach to ab initio protein secondary structure prediction, IEEE/ACM Trans. Comput. Biol. Bioinf. 12 (1) (2015) 103–112.
[36] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580, 2012.
[37] P. Golik, P. Doetsch, H. Ney, Cross-entropy vs. squared error training: a theoretical and experimental comparison, in: Interspeech, 2013, pp. 1756–1760.
[38] Y. Bengio, Practical recommendations for gradient-based training of deep architectures, Neural Networks: Tricks of the Trade, vol. 7700, Springer, Berlin Heidelberg, 2012, pp. 437–478.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdi-nov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learning Res. 15 (1) (2014) 1929–1958.
[40] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.
[41] J.S. Bergstra, R. Bardenet, Y. Bengio, B. Kegl, Algorithms for hyper-parameter optimization, Adv. Neural Inf. Process. Syst. (2011) 2546–2554.
[42] T. van Laarhoven, S.B. Nabuurs, E. Marchiori, Gaussian interaction profile kernels for predicting drug-target interaction, Bioinformatics 27 (21) (2011) 3036–3043.
[43] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learning Res. 9 (85) (2008) 2579–2605.
[44] L. Wan, M. Zeiler, S. Zhang, Y.L. Cun, R. Fergus, Regularization of neural networks using dropconnect, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), 2013, pp. 1058–1066.