RESEARCH **Open Access**

A comparison of embedding aggregation strategies in drug–target interaction prediction

Dimitrios Iliadis[1*], Bernard De Baets[1], Tapio Pahikkala[2] and Willem Waegeman[1]

*Correspondence:
dimitrios.iliadis@ugent.be

[1] Department of Data Analysis and Mathematical Modelling, Ghent University, Coupure Links 653, 9000 Ghent, Belgium
[2] Department of Computing, University of Turku, 20500 Turku, Finland

Abstract
The prediction of interactions between novel drugs and biological targets is a vital step in the early stage of the drug discovery pipeline. Many deep learning approaches have been proposed over the last decade, with a substantial fraction of them sharing the same underlying two-branch architecture. Their distinction is limited to the use of different types of feature representations and branches (multi-layer perceptrons, convolutional neural networks, graph neural networks and transformers). In contrast, the strategy used to combine the outputs (embeddings) of the branches has remained mostly the same. The same general architecture has also been used extensively in the area of recommender systems, where the choice of an aggregation strategy is still an open question. In this work, we investigate the effectiveness of three different embedding aggregation strategies in the area of drug–target interaction (DTI) prediction. We formally define these strategies and prove their universal approximator capabilities. We then present experiments that compare the different strategies on benchmark datasets from the area of DTI prediction, showcasing conditions under which specific strategies could be the obvious choice.

Keywords: Drug–target interaction prediction, Binding affinity prediction, Recommender systems, Deep learning

Introduction

Drug discovery is a challenging task that has consistently proven to be time-consuming and expensive [1]. As a result, pharmaceutical companies are turning towards computational methods capable of automating more stages of their drug discovery pipelines. One of the earliest stages involves the prediction of interactions between chemical compounds and biological targets, also known as drug–target interaction (DTI) prediction, replacing large-scale biological screening experiments with more efficient approaches. In this area, two groups of computational methods are distinguished: docking methods and machine learning approaches. Docking methods simulate the physical interaction of molecules and proteins in the 3D space accounting for the physical structure [2]. Machine learning approaches, on the other hand, predict drug–target interactions by learning from data, using databases that contain results of traditional high-throughput screening experiments [3].

Table 1 Two-branch methods

Method	Compound representation	Compound branch	Protein representation	Protein branch
DeepConv-DTI [4]	Morgan	MLP	Amino acid sequence	CNN
DeepDTA [5]	SMILES	CNN	Amino acid sequence	CNN
Shin et al. [6]	SMILES	Transformer	Fasta	CNN
MDeePred [7]	$ECFP_4$	MLP	Multi-channel protein features	CNN
Chen et al. [8]	Molecular graph	GNN	Amino acid sequence	BERT
Torng et al. [9]	Molecular graph	GCNN	Protein graph	GCNN
SSnet [10]	Fingerprint	MLP	Backbone curvature, torsion	CNN
Tsubaki et al. [11]	Molecular graph	GNN	Amino acid sequence	CNN
Kang et al. [12]	SMILES	BERT	Amino acid sequence	BERT
Nguyen et al. [13]	Molecular graph	GCN, GAT, GIN, GAT-GCN	Amino acid sequence	CNN
Mingjian et al. [14]	Molecular graph	GNN	Protein graph	GNN
DeepPurpose [15]	Various encodings	MLP, CNN, CNN–RNN...	Various encodings	MLP, CNN, CNN–RNN...

Recently, deep learning methods have gained a lot of interest in DTI prediction. In these methods, explicit features for the compounds and proteins are available beforehand and passed to a deep neural network. Even though many approaches exist, we will focus on a quite popular sub-category of two-branch deep neural networks. As shown in Fig. 1, these architectures consist of two neural network components that encode the compound and protein features, respectively. The two embedding vectors generated by the branches are then aggregated to obtain the final prediction. A non-exhaustive list of recent approaches that consider an architecture of that kind can be found in Table 1.

DTI prediction is closely related to recommender systems based on collaborative filtering, such as the well-known Netflix challenge [16]. In both cases, a standard dataset takes the form of triplets: $\{compound, protein, activity\}$ or $\{user, item, interaction\}$. These triplets can be arranged in a sparse matrix, and in its simplest form the prediction task is matrix completion. However, when explicit feature representations are used, three additional prediction settings become feasible, in addition to matrix completion [17]: making predictions for new compounds, new proteins or new protein-compound combinations. In DTI prediction, every method utilizes the explicit features that are available for the compounds and proteins. In contrast, similar methods in recommender systems often do not use explicit features, but create one-hot dummy vectors (implicit features) to characterize the users and items [18, 19].

DTI prediction and collaborative filtering methods also consider different aggregation strategies for the drug–target or user-item embeddings. Every DTI prediction method in Table 1 concatenates the two embeddings and then passes the resulting vector to a multi-layer perceptron (MLP strategy, see Fig. 1). Conversely, the area of collaborative filtering has a more diverse landscape, with various aggregation strategies that are regularly used,

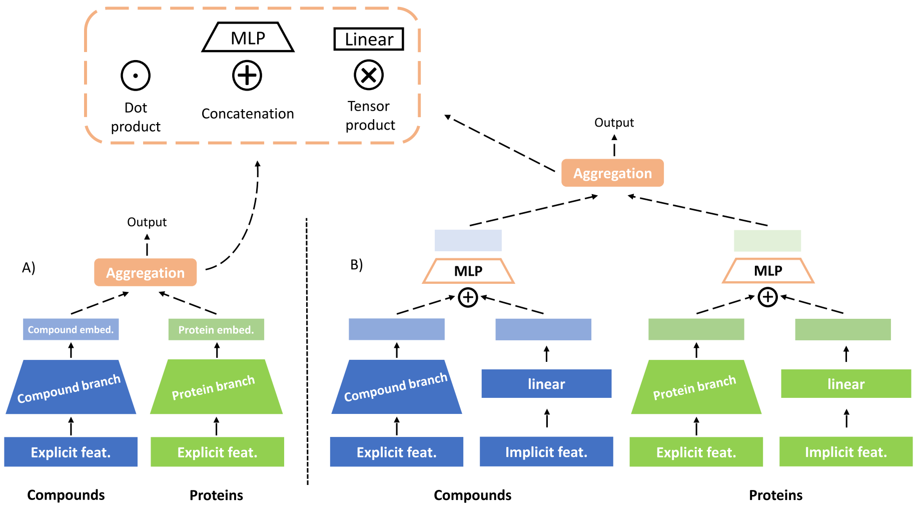


Fig. 1 A summary of the two main versions of the architecture we used. First, the two-branch architecture (**A**) encodes the explicit feature representations that are available for the compounds and proteins. The resulting embeddings can be aggregated with one of the three strategies (MLP, dot product, tensor product). The second version of the architecture combines both the implicit and explicit information available for the compounds and proteins (**B**). The explicit features can be encoded in the exact same manner as shown in (**A**), while the one-hot encoded dummy vectors for the compounds and proteins can be transformed into dense representations using a single fully-connected layer. The intermediate embeddings from the explicit and implicit features are aggregated using the MLP strategy. This output of the MLPs is the compound and protein embeddings that can then be aggregated with any of the three available strategies

see e.g. [18, 20–26]. Especially the dot product has been extensively used in that area, and papers benchmarking the dot product versus the MLP have been published [27, 28]. Currently, a similar investigation is missing from the area of DTI prediction.

In this work, we will discuss the behavior of different embedding aggregation strategies in DTI prediction. To this end, we will analyze the above-mentioned MLP and dot product strategies, as well as a third strategy, the tensor product, which used to be popular in the era of kernel methods [29–31]. First, we will present theoretical results highlighting the universal approximation properties of all three strategies, departing from well-known mathematical building blocks. Subsequently, we will present benchmarking results of the three strategies on combinations of DTI datasets, branch types and prediction settings. Furthermore, we will investigate the effect of adding implicit feature representations, and we will interpret the learned embeddings. As a result, the main goal of this paper is to compare the aggregation strategies in detail and not to present a winning architecture that achieves state-of-the-art performance.

Methods

Aggregation strategies

We first describe the details of the considered aggregation strategies. As shown in Fig. 1, we compare three strategies: the dot product, the multi-layer perceptron and the tensor product.

1. *Dot product* In the dot product strategy, the dot product of the two embeddings is directly computed and used as the prediction of the model. This strategy requires

both embeddings to have the same size, a restriction that we will further investigate in a later section.

2. *Multi-layer perceptron* The MLP strategy concatenates the embeddings and then passes the resulting vector as input to a multi-layer perceptron, which, in turn, terminates at a single output node. As stated before, the use of an MLP increases the capacity of the overall model compared to the simple dot product operation, but, perhaps, also introduces an unwanted overhead.
3. *Tensor product* The tensor product strategy first computes the tensor product of the two embeddings and then uses a single fully-connected layer that terminates at an output node. In fact, this aggregation strategy has never been suggested as an embedding aggregation strategy for deep neural networks, but it has been extensively used in kernel methods [29–31].

For reproducibility reasons, and to describe the universal approximation capabilities of the three aggregation strategies, we present formal definitions of the three strategies. Let $\mathcal{X}$ and $\mathcal{T}$ be two Euclidean spaces for compounds $\vec{x}$ and proteins $\vec{t}$, respectively. We formally define the problem of DTI prediction as that of estimating functions $f : \mathcal{X} \times \mathcal{T} \to \mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}$ in case of regression problems. Let us consider hypothesis spaces

$$\begin{aligned}\mathcal{H}_{\mathcal{X}} &= \left\{\vec{g}_{\vec{\theta}_{\mathcal{X}}} : \mathcal{X} \to \mathbb{R}^{D_1} \mid D_1 \in \mathbb{N}\right\}\\ \mathcal{H}_{\mathcal{T}} &= \left\{\vec{h}_{\vec{\theta}_{\mathcal{T}}} : \mathcal{T} \to \mathbb{R}^{D_2} \mid D_2 \in \mathbb{N}\right\}\end{aligned}$$

for learning compound embeddings from $\mathcal{X}$ to $\mathbb{R}^{D_1}$ with dimensionality D_1, and protein embeddings from $\mathcal{T}$ to $\mathbb{R}^{D_2}$ with dimensionality D_2. $\vec{\theta}_{\mathcal{X}}$ and $\vec{\theta}_{\mathcal{T}}$ denote the parameterizations of the two types of functions. Moreover, let us consider the space $C(\mathcal{X} \times \mathcal{T})$ of all continuous real-valued functions $f : \mathcal{X} \times \mathcal{T} \to \mathbb{R}$. A subspace $\mathcal{H}_{\mathrm{DP}}$ of $C(\mathcal{X} \times \mathcal{T})$ corresponds to the dot product strategy of the two-branch architecture, *i.e.*, functions of the form

$$f(\vec{x}, \vec{t}) = \sum_{d=1}^{D} g_d(\vec{x})\, h_d(\vec{t}),$$

with $g_{\vec{\theta}_{\mathcal{X}}} = (g_1, \ldots, g_D) \in \mathcal{H}_{\mathcal{X}}$, $h_{\vec{\theta}_{\mathcal{T}}} = (h_1, \ldots, h_D) \in \mathcal{H}_{\mathcal{T}}$, and D the common dimensionality of the two embeddings.

A second subspace $\mathcal{H}_{\mathrm{MLP}}$ corresponds to the MLP strategy of the two-branch architecture, in which the MLP is comprised of one hidden layer of size $L \in \mathbb{N}$, *i.e.*, functions of the form

$$f(\vec{x}, \vec{t}) = \vec{C}^{(3)}(\sigma \circ (\vec{B}^{(3)}[\vec{g}(\vec{x}), \vec{h}(\vec{t})] + \vec{b}^{(3)})),$$

where $\vec{C}^{(3)} \in \mathbb{R}^{L \times (D_1 + D_2)}$, $\vec{B}^{(3)} \in \mathbb{R}^{1 \times L}$, $\vec{b}^{(3)} \in \mathbb{R}^{D_1 + D_2}$, and $[\vec{g}, \vec{h}]$ denotes vector concatenation of the D_1-dimensional vector $\vec{g}$ and the D_2-dimensional vector $\vec{h}$. $\sigma \circ$ represents an elementwise nonlinear transformation.

We introduce a third and final subspace $\mathcal{H}_{\mathrm{TP}}$ that defines the tensor product strategy of the two-branch architecture, in which compound and protein embeddings are

Table 2 Dataset statistics

Dataset	#Drugs	#Targets	#Interactions
DAVIS	68	379	30,056
KIBA	2068	229	118,254

aggregated by means of a tensor product, followed by a linear layer with a single output neuron. The resulting functions are of the form

$$f(\vec{x},\vec{t}) = \sum_{k=1}^{D_1}\sum_{l=1}^{D_2} w_{kl}\, g_k(\vec{x})\, h_l(\vec{t}), \tag{1}$$

where $\vec{g_{\theta_\mathcal{X}}} = (g_1,\ldots,g_{D_1}) \in \mathcal{H}_\mathcal{X}$, $\vec{h_{\theta_\mathcal{T}}} = (h_1,\ldots,h_{D_2}) \in \mathcal{H}_\mathcal{T}$, and $\vec{W} \in \mathbb{R}^{D_1\times D_2}$.

Datasets

The proposed variants of the two-branch neural network architecture were evaluated on two benchmarks, Davis [32] and KIBA [33], because of their use as benchmarks in many of the studies presented in Table 1 and their varying numbers of compounds, proteins, and recorded affinities (see Table 2). The models were trained on the regression task, which aims to predict the affinity scores for compound-target pairs.

Prediction settings

Collaborative filtering methods usually predict missing interactions between collections of users and items that have been observed during training (random-split), something that does not require any explicit feature representations. In contrast, the availability of such features in the typical DTI prediction task makes three additional prediction settings feasible [17]. The model could be expected to generate predictions for novel drugs (cold-drug) or novel targets (cold-target) that have not been observed during training. A fourth option that combines the strategies of the previous two is concerned with the prediction for pairs of novel drugs and targets that have not been observed during training. In the collaborative filtering area, these types of prediction settings are used less frequently but also witness a growing interest (cold start collaborative filtering). In this paper, we run experiments for the first three prediction settings: random-split, cold-drug and cold-target. The fourth prediction setting (combination of cold-drug and cold-target) was excluded from our analysis, as it is not implemented in the DeepPurpose library, which is used as the building block for protein and compound branches—see next paragraph. For each prediction setting, every dataset is split into training, validation and test sets (70–10–20%). However, the way the data is separated differs. For the cold-target setting, 70% of the targets only appear in the training dataset, 10% only appear in the validation datasets, and the remaining 20% only appear in the test set. In the cold-drug setting, the same ratios are used to split the drugs. Finally, for the random-split, the ratios are used to split the $\{compound, protein, activity\}$ triplets in the dataset.

Benchmarking experiments

For the implementation of the two-branch architectures, the DeepPurpose library [15] was chosen as the starting point. A forked version of the repository, which has been heavily modified is available online.[1]

Since multiple branch pairings were possible, we included three combinations, reflecting varying degrees of descriptor and branch baseline complexity. These combinations included:

- An MLP compound branch on Morgan fingerprints [34] and an MLP protein branch on amino acid composition descriptors [35].
- A 1D Convolutional Neural Network (CNN) [36] compound branch on SMILES strings and a CNN protein branch on amino acid sequences.
- A Message-Passing Neural Network (MPNN) [37] compound branch on molecular graphs and a CNN protein branch on amino acid sequences.

Implicit feature representations

All the experiments mentioned above utilize different forms of explicit feature representations for both compounds and proteins, but not the structure of the interaction matrix. To better investigate the quality of these sources of information, we conducted additional experiments with the following differences:

- An MLP branch for the compounds and proteins that, instead of using explicit features, utilizes one-hot encoded dummy vectors. Since no generalization to new compounds or new proteins is possible when using this type of feature, we only focus on the random-split setting. When the dot product is used as the aggregation strategy the resulting architecture is a close analogue to traditional matrix factorization methods.
- A two-branch architecture where each branch is comprised of an internal two-branch model (Fig. 1B). The internal model is designed to utilize the explicit and implicit features of the compounds/proteins, something that could potentially lead to improved performance. The MLP strategy is always used for the aggregation of the internal embeddings, while all three strategies of interest are available for the external embeddings.

Hyperparameter optimization

For the comparison of the three aggregation strategies across two DTI prediction datasets and the three prediction settings, we utilized random search as the hyperparameter optimization method of choice. For every optimization round, a budget of 100 configurations was allocated, with each experiment training for up to 100 epochs (early stopping on the validation loss was also used).

[1] https://github.com/diliadis/DeepPurpose.

The hyperparameter ranges of every experiment had to be adapted based on the aggregation strategy and branch architectures used. The full details can be found in the Additional file 1: Appendix. Every model was trained on a single GPU (either NVIDIA Ampere A100 or NVIDIA Volta V100), and all the results were logged using the Weights and Biases platform [38].

Metrics

To guarantee a consistent comparison across all our experiments, we adopted the same regression metrics as used in the majority of the work presented in Table 1. These include metrics like:

(i) Mean squared error (MSE): Measures the differences between the predicted values and the real values. Assuming n drug–target pairs, the MSE is calculated as the average of the squared differences between the predicted affinity scores $\hat{y}$ and the true affinity scores y. The goal is to minimize the MSE score as this means that the predictions are close to the true values:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2. \tag{2}$$

(ii) R-squared (R^2): Also known as the coefficient of determination, R^2 is a statistical measure that represents the proportion of the variance of the dependent variable that is explained by the independent variables in a regression model. Unlike MSE, which is a measure of the model's absolute error, R^2 is a relative measure of how well the regression predictions approximate the true values. An R^2 of 1 indicates that the regression predictions perfectly fit the data. In the context of drug–target interaction prediction, it quantifies how well the variations in the predicted affinity scores $\hat{y}$ explain the variation in the true affinity scores y. The formula for R^2 is given by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{3}$$

where $\bar{y}$ is the mean of the true affinity scores. A higher R^2 score indicates a better model fit.

(iii) Concordance index (CI): CI is the probability that the predicted affinity scores of two randomly chosen drug–target pairs are in the correct order:

$$\mathrm{CI} = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{I}(y_i > y_j)}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{I}(y_i > y_j)\mathrm{I}(\hat{y}_i > \hat{y}_j), \tag{4}$$

where I is the indicator function, taking value 1 if its argument is true, 0 otherwise. A higher value of the concordance index indicates a better model fit.

Results

Universal approximation

Using existing mathematical results from [39, 40] as building blocks, one can easily show that all three aggregation strategies are universal approximators. We provide further details and formal derivations in Additional file 1: Appendix, but summarize the main insights here. Broadly speaking, universal approximation theorems imply that neural networks can represent a wide variety of interesting functions when given appropriate weights. On the other hand, they typically do not provide a construction for the weights, but merely state that such a construction is possible.

In the setting of DTI prediction, universal approximation can only be guaranteed if the protein branch, compound branch and the aggregation strategy are universal approximators. For the formal results presented in the Additional file 1: Appendix, we assume that the protein and compound branches are universal approximators, and we show that this is sufficient to prove universal approximation of the three aggregation strategies. For simplicity, we also assume that protein and compound feature vectors can be represented in Euclidean spaces, with multi-layer perceptrons as protein and compound branches. Similar universal approximation theorems could be easily derived for different activation functions [41, 42], non-Euclidean spaces [43], and other types of neural network architectures, such as deep convolutional neural networks [44].

Benchmarking experiments

In this section, we present extensive comparisons of the three embedding aggregation strategies on popular benchmark datasets from the field of DTI prediction. The experiments span two different DTI prediction datasets, three prediction settings (random, cold-drug, cold-target), as well as three different combinations of input feature representations and compound-protein branch architectures. Table 3 offers a quick summary of the results that have been obtained.

In the majority of cases and for both recorded metrics, we see that the dot product and tensor product strategies can be seen as competitive alternatives to the MLP. In many cases, they achieve superior performance. At the same time, none of the three strategies can be highlighted as the strategy of choice purely based on the final performance, as none of them consistently outperforms the others. These results confirm our theoretical findings presented in the Additional file 1: Appendix, as all three strategies can approximate any target function. Interestingly, specific combinations of dataset, prediction setting and branch pair exist, in which the three strategies result in unexpected performance differences. More specifically, when we used an MPNN as the compound branch and a CNN as the protein branch, the dot product and tensor product strategies clearly failed to reach the performance of the MLP strategy in both randomly split datasets. A more detailed investigation of the reasons behind this result as well as a potential remedy for the dot product and tensor product strategies are presented in a later subsection.

An important characteristic of any model that can influence many practical aspects of the training process is its overall capacity. Even though capacity measures of neural networks exist (e.g. Vapnik–Chervonenkis dimension [45]), they are primarily dealing with simple MLP architectures. Since the two-branch architecture we utilize can be equipped with more complex branches (CNNs, MPNNs, etc.) we decided to simplify things and instead use the total number of trainable parameters as the measure of model capacity. Since a smaller network with fewer parameters can result in reduced memory and lower computational requirements, a smaller model that can still achieve a similar performance compared with a larger counterpart is highly desirable. Our initial hypothesis

Table 3 Performance results obtained for every possible combination of two datasets (KIBA, DAVIS), three prediction settings (random, cold-drug, cold-target), three embedding aggregation strategies and three compound-target branch pairs (MLP–MLP, CNN–CNN, MPNN–CNN)

Dataset	Pred setting	Aggregation	MLP–MLP				CNN–CNN				MPNN–CNN			
			MSE	R^2	CI	Params	MSE	R^2	CI	Params	MSE	R^2	CI	Params
DAVIS	Random	Dot prod.	0.2656	0.6733	0.8702	2.1	0.2622	0.6776	0.8727	0.6	0.5561	0.3161	0.7687	0.2
		Tensor prod.	0.2828	0.6522	0.8622	6.7	0.2708	0.6669	0.8699	0.4	0.5537	0.3190	0.7720	0.4
		MLP	0.2722	0.6652	0.8759	2.4	0.2489	0.6938	0.8791	0.8	0.2647	0.6745	0.8722	1.9
	Cold	Dot prod.	0.6189	− 0.0426	0.7304	6.3	0.6593	− 0.1107	0.6657	0.1	0.5112	0.1387	0.7287	0.2
	Drug	Tensor prod.	0.6828	− 0.1502	0.6929	1.4	0.6807	− 0.1467	0.6142	0.3	0.5125	0.1366	0.7326	0.4
		MLP	0.6764	− 0.1394	0.6949	4.9	0.6464	− 0.0889	0.6484	0.5	0.5123	0.1370	0.7385	3.3
	Cold	Dot prod.	0.5895	0.3369	0.7974	1.1	0.4839	0.4556	0.8137	0.6	0.6783	0.2369	0.7294	0.1
	Target	Tensor prod.	0.5750	0.3532	0.8077	1.6	0.4803	0.4596	0.8069	0.5	0.6653	0.2515	0.7471	0.1
		MLP	0.5966	0.3288	0.8023	2.0	0.4905	0.4482	0.8092	1.7	0.5626	0.3671	0.8014	0.9
KIBA	Random	Dot prod.	0.1994	0.7187	0.8379	2.1	0.2169	0.6940	0.8329	0.6	0.5490	0.2456	0.7154	0.2
		Tensor prod.	0.2092	0.7050	0.8408	4.4	0.2222	0.6865	0.8281	0.4	0.5534	0.2194	0.6996	0.3
		MLP	0.2093	0.7048	0.8496	2.6	0.1953	0.7245	0.8565	1.2	0.2544	0.6412	0.8190	2.3
	Cold Drug	Dot prod.	0.4167	0.4498	0.7510	9.9	0.4730	0.3756	0.7274	0.5	0.5930	0.2171	0.6985	0.4
		Tensor prod.	0.4345	0.4264	0.7460	7.8	0.4779	0.3691	0.7310	0.4	0.5965	0.2124	0.6978	0.3
		MLP	0.4258	0.4378	0.7538	3.8	0.4802	0.3661	0.7348	1.1	0.5172	0.3172	0.7211	2.6
	Cold Target	Dot prod.	0.4197	0.3666	0.7069	1.3	0.3839	0.4206	0.7281	0.7	0.6328	0.0450	0.6023	0.3
		Tensor prod.	0.4126	0.3773	0.7142	1.7	0.3904	0.4108	0.7192	0.5	0.6326	0.0453	0.6040	0.4
		MLP	0.3856	0.4181	0.7203	1.8	0.3795	0.4272	0.7308	0.5	0.4172	0.3704	0.7168	2.6

For every combination, the average MSE, R^2, CI and number of trainable parameters of the top-5 performing (lowest overall loss) configurations are reported was that the overhead of the MLP strategy introduced by the fully-connected layers after the concatenation of the compound and protein embeddings would result in larger models. Based on the results shown in Table 3 the aforementioned competitiveness of the dot product strategy is usually achieved by small models. By accounting for this extra information, we can more confidently suggest the dot product strategy as a replacement for the MLP-based architecture.

Revisiting the close connection with recommender systems, a highly-cited publication by He et al. [18] first presented the dot product strategy as the simplest neural network approximation of matrix factorization. He et al. [18] then suggested the MLP strategy as a more powerful approach with the capacity to model more complex relationships between the items and users. This experimentally-backed strategy was then adopted by a series of subsequent publications [20–23] in the area of recommender systems, while proposals with the dot product strategy continued to be considered [24–26].

The superiority of the MLP aggregation strategy in the area of collaborative filtering was questioned by several subsequent publications. Rendle et al. [27] showed that, with careful hyperparameter selection, the dot product strategy could outperform the MLP strategy. They also pointed out that an MLP cannot trivially approximate the seemingly basic dot product operation. Xu et al. [28] offered a more rigorous comparison by investigating the limiting expressivity of each strategy, the convergence under the practical gradient descent algorithm, and the generalization potential. The two aforementioned publications approach the comparison of strategies exclusively in the area of recommender systems using benchmark datasets that are missing any explicit features for the users or items. In our investigation, which includes explicit feature representation and multiple generalization settings, we formulate similar conclusions as [27] and [28].

Implicit feature representations

Furthermore, Table 4 contains the results for two additional neural network configurations: two-branch neural networks that only use implicit features, and two-branch neural networks that use implicit and explicit features. So, in combination with the results of Table 3, which summarized the results for two-branch neural networks that only included explicit features, we compare here three types of two-branch neural networks. Overall, the initial setup with only explicit features gives the best results, but the differences between the three variants is small. The negligible differences let us conclude that adding implicit features does not have benefits for the datasets and models that we considered. However, the neural network that only uses implicit features still yields a satisfactory performance, so a clear structure must be present in the interaction matrices of the two datasets.

In the last decade, matrix factorization methods have been extensively used to exploit the structure in the interaction matrix by decomposing it into two small matrices that contain the implicit features [46]. Formally speaking, the structure of the interaction matrix can be summarized using the singular values of that matrix. If most singular values differ from zero, the interaction matrix has a high rank, so approximating it as a product of two smaller matrices will lead to little predictive power and meaningless implicit features. Conversely, if most singular values are equal to zero, the interaction matrix has a low rank, and low-rank matrix approximation via matrix factorization will result in good predictive performance and meaningful implicit features.

Matrix factorization methods share major similarities with the two-branch architectures we consider in this work. The simplest version of the two-branch architecture, which uses only the implicit features and aggregates the embeddings via the dot product strategy, can be seen as a way of performing matrix factorization [18, 28, 47–49]. For explicit feature representations and other aggregation strategies, the link with matrix factorization is less obvious, and the models become more difficult to analyze in a formal way. However, we believe that the structure of the interaction matrix is also exploited by the models in that case, because low-dimensional embeddings of proteins and compounds are constructed. To our opinion, that's the main reason why adding implicit features does not lead to performance gains in our experiments.

Let us remark that matrix factorization methods have been extensively used for DTI prediction. Early work by Cobanoglu et al. [50] used probabilistic matrix factorization combined with active learning without relying on compound or protein similarities. Ezzat et al. [51] proposed a graph regularized matrix factorization method (and a weighted variant) to perform manifold learning and improve the performance in the cold-drug and cold-target prediction settings. More recent work by Mazzone et al. [52] used the NXTfusion [53] library that extends traditional matrix factorization methods in a non-linear fashion by inferring over an arbitrary number of data matrices, which are built as an entity relation graph. The data fusion step was performed by training a multi-task neural network that includes side information.

The concept of combining implicit and explicit information to improve the performance of DTI prediction tasks has also been incorporated into various kernel-based methods. Gonen [29] proposed a Bayesian formulation that combines kenrel-based non-linear dimensionality reduction, matrix factorization and binary classification to predict interactions using only the chemical similarity between compounds and genomic similarity between proteins. Another kernel method, called NMTF-DTI [54], used a non-negative tri-factorization technique based on Laplacian regularization and multiple similarity matrices for drugs and targets. A third approach defined the Gaussian interaction profile kernel [55] to capture topological information from the drug–target interaction network, and a variation of that method improved the predictive performance even further by incorporating extra sources of chemical and genomic information with additional kernels. These results do not agree with our findings, but the datasets used in most kernel-based papers were significantly smaller than those we analyzed.

Similar to the two-branch neural networks we discuss in this work, kernel methods enable the projection of the compounds and proteins into a shared space from which the predictions are generated. However, in two-branch neural networks, the branches "learn" the compound and protein embeddings, while kernel-based methods pre-compute these embeddings by specifying a specific kernel. In general, one can assume that learning the embeddings is better than defining them based on a kernel, especially when datasets are large. That's probably the reason why deep learning methods have won the battle against kernel methods in DTI prediction. Furthermore, it is worth mentioning that the use of multiple kernels for drugs and targets with the goal of improving performance [56, 57] shows many similarities with multi-modal neural network architectures

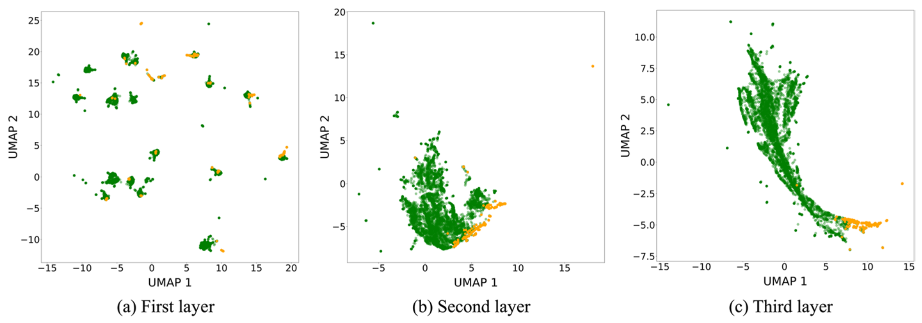


Fig. 2 UMAP visualization of the test compound-protein embeddings extracted from the fully-connected layers of the best MLP configuration trained on the DAVIS dataset. The configuration uses CNN compound and protein branches. The points are colored using the binarized affinities (orange for active and green for inactive). From the figure, we observe that starting from the first layer (after the concatenation of the compound and protein branch embeddings) to the third, the separation of the two classes becomes clearer

that utilize multiple branches (one per feature representation) before the aggregation step [58–61]. Both concepts essentially try to boost performance by including different types of information from the same entity.

Embedding visualizations

To better understand the similarities and differences between the MLP and dot product, we experimented with various visualizations and included the most interesting examples in this work. From this point onwards, our analysis skips the tensor product, as it does not provide any performance gains or distinct characteristics compared with the MLP and dot product strategies. In their simplified form, one of the key conceptual distinctions among these strategies lies in the location of the learning process. In the dot product strategy, this is exclusively done in the two branches since the aggregation strategy is just the dot product operation. In contrast, the MLP strategy shares the learning between the two branches and the fully-connected layers after the concatenation of the embeddings. A basic visualization of the compound-protein embeddings obtained from the fully-connected layers (Fig. 2) of a well-performing MLP strategy shows an improving separation of the active and inactive compound-protein pairs the closer we get to the output node.

To investigate the quality of the compound embeddings, we experimented with another type of visualization in Fig. 3. The four subplots visualize the affinity scores of four proteins (with the highest number of recorded affinities in the KIBA dataset) after the Uniform Manifold Approximation and Projection (UMAP) [62] is applied on the compound embeddings.

Even though the affinity differences observed in compound clusters between the four proteins are quite interesting, we focus on the relative comparison of the learned clusters. Based on the aforementioned discussion about the location of the learning process, the well-defined clusters of the compound embeddings from the dot product strategy are largely expected, as the two branches are exclusively responsible for learning to predict affinities. At the same time and, somewhat surprisingly, the visual inspection of the compound embeddings obtained from the MLP strategy shows similar embedding

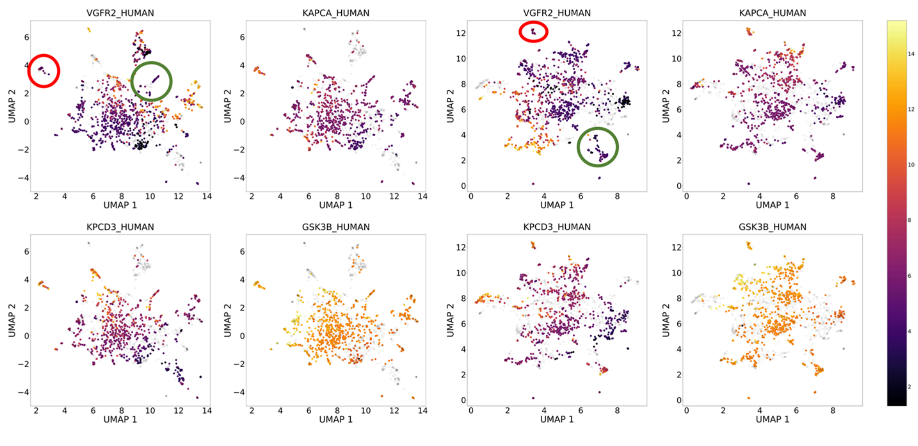


Fig. 3 UMAP visualization of the compound embeddings extracted from the last fully-connected layer of the compound branch (CNN) of the best MLP (left) and dot product (right) configurations trained on the KIBA dataset. The points are colored based on the affinities of the four proteins in the training set with the highest number of recorded entries. Comparing the two strategies, we observe that both create clusters of similar quality

quality as the dot product, even though the fully-connected layers after the embedding concatenation share the learning responsibility.

Oversmoothing effect and its importance when selecting an embedding aggregation strategy

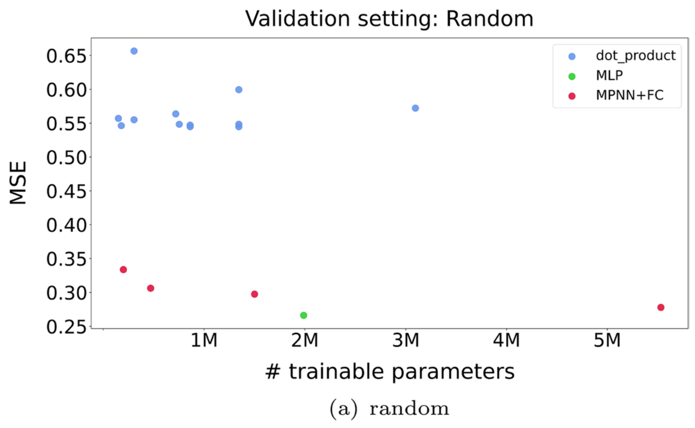
For specific combinations of dataset, prediction setting and branch pair, Table 3 includes dot product and tensor product examples that are clearly outperformed by the MLP aggregation strategy. These cases include the use of the MPNN compound branch and the CNN protein branch on the randomly split version of the two datasets. As the CNN protein branch had been successfully used in the CNN–CNN combination, we focused our efforts on the MPNN branch as the defective model. In an effort to increase its capacity, we first tested configurations with deeper MPNNs, something that ultimately did not yield any improvements. Our hypothesis was that the over-smoothing effect that many graph neural networks suffer from [63] made any attempt to increase the capacity fail performance-wise. This is shown in the two plots of Fig. 4, where points from models that use the dot product strategy and varying sizes of the MPNN branch architecture fail to approach the performance the MLP strategy achieves. Assuming that the superior performance of the MLP-based architecture was a result of the fully-connected layers, we decided to test new dot-product-based configurations by appending fully-connected layers after a smaller MPNN model and before the embedding aggregation step. With this strategy, we managed to both increase the capacity of the compound branch and avoid the over-smoothing effect that larger MPNN models suffer from. Referring back to Fig. 4, these modified architectures colored in red show major improvements, as they became comparable performance-wise with the MLP strategy.

Oversmoothing is a well-recognized challenge in GNNs, and numerous research papers have been dedicated to address this issue [64–66]. The underlying message of this exploration is that the dot product has the inability to counterbalance weaknesses that

Table 4 Performance results obtained for testing combinations of implicit and explicit compound and protein feature representations

Dataset	Aggregation	MLP–MLP (implicit)				MLP–MLP (implicit+explicit)				CNN–CNN (implicit+explicit)			
		MSE	R^2	CI	Params (M)	MSE	R^2	CI	Params (M)	MSE	R^2	CI	Params (M)
DAVIS	Dot prod.	0.2804	0.6551	0.8653	5.2	0.2859	0.6484	0.8702	5.2	0.2690	0.6692	0.8752	2.3
	Tensor prod.	0.3008	0.6301	0.8654	9.6	0.3138	0.6141	0.8632	9.6	0.3394	0.5825	0.8574	1.3
	MLP	0.3104	0.6182	0.8585	8.3	0.3217	0.6043	0.8554	8.3	0.3135	0.6145	0.8580	2.0
KIBA	Dot prod.	0.2481	0.6500	0.8326	7.2	0.2277	0.6789	0.8365	7.2	0.2296	0.6761	0.8402	3.5
	Tensor prod.	0.2625	0.6297	0.8276	12.9	0.2555	0.6396	0.8273	12.9	0.2631	0.6290	0.8221	2.6
	MLP	0.2440	0.6559	0.8387	12.0	0.2590	0.6347	0.8285	12.0	0.2489	0.6490	0.8347	3.2

The first combination uses to MLP branches that process dummy one-hot encoded features for the compounds and proteins. The second and third combinations represent two-level two-branch architectures. The outer level is comprised of the classic two-branch architecture with embeddings that can be combined using the three strategies we investigate. The two outer branches are composed of two-branch models that utilize both explicit and implicit features. For every combination, the average MSE, R^2, CI, and number of trainable parameters of the top-5 performing (lowest overall loss) configurations are reported



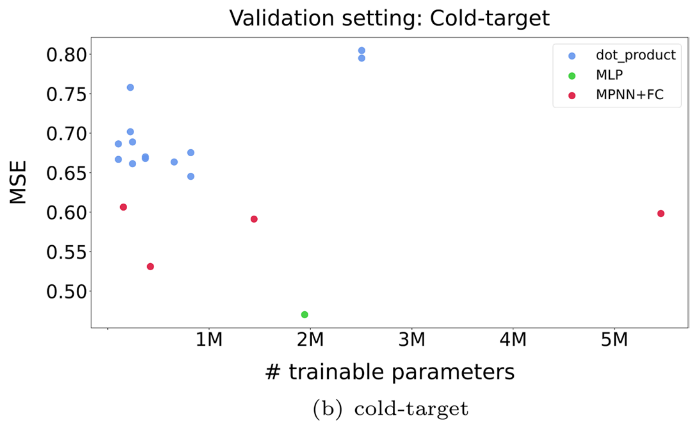


Fig. 4 Plot of the capacity of different models against their test MSE. All the points represent experiments with an MPNN as the compound branch and a CNN as the protein branch. The point colored green represents a well-performing experiment of the MLP strategy, while the blue points represent dot product strategies with similar branch configurations. The results illustrate the inability of the dot product to achieve comparable performance with the MLP strategy when one of the branches fails to utilize its input properly. Even after testing dot-product-based configurations with varying capacity (increasing the MPNN depth) exceeding that of the MLP benchmark, the performance does not improve. Our hypothesis is that the main reason for the underperforming experiments that use the dot product is the over-smoothing effect that graph neural networks suffer from. This assumption is supported by the significant improvements that a small modification attains (colored red). Instead of increasing the capacity of the MPNN directly, we instead append fully-connected layers immediately after it and before the embedding aggregation operation, thus increasing the overall capacity of the compound branch and bypassing the MPNN. The experiments colored red demonstrate that the dot product strategy can reach a comparable performance as the MLP strategy while keeping the overall capacity low

may arise from the branches, as it lacks the trainable parameters that the MLP strategy has. This problem can be tackled by increasing the capacity of the defective branch. However, cases still exist where this strategy is inadequate because of branch-specific weaknesses (e.g., over-smoothing in GNNs). In such cases, it is suggested to increase the capacity of the defective branch with fully-connected layers or replace the branch altogether.

Conclusion

In this work, we analyzed alternatives to the traditional embedding aggregation strategy that has dominated the two-branch architectures in the field of DTI prediction. We presented formal and experimental results which show that all three analyzed strategies can be used to aggregate embeddings. We identified conditions under which a particular type of aggregation strategy might outperform others, and we presented various visualizations. We believe that this work can be the first step in convincing the DTI prediction community to also focus on the embedding aggregation options. Even though this may not seem vital when aggregating only two embeddings, it can become an important choice in multi-modal architectures or when more than two embeddings have to be aggregated (e.g. drug–drug–protein interaction prediction).

With regard to future work, we intend to test attention-based embedding aggregation methods, which have become popular in other application domains. Furthermore, increasing the number of embeddings that are aggregated is also an interesting avenue we intend to explore. This approach is used when different representations are available for the same entity. For example, different feature representations for a given compound (Morgan fingerprint, 2D image, molecular graph) could be combined in different ways. Thus, the order and strategy used at every stage of aggregating embeddings is a complex task. Finally, the evaluation of the added value that pre-trained embeddings from different strategies can bring to transfer learning tasks is an interesting topic that we intend to investigate in future work.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-024-05684-y.

Additional file 1: A) Contains theorems that show how all three aggregation strategies we consider are universal approximators. **B)** Contains detailed information about the hyperparameter ranges of every experiment included in this work.

Author contributions

DI performed the research. DI, WW, TP and BB contributed to the manuscript, read and approved the final version.

Funding

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiele Intelligentie (AI) Vlaanderen" programme.

Availability of data and materials

The KIBA and DAVIS datasets are publicly available https://github.com/futianfan/DeepPurpose_Data. The scripts used to perform the experiments are also available online https://github.com/diliadis/DeepPurpose as a forked and modified repository of the original DeepPurpose project.

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Received: 25 September 2023 Accepted: 30 January 2024
Published online: 06 February 2024

References

1. Sinha S, Vohora D. Drug discovery and development: an overview. Pharm Med Transl Clin Res. 2018;19–32.
2. Pujadas G, Vaque M, Ardevol A, Blade C, Salvado MJ, Blay M, Fernandez-Larrea J, Arola L. Protein-ligand docking: a review of recent advances and future perspectives. Curr Pharm Anal. 2008;4(1):1–19. https://doi.org/10.2174/157341208783497597.

3. Zanni R, Gálvez-Llompart M, Gálvez J, García-Domenech R. QSAR multi-target in drug discovery: a review. Curr Comput Aided Drug Des. 2014;10(2):129–36. https://doi.org/10.2174/157340991002140708105124.
4. Lee I, Keum J, Nam H. DeepConv-DTI: prediction of drug–target interactions via deep learning with convolution on protein sequences. PLoS Comput Biol. 2019;15(6):1007129.
5. Öztürk H, Özgür A, Ozkirimli E. DeepDTA: deep drug–target binding affinity prediction. Bioinformatics. 2018;34:821–9.
6. Shin B, Park S, Kang K, Ho JC. Self-attention based molecule representation for predicting drug–target interaction. Proc Mach Learn Res (PMLR). 2019;106:1–18.
7. Rifaioglu AS, Atalay RC, Kahraman DC, Doğan T, Martin M, Atalay V. MDeePred: novel multi-channel protein featurization for deep learning-based binding affinity prediction in drug discovery. Bioinformatics. 2021;37(5):693–704.
8. Chen W, Chen G, Zhao L, Yu-Chian Chen C. Predicting drug–target interactions with deep-embedding learning of graphs and sequences. J Phys Chem. 2021;125:5642.
9. Torng W, Altman RB. Graph convolutional neural networks for predicting drug–target interactions. J Chem Inf Model. 2019.
10. Karki N, Verma N, Trozzi F, Tao P, Kraka E, Zoltowski B. SSnet: a deep learning approach for protein–ligand interaction prediction. Int J Mol Sci. 2021;22(3):1392.
11. Tsubaki M, Tomii K, Sese J. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. Bioinformatics. 2019;35(2):309–18.
12. Kang H, Goo S, Lee H, Chae J-W, Yun H-Y, Jung S. Fine-tuning of Bert model to accurately predict drug–target interactions. Pharmaceutics. 2022;14(8):1710.
13. Nguyen T, Le H, Quinn TP, Nguyen T, Le TD, Venkatesh S. GraphDTA: predicting drug–target binding affinity with graph neural networks. Bioinformatics. 2021;37(8):1140–7.
14. Jiang M, Li Z, Zhang S, Wang S, Wang X, Yuan Q, Wei Z. Drug–target affinity prediction using graph neural network and contact maps. RSC Adv. 2020;10(35):20701–12.
15. Huang K, Fu T, Glass LM, Zitnik M, Xiao C, Sun J. DeepPurpose: a deep learning library for drug–target interaction prediction. Bioinformatics. 2020;36(22–23):5545–7.
16. Bennett J, Lanning S. The Netflix Prize. In: Proceedings of KDD cup and workshop 2007. https://www.semanticscholar.org/paper/The-Netflix-Prize-Bennett-Lanning/31af4b8793e93fd35e89569ccd663ae8777f0072. Accessed 16 Feb 2023.
17. Pahikkala T, Airola A, Pietilä S, Shakyawar S, Szwajda A, Tang J, Aittokallio T. Toward more realistic drug–target interaction predictions. Brief Bioinform. 2015;16(2):325–37.
18. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS. Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web (WWW), 2017. pp. 173–182.
19. Wu Y, DuBois C, Zheng AX, Ester M. Collaborative denoising auto-encoders for top-n recommender systems. In: Proceedings of the ninth ACM international conference on web search and data mining (WSDM), 2016. pp. 153–162.
20. Chen W, Cai F, Chen H, Rijke MD, Chen H. Joint neural collaborative filtering for recommender systems. ACM Trans Inf Syst (TOIS). 2019;37(4):39.
21. Karolina Dziugaite G, Roy DM. Neural network matrix factorization. arXiv preprint arXiv:1511.06443 2015.
22. Liu Y, Wang S, Khan MS, He J. A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering. Big Data Min Anal. 2018;1(3):211–21.
23. Nguyen DM, Tsiligianni E, Deligiannis N. Extendable neural matrix completion. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), 2018. pp. 6328–6332.
24. Wang T, Brovman YM, Madhvanath S. Personalized embedding-based e-commerce recommendations at eBay. arXiv preprint arXiv:2102.06156 2021.
25. Yang J, Yi X, Zhiyuan Cheng D, Hong L, Li Y, Xiaoming Wang S, Xu T, Chi EH. Mixed negative sampling for learning two-tower neural networks in recommendations. In: Companion proceedings of the web conference 2020 (TheWebConf), 2020. pp. 441–447.
26. Yi X, Yang J, Hong L, Cheng DZ, Heldt L, Kumthekar A, Zhao Z, Wei L, Chi E. Sampling-bias-corrected neural modeling for large corpus item recommendations. In: Proceedings of the 13th ACM conference on recommender systems (RecSys), 2019. pp. 269–277.
27. Rendle S, Krichene W, Zhang L, Anderson J. Neural collaborative filtering vs. matrix factorization revisited. In: Proceedings of the 14th ACM conference on recommender systems (RecSys), 2020. pp. 240–248.
28. Xu D, Ruan C, Korpeoglu E, Kumar S, Achan K. Rethinking neural vs. matrix-factorization collaborative filtering: the theoretical perspectives. In: International conference on machine learning (ICML). PMLR; 2021. pp. 11514–11524.
29. Gönen M. Predicting drug–target interactions from chemical and genomic kernels using Bayesian matrix factorization. Bioinformatics. 2012;28(18):2304–10.
30. Stock M, Pahikkala T, Airola A, De Baets B, Waegeman W. A comparative study of pairwise learning methods based on kernel ridge regression. Neural Comput. 2018;30(8):2245–83.
31. Vert J-P, Qiu J, Noble WS. A new pairwise kernel for biological network inference with support vector machines. In: BMC bioinformatics, vol. 8. Springer; 2007. pp. 1–10.
32. Davis MI, Hunt JP, Herrgard S, Ciceri P, Wodicka LM, Pallares G, Hocker M, Treiber DK, Zarrinkar PP. Comprehensive analysis of kinase inhibitor selectivity. Nat Biotechnol. 2011;29(11):1046–51.
33. Tang J, Szwajda A, Shakyawar S, Xu T, Hintsanen P, Wennerberg K, Aittokallio T. Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. J Chem Inf Model. 2014;54(3):735–43.
34. Rogers D, Hahn M. Extended-connectivity fingerprints. J Chem Inf Model. 2010;50(5):742–54.
35. Reczko M, Bohr H. The def data base of sequence based protein fold class predictions. Nucleic Acids Res (NAR). 1994;22(17):3616.
36. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Commun ACM. 2017;60(6):84–90.

37. Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: International conference on machine learning (ICML), Proceedings of machine learning research (PMLR). 2017. pp. 1263–1272.
38. Biewald L. Experiment tracking with weights and biases. Software available from wandb.com 2020. https://www.wandb.com/.
39. Allenby PD, Labuschagne CCA. On the uniform density of c(x) ⊗ c(y) in c(x×y). Indag Math. 2009;20(1):19–22. https://doi.org/10.1016/S0019-3577(09)00015-9.
40. Cybenko G. Approximation by superpositions of a sigmoidal function. Math Control Signals Syst. 1989;2(4):303–14.
41. Leshno M, Lin VY, Pinkus A, Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw. 1993;6(6):861–7. https://doi.org/10.1016/S0893-6080(05)80131-5.
42. Pinkus A. Approximation theory of the MLP model in neural networks. Acta Numer. 1999;8:143–95.
43. Brüel Gabrielsson R. Universal function approximation on graphs. In: Larochelle H, Ranzato M, Hadsell R, Balcan MF, Lin H, editors. Advances in neural information processing systems (NIPS), vol. 33. Curran Associates, Inc; 2020. pp. 19762–19772. https://proceedings.neurips.cc/paper_files/paper/2020/file/e4acb4c86de9d2d9a41364f93951028d-Paper.pdf.
44. Zhou D-X. Universality of deep convolutional neural networks. Appl Comput Harmon Anal (ACHA). 2020;48(2):787–94.
45. Vapnik VN, Chervonenkis AY. On the uniform convergence of relative frequencies of events to their probabilities. In: Measures of complexity: festschrift for Alexey Chervonenkis. Springer, Cham; 2015. pp. 11–30.
46. Waegeman W, Dembczyński K, Hüllermeier E. Multi-target prediction: a unifying view on problems and methods. Data Min Knowl Discov (KDD). 2019;33(2):293–324.
47. Chen X, Zhang Y, Ai Q, Xu H, Yan J, Qin Z. Personalized key frame recommendation. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, 2017. pp. 315–324.
48. Wang X, He X, Nie L, Chua T-S. Item silk road: recommending items from information domains to social users. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, 2017. pp. 185–194.
49. Zhang S, Yao L, Sun A, Tay Y. Deep learning based recommender system: a survey and new perspectives. ACM Comput Surv (CSUR). 2019;52(1):1–38.
50. Cobanoglu MC, Liu C, Hu F, Oltvai ZN, Bahar I. Predicting drug–target interactions using probabilistic matrix factorization. J Chem Inf Model. 2013;53(12):3399–409.
51. Ezzat A, Zhao P, Wu M, Li X-L, Kwoh C-K. Drug–target interaction prediction with graph regularized matrix factorization. IEEE/ACM Trans Comput Biol Bioinform (TCBB). 2016;14(3):646–56.
52. Mazzone E, Moreau Y, Fariselli P, Raimondi D. Nonlinear data fusion over entity–relation graphs for drug–target interaction prediction. Bioinformatics. 2023;348.
53. Raimondi D, Simm J, Arany A, Moreau Y. A novel method for data fusion over entity–relation graphs and its application to protein–protein interaction prediction. Bioinformatics. 2021;37(16):2275–81.
54. Jamali AA, Kusalik A, Wu F. NMTF-DTI: a nonnegative matrix tri-factorization approach with multiple kernel fusion for drug–target interaction prediction. IEEE/ACM Trans Comput Biol Bioinform (TCBB). 2021.
55. Van Laarhoven T, Nabuurs SB, Marchiori E. Gaussian interaction profile kernels for predicting drug–target interaction. Bioinformatics. 2011;27(21):3036–43.
56. Nascimento AC, Prudêncio RB, Costa IG. A multiple kernel learning algorithm for drug–target interaction prediction. BMC Bioinform. 2016;17:1–16.
57. Zheng X, Ding H, Mamitsuka H, Zhu S. Collaborative matrix factorization with multiple similarities for predicting drug–target interactions. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD), 2013;1025–1033.
58. Boezer M, Tavakol M, Sajadi Z. FastDTI: drug–target interaction prediction using multimodality and transformers. In: Proceedings of the northern lights deep learning workshop, vol. 4. 2023.
59. Ren Z-H, You Z-H, Zou Q, Yu C-Q, Ma Y-F, Guan Y-J, You H-R, Wang X-F, Pan J. DeepMPF: deep learning framework for predicting drug–target interactions based on multi-modal representation with meta-path semantic analysis. J Transl Med. 2023;21(1):1–18.
60. Yang X, Niu Z, Liu Y, Song B, Lu W, Zeng L, Zeng X. Modality-DTA: multimodality fusion strategy for drug–target affinity prediction. IEEE/ACM Trans Comput Biol Bioinform (TCBB). 2022;20(2):1200–10.
61. Zhou D, Xu Z, Li W, Xie X, Peng S. MultiDTI: drug–target interaction prediction based on multi-modal representation learning to bridge the gap between new chemical entities and known heterogeneous network. Bioinformatics. 2021;37(23):4485–92.
62. McInnes L, Healy J, Melville J. UMAP: uniform manifold approximation and projection for dimension reduction. arXiv 2020. https://doi.org/10.48550/arXiv.1802.03426 . arxiv: 1802.03426
63. Rusch TK, Bronstein MM, Mishra S. A survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993 2023.
64. Oono K, Suzuki T. Graph neural networks exponentially lose expressive power for node classification. arXiv preprint arXiv:1905.10947 2019.
65. Chen D, Lin Y, Li W, Li P, Zhou J, Sun X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: Proceedings of the AAAI conference on artificial intelligence, vol. 34. 2020. pp. 3438–3445.
66. Zhao B-W, Su X-R, Hu P-W, Huang Y-A, You Z-H, Hu L. IGRLDTI: an improved graph representation learning method for predicting drug–target interactions over heterogeneous biological information network. Bioinformatics. 2023;39(8):451.