Briefings in Bioinformatics, 2022, **23(4)**, 1–15
https://doi.org/10.1093/bib/bbac260
Advance access publication date: 7 July 2022
Problem Solving Protocol

BatchDTA: implicit batch alignment enhances deep learning-based drug–target affinity estimation

Hongyu Luo†, Yingfei Xiang†, Xiaomin Fang, Wei Lin, Fan Wang, Hua Wu and Haifeng Wang

Corresponding authors. Xiaomin Fang, PaddleHelix team, Baidu Inc., 518000, Shenzhen, China; E-mail: fangxiaomin01@baidu.com; Fan Wang, PaddleHelix team, Baidu Inc., 518000, Shenzhen, China and; E-mail: wang.fan@baidu.com

†Hongyu Luo and Yingfei Xiang contributed equally to this work.

Abstract

Candidate compounds with high binding affinities toward a target protein are likely to be developed as drugs. Deep neural networks (DNNs) have attracted increasing attention for drug–target affinity (DTA) estimation owning to their efficiency. However, the negative impact of batch effects caused by measure metrics, system technologies and other assay information is seldom discussed when training a DNN model for DTA. Suffering from the data deviation caused by batch effects, the DNN models can only be trained on a small amount of 'clean' data. Thus, it is challenging for them to provide precise and consistent estimations. We design a batch-sensitive training framework, namely BatchDTA, to train the DNN models. BatchDTA implicitly aligns multiple batches toward the same protein through learning the orders of candidate compounds with respect to the batches, alleviating the impact of the batch effects on the DNN models. Extensive experiments demonstrate that BatchDTA facilitates four mainstream DNN models to enhance the ability and robustness on multiple DTA datasets (BindingDB, Davis and KIBA). The average concordance index of the DNN models achieves a relative improvement of 4.0%. The case study reveals that BatchDTA can successfully learn the ranking orders of the compounds from multiple batches. In addition, BatchDTA can also be applied to the fused data collected from multiple sources to achieve further improvement.

Keywords: Drug–target affinity, Deep neural network, Batch alignment, Batch effects

Introduction

Evaluation of drug–target affinity (DTA), also known as receptor–ligand affinity, acts as a critical guidance for identifying potential drug candidates in drug discovery. The drug–target binding affinity indicates the strength of binding interaction between the drug (ligand/compound) and target (receptor/protein). Laboratory experiments [35], e.g. *in vivo* and *in vitro* experiments, are usually involved in measuring the affinities between the protein and the candidate compounds. The compounds with the highest measured affinities are screened for further validation and could be future drugs. Since laboratory experiments are laborious, expensive and time-consuming, DTA prediction using machine learning methods, especially deep neural networks (DNNs), has become a popular and efficient approach to help accelerate the whole drug discovery process.

Machine learning methods [5–7, 15, 34], especially DNNs [17, 20, 21, 28, 31, 36–40, 42, 43], have gained increasing attention of many scholars, which utilize the affinity data collected from laboratory experiments to infer the affinities of other protein–compound interactions. Some DNN-based DTA models [17, 39] regard the proteins and the compounds as sequences, e.g. amino acid sequences for proteins and SMILES strings for compounds, which can be applied to the DTA scenario with unknown protein structures or compound structures. Due to the lack of spatial structure information, the upper bound of the model accuracy is relatively low. Advanced DTA models attempt to employ the protein structures and the compound structures by regarding the proteins and the compounds as graphs [21, 36] or taking the target–ligand complex as graphs/3D images [20, 22, 31]. Considering the geometry information can better model the spatial relationship between proteins and compounds to achieve better performance.

Usually, the model parameters of the mainstream DNN models for DTA [17, 21, 28, 36, 39, 40] are optimized through the pointwise training framework, which minimizes the distance between the estimated affinity and the ground-truth affinity for each protein–compound interaction. Due to the lack of consideration of batch effects in model training, the DNN models trained by the pointwise framework is difficult to fully exploit the existing affinity data from multiple sources, measured by various techniques and systems. The batch effects have been widely discussed in many previous studies [8, 30], and this paper defines the batch effects as the systematic batch variation caused by differences in measure metrics, system technologies, laboratory conditions, reagents and other assay information. Many published studies [1, 3] have shown that the systematic variation in experimentally collected data may lead to incorrect biological conclusions. Lots of techniques [12,

Received: April 14, 2022. **Revised:** May 23, 2022. **Accepted:** June 3, 2022

29, 49] have attempted to detect and adjust batch effects in laboratory experiments, but the negative impact of batch effects is still barely discussed in the field of DTA estimation, especially DTA estimation through DNN models. Batch effects could lead to the following issues for DNN-based DTA models:

First, different affinity assays may adopt various metrics, e.g. dissociation constant (K_D), inhibition constant (K_I), half-maximal inhibitory concentration (IC_{50}) and half-maximal effective concentration (EC_{50}), as the assay readout. The affinity values of different metrics should be interpreted differently and are usually incomparable (refer to Appendix B). A DNN model trained by the pointwise framework is confused by the multifarious incomparable 'affinities' reporting for the same protein–compound interaction from multiple assays/batches, thus struggling to provide a consistent estimation. Previous academic works tend to train a DNN model on the data points measured by the same metric to get around the metric incomparability issue. Nevertheless, the data points with respect to a specific metric are insufficient to train a capable DNN model, usually with numerous parameters and thus requiring lots of training data points.

Second, even though only utilizing those data points of a specific metric to train a model, the trained model still suffers from variance caused by other assay information. For example, some metrics, e.g. IC_{50} and EC_{50}, are highly assay-specific [23], and it is not recommended to directly compare the results of these metrics across different assay settings. The DNN model is confounded by the multifarious 'affinities' from multiple assays of the same protein–compound interaction, thus struggling to provide consistent estimates.

Consequently, the batch effects caused by various metrics, assay settings and other factors obstruct advanced DNN models from fully exploiting the collected data points or achieving more satisfactory precision. It is also attractive to take advantage of fused multi-source data points with various assay settings and align these data points to further enhance the DNN model's ability.

To facilitate the effectiveness of the DNN models for DTA, we attempt to alleviate the negative impact of batch effects through a batch-sensitive training framework, namely BatchDTA. BatchDTA learns the ranking orders of the comparable protein-compound interactions within the same batch and uses the common interactions shared between different batches as references for inter-batch alignment, making the comparison between different batches possible. Extensive experiments were conducted to verify that through training with BatchDTA framework, multiple mainstream DNN models, including DeepDTA [39], GraphDTA [36] and MolTrans [17], can achieve higher concordance indexes (CIs) and lower deviation. We also visualized a case to demonstrate that BatchDTA can successfully learn the ranking orders of the compounds from different batches through the reference interaction. Furthermore, BatchDTA was applied to utilize fused data points measured by multiple metrics from multiple sources as the training data. The CIs [11] of the DNN models are promoted in most cases, exhibiting the potential practical value of BatchDTA.

Materials and Methods

Problem Formulation

Advanced proposed DNN models for DTA utilize the pointwise training framework, which regards DTA estimation as a classification or regression problem. Since the goal of the DTA estimation task is to locate compounds with higher affinities toward a target protein, we regard the task of DTA estimation as a ranking problem instead. More concretely, a ranking task of DTA is defined as a batch (or an assay) with batch $b = (p, \mathcal{C})$, containing a protein p and a set of the candidate compounds $\mathcal{C}$. The set of candidate compounds is defined as $\mathcal{C} = \{c_i\}|_{i=1}^{n_\mathcal{C}}$, where $n_\mathcal{C}$ represents the number of candidate compounds in the corresponding batch. We expect to screen out the most promising compounds in $\mathcal{C}$ as the potential drugs for target protein p for batch b. Usually, hundred to tens of thousands of batches can be collected from a data source, $\mathcal{S} = \{(b_i, \mathbf{y}_i)\}|_{i=1}^{n_s}$, where n_s represents the number of batches in data source $\mathcal{S}$. The set of the corresponding labels, i.e. collected affinities, of the candidate compounds $\mathcal{C}$ in batch b is denoted by $\mathbf{y} = \{y_i\}|_{i=1}^{n_\mathcal{C}}$. Additionally, we attempt to simultaneously exploit the data from various data sources $\{\mathcal{S}_i\}|_{i=1}^{n_S}$ to train the DNN models for DTA to test the prospect of data fusion, where n_S denotes the number of data sources.

Implicit Batch Alignment

We attempt to align a target protein's corresponding protein–compound interactions from multiple batches. Those interactions are implicitly aligned through the reference interactions that simultaneously appear in multiple batches. The demonstration of implicit batch alignment is shown in Figure 1, and the details will be introduced in the following text.

We first define the ranking order of two compounds concerning a batch. Since two interactions from different batches may target different proteins or their corresponding affinities may be evaluated under different assay settings, the ranking orders of most compound pairs are valueless. Thus, we concentrate on learning the ranking order of two candidate compounds c_i and c_j for a given batch $b = (p, \mathcal{C})$ with $c_i, c_j \in \mathcal{C}$. To formalize, we introduce a new operator $>_b$ for comparison of two compounds. Formula $c_i >_b c_j$ indicates the affinity of compound c_i is significantly larger than that of compound c_j for the same batch b. Since affinities measured by the laboratory assays are usually noisy, a hyperparameter, i.e. deviation ϵ is introduced to determine whether the two measured affinities differ significantly. More concretely, formula $c_i >_b c_j$ indicates that $y_i > y_j + \epsilon$, where y_i and y_j denote the assay-measured affinities of compound c_i and compound c_j in the compound set $\mathcal{C}$ of the batch $b = (p, \mathcal{C})$. Then, for a batch $b = (p, \mathcal{C})$, we can

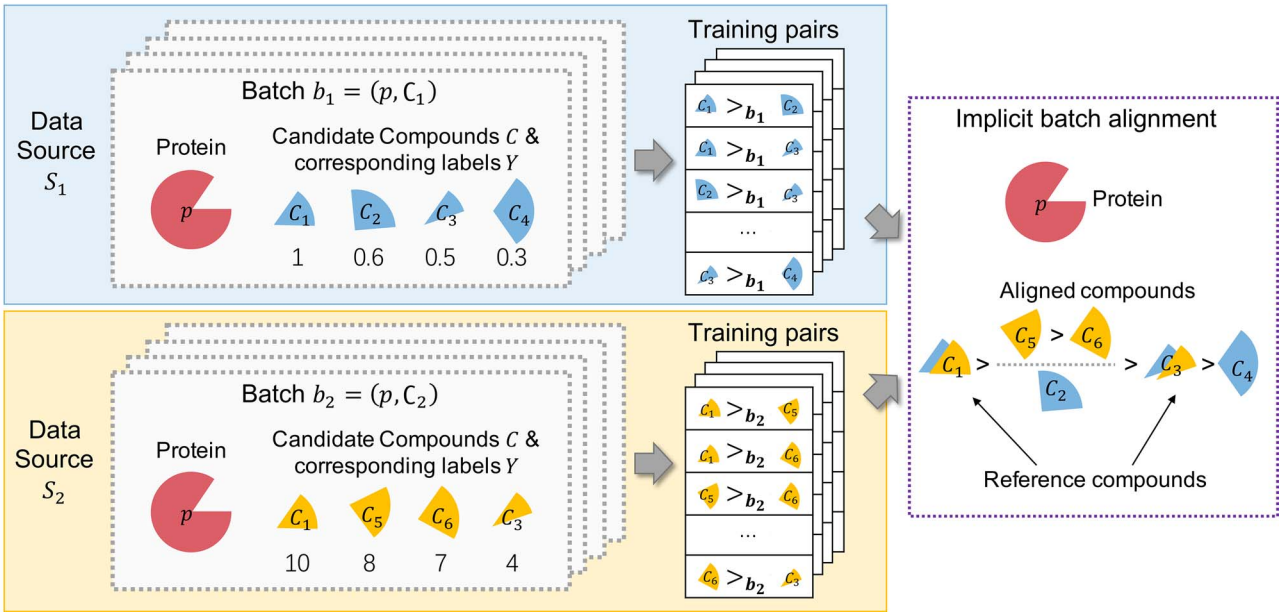


Figure 1. Implicit batch alignment. Compound pairs for training are generated from each batch. The compounds in multiple batches for the same protein are implicitly aligned through the compounds in both batches. (Notched circles represent proteins and sectors of various colors represent compounds)

theoretically generate $\mathcal{O}(n_C^2)$ training pairs of compounds: $\{(c_i, c_j), s.t., c_i >_b c_j\}_{i,j=1}^{n_C}$. We define the training dataset as $\mathbf{D} = \{(p, c_i, c_j), s.t., (b, \mathbf{y}) \in \mathcal{S}_k, b = (p, \mathcal{C}), c_i >_b c_j\}_{k=1}^{n_S}$. Pair (c_i, c_j) is called a training pair as demonstrated on the left of Figure 1, and triple (p, c_i, c_j) is called a training sample.

Then, we introduce reference compounds to align the interactions from different batches. A reference compound is defined as a compound that appears in multiple batches toward the same protein. Given a protein p, there might be multiple batches studying that protein. We assume both batch $b_1 = (p, \mathcal{C}_1)$ and batch $b_2 = (p, \mathcal{C}_2)$ study protein p. A reference compound c^{ref} should satisfy that $c^{ref} \in \mathcal{C}_1 \wedge c^{ref} \in \mathcal{C}_2$. For batch $b_1 = (p, \mathcal{C}_1)$, we denote the compounds in $\mathcal{C}_1$ with the corresponding ground-truth affinities smaller than that of the reference compound c^{ref} as $\mathcal{C}_1^{worse}$. For batch $b_2 = (p, \mathcal{C}_2)$, the compounds in $\mathcal{C}_2$ with the corresponding ground-truth affinities larger than that of the reference compound c^{ref} as $\mathcal{C}_2^{better}$. It is likely the affinity of compound $c_2 \in \mathcal{C}_2^{better}$ is larger than that of compound $c_1 \in \mathcal{C}_1^{worse}$. In this way, the compounds from different batches can be implicitly aligned. A toy case is exhibited in Figure 1. Both the batch $b_1 = (p, \mathcal{C}_1)$ in data source S_1 and the batch $b_2 = (p, \mathcal{C}_2)$ in data source S_2 study the protein p, where $\mathcal{C}_1 = \{c_1, c_2, c_3, c_4\}$ and $\mathcal{C}_2 = \{c_1, c_5, c_6, c_3\}$. Due to the batch effects, the affinity values collected from various sources are incomparable (e.g. from 0.3 to 1.0 in batch b_1 and from 4 to 10 in batch b_2). Training pairs (compound pairs) are extracted from each batch based on the ranking order of the compounds. The training pairs from different batches with respect to protein p are collected. As compound c_3 appears in $\mathcal{C}_1$ and $\mathcal{C}_2$, compound c_3 is taken as a reference compound. Through the reference compound c_3, compound c_4 from batch b_1, and compounds c_5 and c_6 from batch b_2 become comparable.

Since compounds from different batches are comparable through implicit batch alignment, we can effortlessly fuse the data points from multiple sources to train a more capable model for DTA.

Learning the Ranking Orders

Many studies utilized DNNs to predict the affinity between proteins and compounds, including DeepDTA [39], GraphDTA [36] and MolTrans [17]. BatchDTA trains these DNN models by learning the ranking orders of compounds in the training pairs $\{(c_i, c_j), s.t., c_i >_b c_j\}_{i,j=1}^{n_C}$, e.g. $\{(c_1, c_2), (c_1, c_3), \cdots, (c_3, c_4), (c_1, c_5), (c_5, c_6), \cdots, (c_6, c_3)\}$ demonstrated in Figure 1. Training through BatchDTA, the affinity of reference compound c_3 estimated by a trained DNN is expected to be larger than that of compound c_4, i.e. $\hat{y}_3 > \hat{y}_4$ for batch b_1. Similarly, $\hat{y}_5 > \hat{y}_3$ and $\hat{y}_6 > \hat{y}_3$ for batch b_2. Then, the compounds from two batches can be compared: $\hat{y}_5, \hat{y}_6 > \hat{y}_3 > \hat{y}_4$. In this subsection, we first elaborate on the notations for a typical DNN and then the method to learn the ranking orders of the compounds in the training pairs.

A typical DNN model for DTA prediction consists of three components: Protein Encoder, Compound Encoder and Interaction Estimator, as shown on the left of Figure 2.

- **Protein Encoder.** A protein can be depicted by an amino acid sequence or a three-dimensional structure. A Protein Encoder applies various kinds of neural networks, e.g. Convolutional Neural Networks (CNNs) [26, 27], Transformers [45] and Graph Neural Networks (GNNs) [20, 21, 31] to produce the representation vectors of the proteins. We formalize

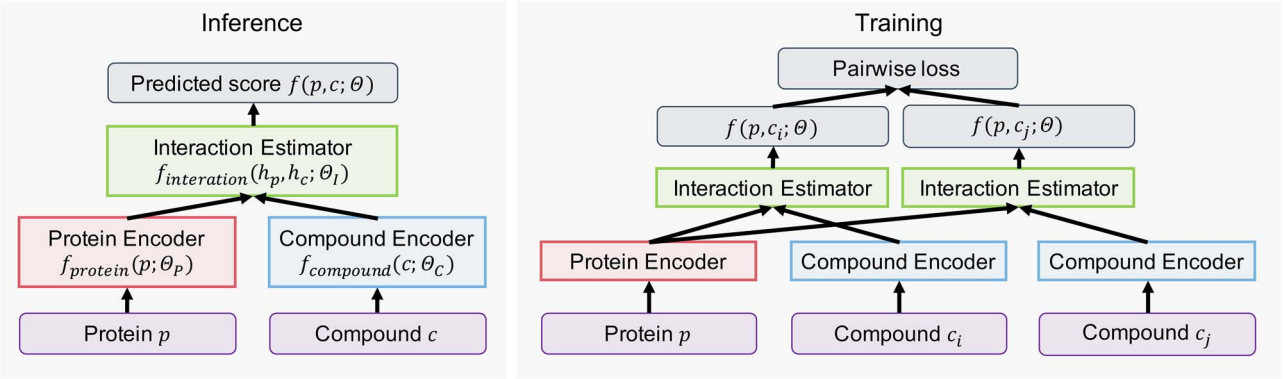


Figure 2. Inference of a typical DNN model $f(p,c;\Theta)$ for DTA and the training framework that trains the DNN model $f(p,c;\Theta)$ through learning the ranking orders of the candidate compounds.

the Protein Encoder as a function $f_{protein}(p;\Theta_P)$, taking a protein p as input, where Θ_p represents the corresponding model parameters. The representation vector of a protein p is defined as $h_p = f_{protein}(p;\Theta_P)$.

- **Compound Encoder.** A Compound Encoder typically generates the compounds' representation vectors through molecular fingerprints (e.g. ECFP [41] and MACCS [10]), sequence-based representation methods (e.g. LSTMs [16] and Transformers) or graph-based representation methods (e.g. GNNs) to model the structures [36]. We describe the Compound Encoder as a function $f_{compound}(c;\Theta_C)$, taking a compound c as input, where Θ_C represents the model parameters needed to be optimized. Then, the representation vector of a compound c can be written as $h_c = f_{compound}(c;\Theta_C)$.
- **Interaction Estimator.** The Interaction Estimator estimates the interaction score between a protein p and a compound c in accordance to a function $f_{interation}(h_p,h_c;\Theta_I)$, where Θ_I denotes the function's learnable parameters. Function $f_{interation}(h_p,h_c;\Theta_I)$ takes the representation vectors h_c and h_p as inputs and outputs an estimated affinity score.

The parameters of whole model $f(p,c;\Theta)$ can be formulated as Eq. 1, combining $f_{protein}(p;\Theta_P)$, $f_{compound}(c;\Theta_C)$ and $f_{interaction}(h_p,h_c;\Theta_I)$:

$$f(p,c;\Theta) = f_{interaction}(f_{protein}(p;\Theta_P), f_{compound}(c;\Theta_C);\Theta_I), \quad (1)$$

where $\Theta = \{\Theta_C,\Theta_P,\Theta_I\}$ are the model parameters that need to be optimized. The candidate compounds are ranked in accordance with the affinities estimated by the DNN model $f(p,c;\Theta)$, and the compounds with the highest estimated affinities are selected as promising drugs for further validation.

Advanced studies optimize the model parameters Θ to minimize the absolute error between the assay-measured affinity and the model-estimated affinity of a protein–compound pair. In contrast to the previously applied training framework (pointwise training), we adopt a pairwise training framework to learn the ranking orders of the candidate compounds of a given batch, as shown on the right of Figure 2. Pairwise training frameworks are widely used in information retrieval and recommender systems [4, 32], which learn the ranking orders. More concretely, the pairwise training framework takes the sample $(p,c_i,c_j) \in \mathbf{D}$ as input to learn the order of compounds c_i and c_j with respect to protein p. Following the classical pairwise ranking method RankNet [4], we regard the pairwise order learning problem as a binary classification task, where the cross-entropy is used as the loss function for the sample $(p,c_i,c_j) \in \mathbf{D}$:

$$\begin{aligned} L(p,c_i,c_j;\Theta) &= -P_{p,c_i,c_j} log \hat{P}_{p,c_i,c_j} \\ &\quad - (1-P_{p,c_i,c_j}) log(1-\hat{P}_{p,c_i,c_j}), \\ P_{p,c_i,c_j} &= \delta(y_i,y_j), \\ \hat{P}_{p,c_i,c_j} &= \frac{exp(o_{p,c_i,c_j})}{1+exp(o_{p,c_i,c_j})}, \\ o_{p,c_i,c_j} &= f(p,c_i;\Theta) - f(p,c_j;\Theta). \end{aligned} \quad (2)$$

P_{p,c_i,c_j} indicates whether the ground-truth affinity score of compound c_i, i.e. y_{p,c_i} is larger than that of compound c_j, i.e. y_{p,c_j}. $\delta(x,y)$ is an indicator function such that $\delta(x,y)=1$ if $x > y$, and $\delta(x,y)=0$ if $x \le y$. Besides, $\hat{P}_{p,c_i,c_j}$ represents the model-estimated probability whether the affinity of compound c_i is larger than that of compound c_j. $\hat{P}_{p,c_i,c_j}$ applies Sigmoid function [13] on o_{p,c_i,c_j}, i.e. the difference between the estimated affinities $f(p,c_i;\Theta)$ and $f(p,c_j;\Theta)$. The cross-entropy between the ground-truth probability P_{p,c_i,c_j} and the estimated probability $\hat{P}_{p,c_i,c_j}$ is minimized to learn the model parameters of the DNN model $f(p,c;\Theta)$. As an affinity pair $(p,c_i,c_j) \in \mathbf{D}$ satisfies $c_i >_b c_j$ and $y_i > y_j$, $P_{p,c_i,c_j} = 1$, the loss function for a sample (p,c_i,c_j) in Eq. 2 can be simplified as

$$\begin{aligned} L(p,c_i,c_j;\Theta) &= -log\frac{exp(o_{p,c_i,c_j})}{1+exp(o_{p,c_i,c_j})}, \\ o_{p,c_i,c_j} &= f(p,c_i;\Theta) - f(p,c_j;\Theta). \end{aligned} \quad (3)$$

Then, the overall loss function of dataset **D** is defined as

$$L(\mathbf{D};\Theta) = \sum_{(p,c_i,c_j)\in\mathbf{D}} L(p,c_i,c_j;\Theta). \quad (4)$$

We minimize the loss function $L(\mathbf{D};\Theta)$ to optimize the model parameters of $f(p,c;\Theta)$.

As we mentioned above, we can theoretically produce $\mathcal{O}(n_c^2)$ samples for a batch $b=(p,\mathcal{C})$. In order to balance the effects of different batches for training, we randomly sample n_{sample} samples for each compound within a batch b at each iteration, where N is the sampling times. In this way, for each batch, only $O(n_{sample}\cdot n_c)$ samples are exploited for training at each iteration.

Results

We use the pointwise framework and BatchDTA to train multiple advanced DNN models, respectively, to observe the effect of BatchDTA.

Datasets

The DNN models are trained on three DTA datasets, including BindingDB [33], Davis [9] and KIBA [44]. The BindingDB dataset collected affinities with various metrics (K_D, K_I, IC_{50} and EC_{50}) and other assay information from multiple data sources. The affinities in the Davis dataset are measured by K_D, while those in the KIBA dataset are indicated by the KIBA score that integrates multiple bioactivity types, i.e. K_D, K_I and IC_{50}. These datasets are processed to filter out the invalid protein–compound interactions (refer to Appendix. A) for data processing).

Previous DTA studies [21, 36, 40] randomly split the protein–compound interactions of a dataset into a training set and a test set. However, random splitting oversimplifies the task of DTA. Due to multiple batches, a target protein in the test set could have already been observed in the training set, resulting in information leakage and model overfitting. Therefore, we split the datasets on the basis of batches and ensure that the proteins observed in the training set will not appear in the test set, which is more in line with the real-world applications. For each dataset, we strive to identify the batches based on the available assay information recorded in the corresponding dataset. Since the batch IDs are not recorded in the datasets, we cannot completely guarantee the accuracy of batch identification (refer to Appendix. A for batch identification of each dataset). The statistics of the datasets are shown in Table A2.

DNN Models for DTA

We compare two training frameworks, *Pointwise* and *BatchDTA*, by applying them to three previously proposed DNN models for DTA:

Table 1. Data analysis of dataset BindingDB.

	K_D	K_I	IC_{50}	EC_{50}	**All**
Proteins appearing in multiple batches	40.0%	42.2%	40.5%	36.1%	44.0%
Batches with interactions appearing in another batch	43.3%	55.7%	48.5%	48.6%	57.9%
Inconsistent ranking orders	12.4%	8.1%	7.1%	14.6%	13.9%

- **DeepDTA** [39] employs three-layers CNNs as Protein Encoder and Compound Encoder to encode the protein sequences and the compound SMILES strings, respectively. Then, for the Interaction Estimator, the encoded protein and compound are concatenated to predict the affinity score.
- **GraphDTA** [36] regards each compound as a graph and attempts several GNNs, such as GIN, GAT, GCN and GAT-GCN, as the Compound Encoders to represent the compounds. In the meantime, GraphDTA regards each protein as a sequence and adopts CNNs as the Protein Encoder to encode the proteins. The Interaction Estimator is the same as DeepDTA's. In this paper, we implement two versions of GraphDTA, denoted by GraphDTA(GCN) and GraphDTA(GATGCN), respectively.
- **MolTrans** [17] decomposes the compounds' SMILES strings and the proteins' acid amino sequences into high-frequency sub-sequences. Then, it applies Transformers as the Compound Encoder and Protein Encoder to obtain the augmented representation with the chemical semantics. MolTrans uses the outer-product operator and CNN blocks as Interaction Estimator to capture the high-order interaction between the compounds and proteins.

For all the DNN models, we use the hyper-parameters suggested by the corresponding papers.

Training and Evaluation Settings

We follow the settings of the previous works to implement *Pointwise* framework, where the mean squared error (MSE) between the models' predicted values and the ground-truth affinities are taken as the loss function to optimize the model parameters. For the *BatchDTA* framework, we apply grid search to search the hyper-parameters, including sample times, learning rate and batch size. The details of hyper-parameter settings are described in Appendix C.1.

CI [11] is used to evaluate the performance of *Pointwise* and *BatchDTA* training frameworks in terms of batches. The CI for a given batch $b=(p,\mathcal{C})$ is defined as

$$CI(b) = \frac{1}{z}\sum_{c_i,c_j\in\mathcal{C}} I(y_i-y_j)\cdot I(f(p,c_i;\Theta)-f(p,c_j;\Theta)), \quad (5)$$

where $z = \sum_{c_i,c_j \in \mathcal{C}} I(y_i - y_j)$, and $I(.)$ is an indicator function. $I(x) = 1$ if $x > \epsilon^*$, and $I(x) = 0$ otherwise. ϵ^* is used to identify the compound pairs with large gap of their corresponding affinities. ϵ^* set to be 0 by default unless otherwise specified.

Then, we define the overall CI for all the test batches $\mathcal{B} = \{b_i\}|_{i=1}^{n_\mathcal{B}}$ by summarizing $CI(b)$, where $n_\mathcal{B}$ is the number of test batches. The overall CI is formalized as

$$CI = \frac{1}{\sum_{b \in \mathcal{B}} w(b)} \sum_{b \in \mathcal{B}} w(b) CI(b), \tag{6}$$

where $w(b) = n_\mathcal{C}$ for task $b = (p, \mathcal{C})$ is introduced to balance the impact of different batches.

Overall Performance

To verify that implicit batch alignment can alleviate the negative impact of batch effects, we trained multiple previously proposed DNN models for DTA through the *Pointwise* and *BatchDTA* training frameworks, respectively. We expect a DNN model trained by *BatchDTA* framework can achieve further improvement compared with that trained by *Pointwise* framework.

Database BindingDB records the affinities of millions of protein–compound interactions measured by various metrics. Most data points in BindingDB are measured by metrics inhibition constant (K_I) and half-maximal inhibitory concentration (IC_{50}). A total of 200 000 protein–compound interactions from 10 000 batches are measured by K_I, while 600 000 interactions from 20 000 batches are measured by IC_{50} (refer to Table A2 for statistics of the datasets). Since the assay information, such as temperature, pH degree and competitive substrate, affect the values of K_I and IC_{50}, the data points measured by K_I and IC_{50} are suitable for verifying whether *BatchDTA* can effectively mitigate the impact of batch effects. The dataset of each metric is divided into the training set, validation set and test set by 8:1:1. A model is trained on the training set with the best epoch selected by the validation set. For each experimental setting, we trained each model five times and evaluated the CIs on all the compound pairs ($\epsilon^* = 0$) as well as the confident compound pairs ($\epsilon^* = 0.6$) in the test set. Confident compound pairs are introduced to diminish the impact of the experimental variance for the evaluation. We eliminated the compound pairs with small affinity gaps from all the compound pairs, and the remaining pairs are called confident compound pairs (refer to Appendix C.3 for details of confident compound pairs generation).

Figure 3 shows the CIs of four DNN models trained by *Pointwise* framework and *BatchDTA* framework. We can draw the following conclusions:

- Since *BatchDTA* is a batch-sensitive training framework, it can make more rational use of the collected data points than the *Pointwise* framework, which could further enhance the ability and robustness of a DNN model for DTA. *BatchDTA* framework significantly outperforms *Pointwise* framework under 14/16 settings. Besides, in most cases, the standard deviation of CIs of the model trained by the *Pointwise* framework is greater than that by the *BatchDTA* framework. We suspect that the conflicting data between batches cause the larger variance of the model trained by the *Pointwise* framework.
- For each DNN model, the relative improvement of *BatchDTA* compared with *Pointwise* is higher when evaluating on the confident compound pairs than on all the compound pairs. Since the gaps between the measured affinities of some compounds in a batch are not significant, excluding those incomparable compound pairs in the evaluation can highlight the advantages of *BatchDTA*. For example, for K_I, *BatchDTA* achieves an average relative improvement of 3.2% on all the compound pairs, while 5.5% on the confident compound pairs. For IC_{50}, the average relative improvement is 4.7% for all the compound pairs and 7.8% for confident compound pairs.
- Compared with K_I, IC_{50} is also affected by the concentration of enzyme and substrate, and thus the systematic variance of the assays measured by IC_{50} is larger than that measured by K_I. The larger variance makes a DNN model more difficult to provide accurate estimations on the IC_{50} dataset, but we can observe that all the DNN models trained by *BatchDTA* can still achieve further improvement.

Case Study

We further explore whether *BatchDTA* can successfully align and learn the ranking orders of the compounds from multiple batches by case study. We selected a case of two batches toward the same protein from the training set of IC_{50} with a reference compound appearing in both batches. The reference compound is expected to implicitly align the compounds from two batches. In Figure 4, we visualize three matrices with each element representing the affinity gap between a compound in Batch 1 and a compound in Batch 2. The affinity gaps are normalized, and the darker the color in the matrices, the greater the affinity gap. More concretely, we sorted compounds in Batch 1 and Batch 2 according to the ground-truth affinities, respectively. For each matrix, each column stands for a compound from Batch 1, each row stands for a compound for Batch 2 and each element stands for a compound pair with its color indicating the affinity gap. The three matrices describe the gaps between the ground-truth affinities collected in the dataset, the gaps between the predicted affinities of the MolTrans model trained by the *Pointwise* framework and the gaps between the predicted affinities of the MolTrans model trained by *BatchDTA* framework, respectively.

In general, the color pattern of the gaps of affinities estimated by *BatchDTA* framework (Figure 4c) is similar to that of the gaps of ground-truth affinities (Fig. 4a). The colors in both the ground-truth matrix and *BatchDTA*

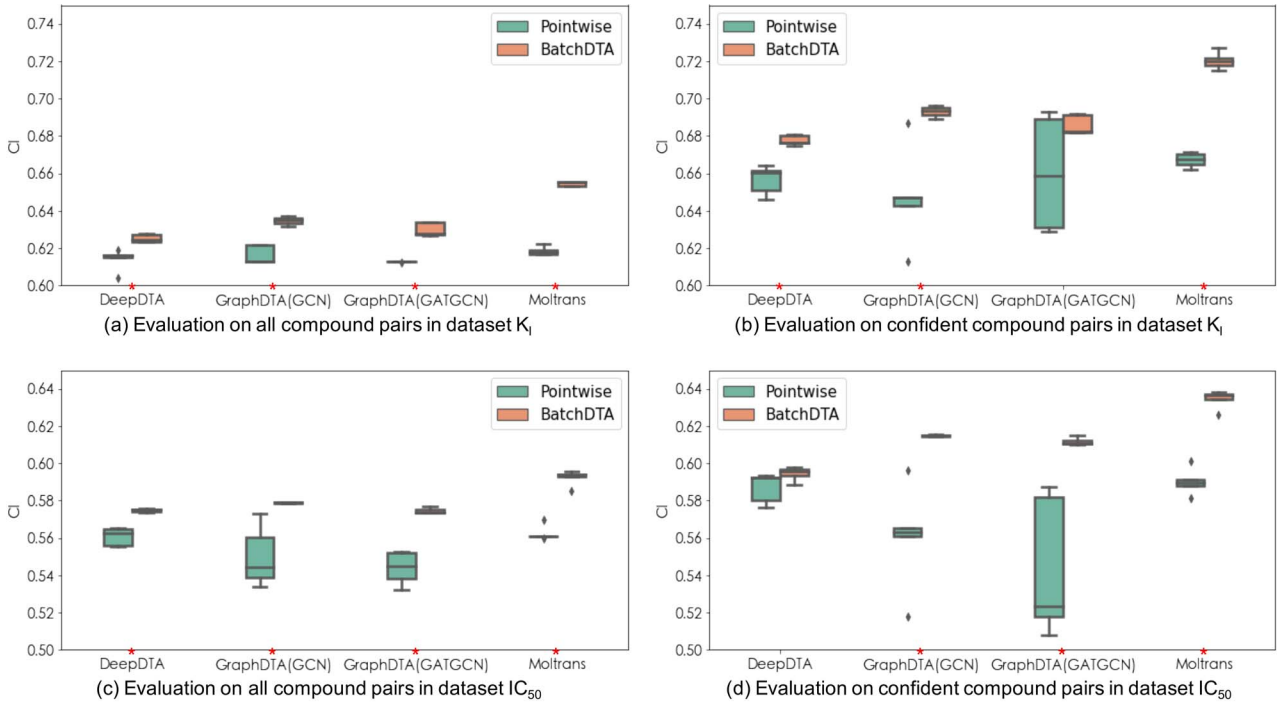


Figure 3. Comparing the CIs of the DNN models trained by *Pointwise* framework and *BatchDTA* framework on database BindingDB under different settings. (A red star stands for statistical significance with P-value less than 0.05 for a two-tailed test.)

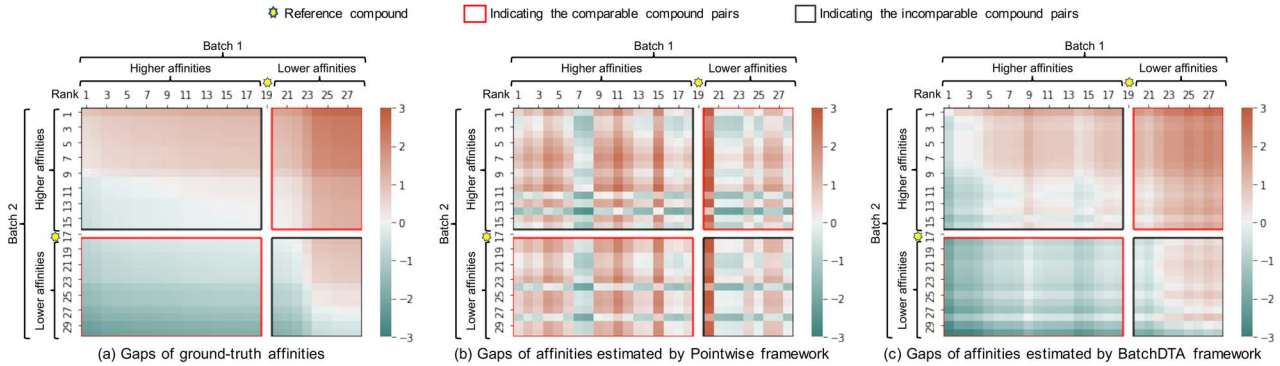


Figure 4. Case study analyzing whether *Pointwise* framework and *BatchDTA* framework can learn the ranking orders of the compound from two batches for the same protein.

matrix transition smoothly, with dark red at the upper right, dark green at the lower left and white at the diagonal. The consistent color patterns reveal that *BatchDTA* can well learn the ranking order of a compound pair with one compound in Batch 1 and the other in Batch 2. In contrast, the color pattern of *Pointwise* is chaotic, which is distinct from that of the ground-truth. More specifically, the grids (compound pairs) in a matrix can be divided into two portions: the comparable compound pairs indicated by the red boxes (lower left and upper right) and the incomparable compound pairs indicated by the black boxes (lower right and upper left). For example, the compounds with higher affinities than the reference compound in Batch 1 (the first 18 columns) should rank ahead of those with lower affinities than the reference compound in Batch 2 (the last 13 rows). Thus, the corresponding compound pairs indicated by the red box at the lower left are regarded as comparable compound pairs. The model trained by *BatchDTA* can infer the compounds' ranking order with a high certainty (indicated by darker color) for most of the comparable compound pairs. For the incomparable compound pairs, although the model trained by *BatchDTA* cannot determine some compounds' ranking order, e.g. the ranking order of the first compound in Batch 1 and the first compound in Batch 2, the overall ranking predictions are consistent with those of the ground-truth.

Fusing Data

The *BatchDTA* framework can also be applied to the fused data points collected from multiple sources. It is attractive to take advantage of fused data points to train the

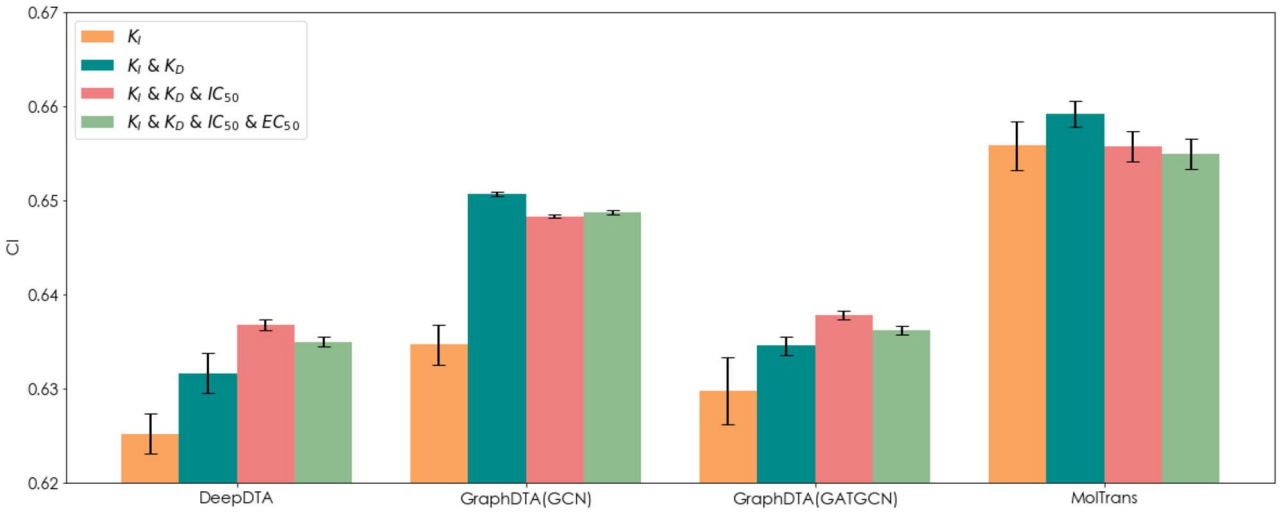


Figure 5. Exploring the performance of the DNN models trained on the fused data points measured by multiple metrics, i.e. K_I, K_D, IC_{50} and EC_{50} by *BatchDTA* framework.

DNN models for DTA. Therefore, we explore the practicality of *BatchDTA* in two data fusion scenarios: fusing data measured with multiple metrics and fusing data from multiple open-source datasets.

Fusing Data Points Measured by Multiple Metrics

We utilized the database BindingDB and trained the DNN models on the fusing data measured with multiple metrics. Each DNN model is trained on four datasets, i.e. K_I, K_I&K_D, K_I&K_D&IC_{50} and K_I&K_D&IC_{50}&EC_{50}, respectively, by *BatchDTA* framework, where operator & denotes fusing the data points measured by different metrics. The trained DNN models are evaluated on the test set of K_I. We consider the correlations between the metrics (refer to Appendix B) when designing the experiments. Since the metrics with higher correlations to K_I are more likely to provide advantages to the DNN models that only trained on the data points of K_I, we first integrated the data points of K_D into the training set, then those of IC_{50} and finally those of EC_{50}. Each experiment is run five times, and the performance of the DNN models trained on the fused data points measured by multiple metrics is shown in Figure 5. We can observe the following phenomena: (1) By adding the data points of K_D to the training set, all the DNN models gain significant performance improvement (average relative improvement of 1.21%). Since K_I and K_D have a strong correlation (Pearson r = 0.83), the data points of K_D potentially augment valuable information. (2) Due to the higher systematic deviations of the data measured by IC_{50} and EC_{50}, the correlation between K_I and IC_{50} as well as the correlation between K_I and EC_{50} are relatively low (Pearson r = 0.74 and 0.64, respectively). Thus, fusing the data points of IC_{50} and EC_{50} cannot further significantly raise the accuracy of the DNN models. These phenomena enlighten us that integrating high correlation data into the training set could improve the effectiveness of existing DNN models on DTA.

Fusing Data Points from Multiple Datasets

Datasets Davis and KIBA are utilized to explore the potential of *BatchDTA* when training the DNN models on the fused data points collected from multiple datasets. Since these two datasets contain only hundreds of batches (refer to Table A2 in Appendix), we reserve one-sixth of the data as the test set and perform 5-fold cross-validation [2] on the remaining five-sixths data to reduce experimental deviation. When evaluating the DNN models on the Davis dataset, we compare three training methods: *Pointwise (Davis)* and *BatchDTA (Davis)* that only utilize the Davis dataset for training, as well as *BatchDTA (Davis + KIBA)* that utilize both the datasets Davis and KIBA for training. The setting is similar when evaluating the DNN models on the KIBA dataset. Figure 6 exhibits the comparison of the DNN models trained on the fused data points collected from multiple datasets. As we expected, the CIs of the DNN models trained on only one dataset by *BatchDTA* framework surpass those by *Pointwise* framework, which is consistent with the evaluation results of Figure 3 on database BindingDB. Furthermore, by combining the datasets Davis and KIBA for training, the DNN models trained by *BatchDTA* achieve better performance in general. The better results of the fused datasets illustrate that as a DNN model usually requires lots of training data, the additional data points for training are likely to benefit the ability of the DNN models for DTA.

Result Summary

In this paper, we train four DNN models through *BatchDTA* framework for DTA estimation to verify that *BatchDTA* can boost various DNN models to achieve better accuracy. We also attempt to fuse the data points from multiple metrics as well as the data points from multiple datasets to demonstrate that integrating more data could further benefit DTA estimation. The result

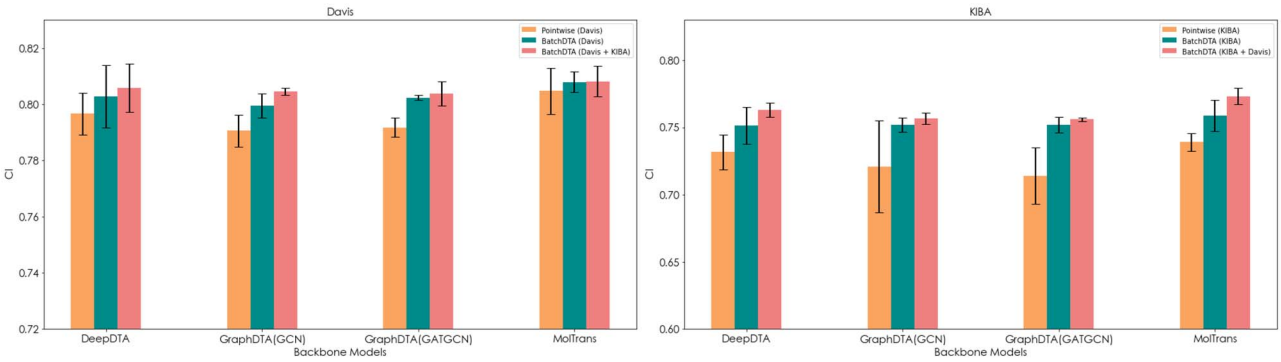


Figure 6. Exploring the performance of the DNN models trained on the fused data points collected from multiple datasets, i.e. Davis and KIBA, by *BatchDTA* framework.

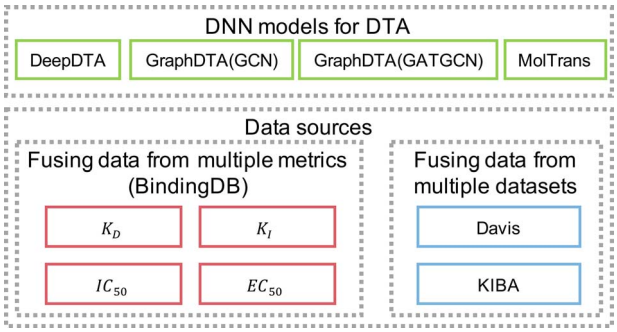


Figure 7. Result summary of BatchDTA for DTA estimation.

summary of *BatchDTA* for DTA estimation is exhibited in Figure 7.

Discussion

Many published literature have discussed the impact of batch effects, but little work has been done to explore the batch effects on the DNN models for DTA. Table 1 reports the data analysis of the database BindingDB. We attempt to analyze the impact scope of batch effects for DTA through the dimensions of proteins and batches (the first two rows in Table 1) and investigate which group of the impacted data points (the last row in Table 1) can be effectively taken advantage of by BatchDTA to achieve better results. From the table, we can observe: (1) About 40% of proteins simultaneously appear in multiple batches, and the collected affinities of corresponding data points could be inconsistent. A batch-sensitive training framework is demanded to alleviate the negative impact of those inconsistent data points. (2) There are 43.3 to 55.7% of batches with interactions appearing in another batch for each metric. That means about half of the batches can be implicitly aligned through those interactions (reference compounds), and BatchDTA has the opportunity to model those batches better. (3) We extracted the compound pairs that appeared in multiple batches against the same protein from the datasets and found that the ranking orders of a small number of compound pairs (7.1 to 14.6%) were inconsistent. We suspect the inconsistency is caused by the assay errors, batch identification errors, biological variance and other factors. Employing BatchDTA to train DNN models cannot overcome the impact of those noisy data points. Fortunately, the ranking orders of only less than 1% of all the compound pairs are inconsistent.

We further investigate the impact of the quality of batch identification. Figure 8 compares the effects of two batch identification methods. For the first method *Batch*, we try our best to distinguish different batches according to the fields recorded in the database. For the second method *Protein*, all the data points with respect to the same protein are grouped in the same batch. Thus, the quality of method *Batch* is higher than that of method *Protein* for batch identification. The results of dataset K_I (Figure 8a) illustrate that the quality of batch identification plays a critical role in the effectiveness of BatchDTA. Besides, for dataset IC_{50}, the CIs of methods *Batch* and *Protein* are comparable. Since IC_{50} is affected by more factors as mentioned, but those factors are not recorded in the database, we cannot precisely identify the batches from the database. We believe that the DNN models trained by BatchDTA will benefit from more detailed assay information.

Since BatchDTA turns a DTA estimation task from a regression/classification problem into a ranking problem, a DNN model trained by BatchDTA cannot quantitatively estimate the affinity score of a compound toward a target protein. The affinity value estimated by the DNN model cannot correspond to any specific metric used in the assays. It is difficult for pharmaceutical chemists to draw effective conclusions from the affinity estimation of a single compound toward a protein. Thus, a DNN model trained by the BatchDTA framework is applicable for virtual drug screening but quantitative affinity evaluation.

Conclusions

Many advanced studies have applied deep learning to DTA estimation to screen the promising compounds toward the given target proteins. Existing studies focus more on designing DNN model architectures for DTA. We argue that a batch-sensitive training framework,

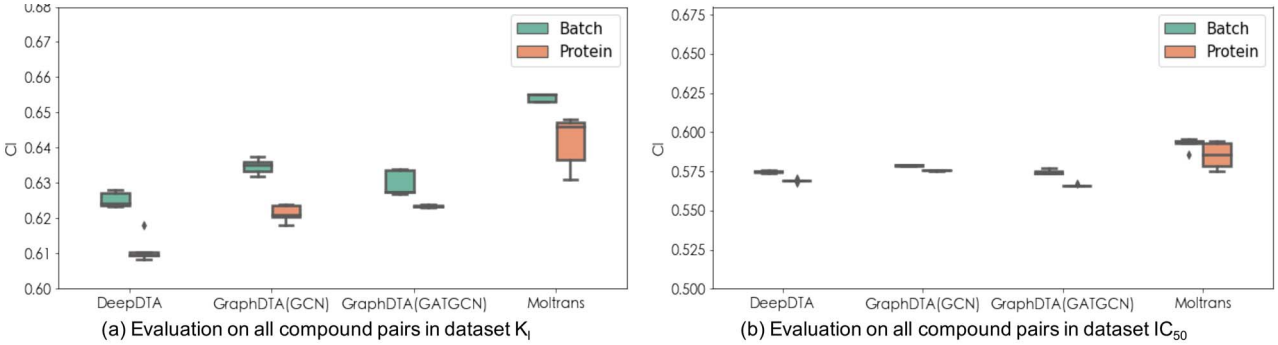


Figure 8. Impact of the quality of batch identification methods.

i.e. BatchDTA, can further enhance the accuracy and robustness of the previously proposed DNN models, weakening the influence of batch effects (systematic variance). Each batch is regarded as a ranking task. BatchDTA implicitly aligns the corresponding compounds toward the same protein from multiple batches by learning the compounds' ranking orders in each batch. Extensive experimental results demonstrate that four advanced DNN models trained by BatchDTA can achieve significant improvement on three datasets for DTA. BatchDTA can also be applied to train the DNN models on fused data collected from multiple sources to further enhance the models' precision. We believe our work could inspire researchers working on applying machine learning methods to drug discovery.

Key Points

- We utilize a batch-sensitive training framework, called BatchDTA, to alleviate the harmful influence of the batch effects and take advantage of data from multiple sources for DTA prediction.
- BatchDTA aims to implicitly align the compounds from multiple batches for the same target protein by learning the ranking orders of the interactions within a batch.
- Extensive experiments demonstrate that BatchDTA can facilitate the precision and robustness of multiple mainstream DNNs in multiple situations.
- BatchDTA can further improve the models' performance when training the models on the fused collected from different sources.

Code Availability

The code is freely available at GitHub https://github.com/PaddlePaddle/PaddleHelix/tree/dev/apps/drug_target_interaction/batchdta to allow replication of the results.

Data Availability

The raw datasets, BindingDB, Davis, and KIBA are collected from https://www.bindingdb.org/bind/index.jsp, http://staff.cs.utu.fi/~aatapa/data/DrugTarget/ and https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0209-z. The processed data is available at GitHub https://github.com/PaddlePaddle/PaddleHelix/tree/dev/apps/drug_target_interaction/batchdta to allow replication of the results.

Author Contributions Statement

XM.F., F.W., H.W. and HF.W. led the research. HY.L, YF.X, XM.F. and F.W. conceived the experiments. HY.L. YF.X and W.L.conducted the experiments. LY.L, XM.F. and YF.X. wrote the manuscript.

Acknowledgments

The authors thank the anonymous reviewers for their valuable suggestions.

References

1. Akey JM, Biswas S, Leek JT, *et al.* On the design and analysis of gene expression studies in human populations. *Nat Genet* 2007;**39**(7):807–8.
2. Arlot S, Celisse A. A survey of cross-validation procedures for model selection. *Statistics surveys* 2010;**4**:40–79.
3. Baggerly KA, Edmonson SR, Morris JS, *et al.* High-resolution serum proteomic patterns for ovarian cancer detection. *Endocr Relat Cancer* 2004;**11**(4):583–4.
4. Burges C, Shaked T, Renshaw E, *et al.* Learning to rank using gradient descent. In: *Proceedings of the 22nd international conference on Machine learning*, 2005, 89–96.
5. Chen R, Liu X, Jin S, *et al.* Machine learning for drug-target interaction prediction. *Molecules* 2018;**23**(9).
6. Cichonska A, Pahikkala T, Szedmak S, *et al.* Learning with multiple pairwise kernels for drug bioactivity prediction. *Bioinformatics* 2018;**34**(13):i509–18.
7. Cichonska A, Ravikumar B, Parri E, *et al.* Computational-experimental approach to drug-target interaction mapping: a case study on kinase inhibitors. *PLoS Comput Biol* 2017;**13**(8):e1005678.
8. Čuklina J, Pedrioli PGA, Aebersold R. Review of batch effects prevention, diagnostics, and correction approaches. In: *Mass spectrometry data analysis in proteomics*. Springer, 2020, 373–87.
9. Davis MI, Hunt JP, Herrgard S, *et al.* Comprehensive analysis of kinase inhibitor selectivity. *Nat Biotechnol* 2011;**29**(11):1046–51.
10. Durant JL, Leland BA, Henry DR, *et al.* Reoptimization of MDL keys for use in drug discovery. *J Chem Inf Comput Sci* 2002;**42**(5):1273–80.

11. Gönen M, Heller G. Concordance probability and discriminatory power in proportional hazards regression. *Biometrika* 2005;**92**(4):965–70.
12. Haghverdi L, Lun ATL, Morgan MD, *et al.* Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors. *Nat Biotechnol* 2018;**36**(5):421–7.
13. Han J, Moraga C. The influence of the sigmoid function parameters on the speed of backpropagation learning. In: *International workshop on artificial neural networks*. Springer, 1995, 195–201.
14. Hastie T, Tibshirani R, Friedman JH, *et al. The elements of statistical learning: data mining, inference, and prediction*, Vol. **2**, 2009, Springer.
15. He T, Heidemeyer M, Ban F, *et al.* Simboost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines. *J Chem* 2017;**9**(1):1–14.
16. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput* 1997;**9**(8):1735–80.
17. Huang K, Xiao C, Glass LM, *et al.* Moltrans: Molecular interaction transformer for drug–target interaction prediction. *Bioinformatics* 2021;**37**(6):830–6.
18. Hulme EC, Trevethick MA. Ligand binding assays at equilibrium: validation and interpretation. *Br J Pharmacol* 2010;**161**(6): 1219–37.
19. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *International conference on machine learning*. PMLR, 2015, 448–56.
20. Jiang D, Hsieh C-Y, Zhenxing W, *et al.* Interactiongraphnet: A novel and efficient deep graph representation learning framework for accurate protein–ligand interaction predictions. *J Med Chem* 2021;**64**(24):18209–32.
21. Jiang M, Li Z, Zhang S, *et al.* Drug–target affinity prediction using graph neural network and contact maps. *RSC Adv* 2020;**10**(35):20701–12.
22. Jiménez J, Skalic M, Martinez-Rosell G, *et al.* K deep: protein–ligand absolute binding affinity prediction via 3d-convolutional neural networks. *J Chem Inf Model* 2018;**58**(2):287–96.
23. Kalliokoski T, Kramer C, Vulpetti A, *et al.* Comparability of mixed ic50 data–a statistical analysis. *PloS one* 2013;**8**(4):e61007.
24. Kingma DP, Ba J. Adam: A method for stochastic optimization-arXiv preprint arXiv:1412.6980. 2014.
25. Krizhevsky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional networks. In: *Proceedings of the Conference Neural Information Processing Systems (NIPS)*, Vol. **1097**.
26. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, Vol. **25**, 2012, 1097–105.
27. LeCun Y, Bengio Y, *et al.* Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 1995;**3361**(10):1995.
28. 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):e1007129.
29. Leek JT, Evan Johnson W, Parker HS, *et al.* The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics* 2012;**28**(6): 882–3.
30. Leek JT, Scharpf RB, Bravo HC, *et al.* Tackling the widespread and critical impact of batch effects in high-throughput data. *Nat Rev Genet* 2010;**11**(10):733–9.
31. Li S, Zhou J, Tong X, *et al.* (eds). Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, 975–85.
32. Liu T-Y, *et al.* Learning to rank for information retrieval. *Found Trends Inf Retr* 2009;**3**(3):225–331.
33. Liu T, Lin Y, Wen X, *et al.* Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic Acids Res* 2007;**35**(suppl_1):D198–201.
34. Luo Y, Zhao X, Zhou J, *et al.* A network integration approach for drug-target interaction prediction and computational drug repositioning from heterogeneous information. *Nat Commun* 2017;**8**(1):573.
35. Macarron R, Banks MN, Bojanic D, *et al.* Impact of high-throughput screening in biomedical research. *Nat Rev Drug Discov* 2011;**10**(3):188–95.
36. Thin Nguyen, Hang Le, Thomas P Quinn, Tri Nguyen, Thuc Duy Le, and Svetha Venkatesh. GraphDTA: predicting drug–target binding affinity with graph neural networks. *Bioinformatics*, **37**(8):1140–7, 2020.
37. Nguyen TM, Nguyen T, Le TM, *et al.* Gefa: early fusion approach in drug-target affinity prediction. *IEEE/ACM Trans Comput Biol Bioinform* 2021.
38. Özçelik R, Öztürk H, Özgür A, *et al.* Chemboost: A chemical language based approach for protein–ligand binding affinity prediction. *Molecular Informatics* 2020.
39. Öztürk H, Özgür A, Ozkirimli E. DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics* 2018;**34**(17):i821, 09–9.
40. Öztürk H, Ozkirimli E, Özgür A. Widedta: prediction of drug-target binding affinityarXiv preprint arXiv:1902.04166. 2019.
41. Rogers D, Hahn M. Extended-connectivity fingerprints. *J Chem Inf Model* 2010;**50**(5):742–54.
42. Shin B, Park S, Kang K, *et al.* Self-attention based molecule representation for predicting drug-target interaction. In: *Machine Learning for Healthcare Conference*. PMLR, 2019, 230–48.
43. Stepniewska-Dziubinska MM, Zielenkiewicz P, Siedlecki P. Development and evaluation of a deep learning model for protein–ligand binding affinity prediction. *Bioinformatics* 2018;**34**(21):3666–74.
44. Tang J, Szwajda A, Shakyawar S, *et al.* Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis. *J Chem Inf Model* 2014;**54**(3):735–43.
45. Vaswani A, Shazeer N, Parmar N, *et al.* Attention is all you need. In: Guyon I, Luxburg UV, Bengio S *et al.* (eds). *Advances in Neural Information Processing Systems*, Vol. **30**. Curran Associates, Inc, 2017.
46. Vaswani A, Shazeer N, Parmar N, *et al.* Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. *Attention is all you need Advances in neural information processing systems* 2017;**30**.
47. Veličković P, Cucurull G, Casanova A, *et al.* Graph attention networksarXiv preprint arXiv:1710.10903. 2017.
48. Zhang S, Tong H, Jiejun X, *et al.* Graph convolutional networks: a comprehensive review. *Computational Social Networks* 2019;**6**(1):1–23.
49. Zhou W, Koudijs KKM, Böhringer S. Influence of batch effect correction methods on drug induced differential gene expression profiles. *BMC Bioinformatics* 2019;**20**(1):437.

A Appendix: datasets and data process

We describe the details of the data process in this section, and the data statistics of the processed datasets are shown in Table A2.

BindingDB

The BindingDB database with the version of May 2021 contains 2221 487 compound–protein interactions, including 7965 proteins and 963 425 compounds.

First, we cleaned up the raw data from BindingDB by the following steps: (1) Keep compound–protein interactions measured by metrics K_D, K_I, IC_{50} and EC_{50}). (2) Remove the recorded affinities with '>' or '<'. (3) Rescale the affinities larger than 10 000 to be 10 000. (4) Drop out the duplicate records. (5) Remove the compounds with illegal SMILES strings that cannot be processed by the Cheminformatics software RDKit (https://rdkit.org). (6) Remove the proteins that cannot be converted to FASTA format (https://zhanggroup.org/FASTA/). 'Second, we identify the batches from the cleaned data: (1) Multiple fields in BindingDB, including 'BindingDB Target Chain Sequence', 'pH', 'Temp (C)', 'Curation/DataSource', 'Article DOI', 'PMID', 'PubChem AID', 'Patent Number', 'Authors' and 'Institution', are considered to identify batches. (2) Remove the batches with less than 10 candidate compounds.

Davis and KIBA

We followed the previous work [39] to process Davis and KIBA datasets, and the implementation of which is at https://github.com/hkmztrk/DeepDTA/tree/master/data. Then, the batches are identified according to 'Gene ID' for the Davis dataset and 'Uniprot ID' for the KIBA dataset.

Table A2. Statistics of the datasets.

Dataset	Metric	#Compound	#Protein	#Interaction	#Batch
BindingDB	K_D	6704	587	31 239	1070
	K_I	126,122	1225	249 598	11 292
	IC_{50}	407 462	2582	653 344	20 041
	EC_{50}	75 142	699	105 364	3143
Davis	K_D	68	442	30,056	442
KIBA	KIBA score	2111	229	118 254	229

B Appendix: relations between metrics of binding affinities

Dissociation constant (K_D), inhibition constant (K_I), half-maximal inhibitory concentration (IC_{50}) and half-maximal effective concentration (EC_{50}) are mainly used for measuring binding affinities. Both K_D and K_I are used to indicate the dissociation equilibrium constant, i.e. the reverse of the association constant, for the dissociated components. K_D is a more general term, while K_I is more specifically used under enzyme-inhibitor complex. IC_{50} is the inhibitory concentration, measured by an inhibitor with the response (or binding) reduced by half. EC_{50} stands for effective concentration of an inhibitor with half-maximal response. Comparing with K_I and K_D, IC_{50} and EC_{50} are influenced by more assay information, such as substrate concentration [18].

In particular, through Cheng–Prusoff equation, K_I can be converted from IC_{50} [23]:

$$K_I = \frac{IC_{50}}{1 + \frac{|S|}{Km}}, \tag{B1}$$

where $|S|$ is the substrate concentration, and K_m is the Michaelis–Menten constant of the substrate. Thus, IC_{50} is assay-specific, and the IC_{50} values under different assay information are incomparable.

In Section. 3.6.1, we evaluate the DNN models that are trained on fused data measured by multiple metrics on the test set of K_I. The data points are added gradually according to the Pearson correlation of the corresponding metrics to K_I. The Pearson correlations are shown in Figure B9. We first added the data points of K_D, then IC_{50} and finally EC_{50}.

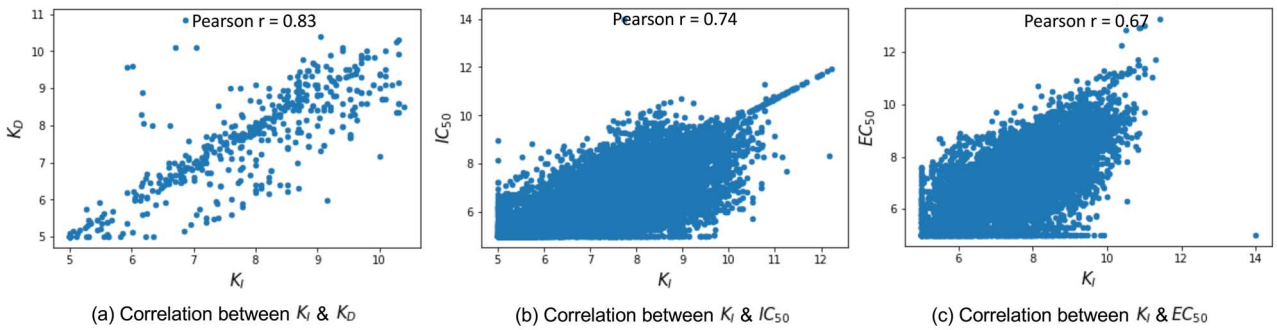
C Appendix: experimental settings

Hyper-parameters of BatchDTA

Following the previous works [15, 21, 36, 39], we normalize the original affinities as labels for training. Take K_D for example. We use pK_D values as labels to train the DNN models:

$$pK_D = -log_{10}(\frac{K_D}{10^9}). \tag{C1}$$

K_I, IC_{50} and EC_{50} and KIBA score are normalized in the same way.



(a) Correlation between K_I & K_D (b) Correlation between K_I & IC_{50} (c) Correlation between K_I & EC_{50}

Figure B9. Pearson correlations between K_I and other metrics.

Table C3. Candidate settings of hyper-parameters for BindingDB.

Hyper-parameter	Candidate settings
deviation ϵ	$\{0.2, 0.5, 1\}$
n_{sample} for K_I	10
n_{sample} for K_D	$\{1, 3, 5\}$
n_{sample} for IC_{50}	$\{0.2, 0.5, 1\}$
n_{sample} for EC_{50}	$\{0.1, 0.2, 0.5\}$
learning rate	$\{1 \times 10^{-3}, 5 \times 10^{-4}\}$
batch size	$\{256, 512\}$

Table C4. Candidate settings of hyper-parameters for Davis and KIBA.

Hyper-parameter	Candidate settings
deviation ϵ	{0.2,0.5,1}
n_{sample}	{0.5, 1, 3, 5, 10}
learning rate	$\{1 \times 10^{-3}, 5 \times 10^{-4}\}$
batch size	{256, 512}

We used Adam optimizer ($\beta = (0.9, 0.999)$, $\epsilon = 1e-08$) [24] and applied grid search to explore the best hyper-parameters for each DNN model on each dataset. In order to ensure the hyper-parameter search has no bias in the test metric, for BindingDB, the hyper-parameters are selected according to the results of the validation set, while for Davis and KIBA, the hyper-parameters are selected according to the results of the validation set of each cross-validation fold.

For BindingDB database, the candidate settings of each hyper-parameter are shown in Table C3.

For Davis and KIBA datasets, the candidate settings of the hyper-parameters for searching are shown in Table C4.

When training a model of typical model architecture, many factors bring randomness to the final trained model, e.g. the initial weights of the model and the training orders of the samples. To alleviate the bias of the evaluation, for BindingDB, we conduct the five independent runs with different random seeds and training orders. We select a model according to its evaluation result on the validation set for each run and average the evaluation results of the five selected models on the test set. For Davis and KIBA, 5-fold cross-validation also brings randomness to the evaluation results since the validation set of each fold is different from that of another fold. Similarly, for each fold, we select a model according to its evaluation result on the validation set and average the evaluation results of the five selected models on the test set.

All the models are trained by NVIDIA Tesla V100 GPUs. Since BatchDTA is a training framework, it would not change the number of model parameters and the time complexity for training.

Architectures of DNN Models

A typical DNN model for DTA contains three modules: Protein Encoder, Compound Encoder and Interaction Estimator. Figure C10 shows the models architectures of three DNN models used in our paper, i.e. DeepDTA, GraphDTAand MolTrans, depicting the detailed structures of their Protein Encoders, Compound Encoders and Interaction Estimators, where

- One-hot Embedding: embedding of one-hot encoding.
- MLP [14]: multilayer perceptron staking multiple fully connnected networks.
- Conv [25]: convolutional layer based on convolution kernels that slide along input features.
- Graph Atten [47]: message passing in the graph by attention mechanism.
- Graph Conv [48]: message passing in the graph by convolution layer.
- Batch Norm [19]: normalization of the layers' inputs by batch-wise rescaling.
- Sub-structure Embedding: embedding of the sub-structure.
- Multi-Head Attention [46]: attention mechanism that utilizes multiple heads.
- Add & Norm: element-wise add and Batch Norm.

Generation of Confident Compound Pairs

To generate the confident compound pairs for BindingDB database, we eliminated the compound pairs with small affinity gaps from all the compound pairs. The probability of the compound pairs with affinity gaps smaller than ϵ^* is shown in Figure C11. When $\epsilon^* = 0.6$, about 50% compound pairs are eliminated.

D Appendix: detailed results

We exhibit the detailed results of all the experiments. T-tests are used to evaluate whether there is a significant difference between the experimental results of the two methods. Two-tailed t-tests are chosen for comparison since we cannot judge in advance whether training a DNN model through the BatchDTA framework with more data points can consistently achieve better performance in all the experimental settings.

Overall Performance

Table D5 shows the detailed results corresponding to Figure 3. The DNN models trained by *BatchDTA* framework significantly outperform those trained by *Pointwise* framework in most cases.

Fusing Data Points Measured by Multiple Metrics

Table D6 shows the detailed results corresponding to Figure5. All the models achieve significant improvement when fusing the data points of K_D. Two models, DeepDTA and GraphDTA(GATGCN), gain significant improvement when fusing the data points of IC_{50}, but no model can get further improvement when fusing the data points of EC_{50}. These results indicate that the data points with higher correlations play a more important role for data argumentation.

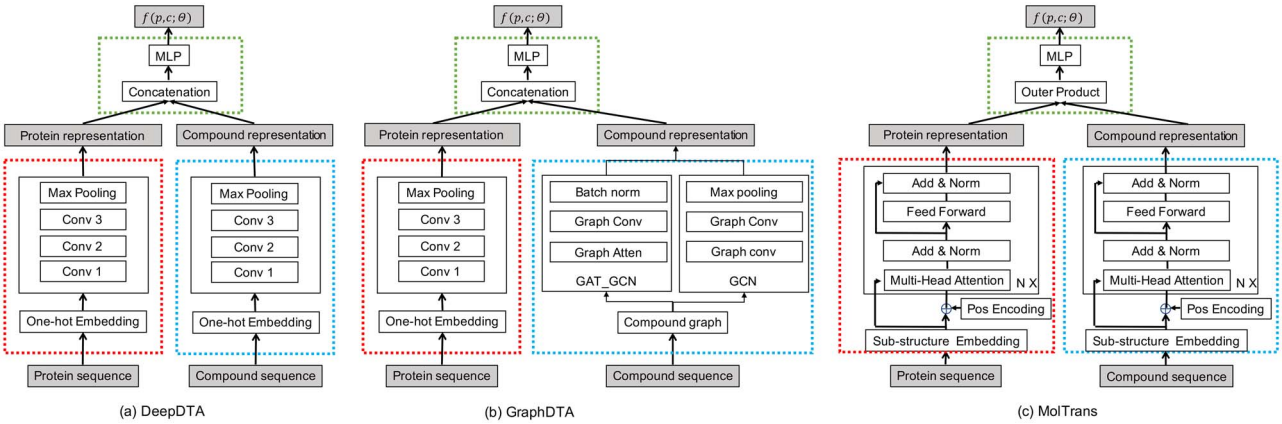


Figure C10. Model architectures of DeepDTA, GraphDTA and MolTrans (red block: Protein Encoder; blue block: Compound Encoder; green block: Interaction Estimator)

Table D5. The detailed results corresponding to Figure 3

Evaluated dataset	K_I			
Evaluated compound pairs	All compound pairs		Confident compound pairs	
Training framework	*Pointwise*	*BatchDTA*	*Pointwise*	*BatchDTA*
DeepDTA	0.6127 (±0.0053)	**0.6252 (±0.0021)**∗	0.6566 (±0.0077)	**0.6775 (±0.0028)**∗
GraphDTA(GCN)	0.6174 (±0.0063)	**0.6347 (±0.0021)**∗	0.6472 (±0.0264)	**0.6929 (±0.0029)**∗
GraphDTA(GATGCN)	0.6168 (±0.0042)	**0.6298 (±0.0036)**∗	0.6600 (±0.0307)	**0.6857 (±0.0053)**
MolTrans	0.6185 (±0.0019)	**0.6559 (±0.0026)**∗	0.6672 (±0.0039)	**0.7204 (±0.0047)**∗
Evaluated dataset	IC_{50}			
Evaluated compound pairs	All compound pairs		Confident compound pairs	
Training framework	*Pointwise*	*BatchDTA*	*Pointwise*	*BatchDTA*
DeepDTA	0.5608 (±0.0042)	**0.5748 (±0.0009)**∗	0.5868 (±0.0081)	**0.5945 (±0.0038)**
GraphDTA(GCN)	0.5527 (±0.0164)	**0.5787 (±0.0002)**∗	0.5606 (±0.0281)	**0.6149 (±0.0004)**∗
GraphDTA(GATGCN)	0.5440 (±0.0090)	**0.5749 (±0.0016)**∗	0.5436 (±0.0379)	**0.6116 (±0.0020)**∗
MolTrans	0.5623 (±0.0037)	**0.5995 (±0.0043)**∗	0.5902 (±0.0071)	**0.6344 (±0.0049)**∗

Note: The standard deviations (std) are given in parenthesis. ∗ stands for the corresponding value is significantly higher than the value on its left (*P*-value $P<0.05$) under the T-test with the two-tailed distribution.

Table D6. The detailed results corresponding to Figure 5

Training framework	**BatchDTA**			
Training dataset	K_I	K_I&K_D	K_I&K_D&IC_{50}	K_I&K_D&IC_{50}&EC_{50}
DeepDTA	0.6252 (±0.0021)	0.6317 (±0.0021)∗	**0.6368 (±0.0005)**∗	0.6350 (±0.0003)
GraphDTA(GCN)	0.6347 (±0.0021)	**0.6507 (±0.0003)**∗	0.6483 (±0.0019)	0.6488 (±0.0002)
GraphDTA(GATGCN)	0.6298 (±0.0036)	0.6346 (±0.0010)∗	**0.6377 (±0.0065)**∗	0.6363 (±0.0006)
MolTrans	0.6559 (±0.0026)	**0.6595 (±0.0004)**∗	0.6558 (±0.0016)	0.6550 (±0.0015)

Note: The standard deviations (std) are given in parenthesis. ∗ stands for the corresponding value is significantly higher than the value on its left (*P*-value $P<0.05$) under the T-test with the two-tailed distribution.

Fusing Data Points from Multiple Datasets

Table D7 shows the detailed results corresponding to Figure 6. As we expected, the CIs of the DNN models trained by *BatchDTA* framework significantly surpass those trained by *Pointwise* framework in most cases, consistent with the results in the BindingDB database (refer to Figure 3). Besides, training on the fusing data points from Davis and KIBA allows the DNN models to achieve better performance than training on a single dataset. Especially when evaluating the models on the KIBA dataset, three models gain significant improvement. Since the KIBA dataset contains only 200 batches, we suspect that combining 400 batches from the Davis dataset enhances the richness of batches.

Table D7. The detailed results corresponding to Figure 6

Evaluated dataset	**Davis**		
Training framework	Pointwise	BatchDTA	
Training dataset	Davis	Davis	Davis+KIBA
DeepDTA	0.7966 (±0.0074)	0.8028 (±0.0111)	**0.8058 (±0.0085)**
GraphDTA(GCN)	0.7906 (±0.0057)	0.7995 (±0.0044)∗	**0.8046 (±0.0013)**
GraphDTA(GATGCN)	0.7917 (±0.0034)	0.8023 (±0.0009)∗	**0.8038 (±0.0042)**
MolTrans	0.8047 (±0.0083)	0.8079 (±0.0037)	**0.8082 (±0.0054)**
Evaluated dataset	KIBA		
Training framework	Pointwise	BatchDTA	
Training dataset	KIBA	KIBA	KIBA+Davis
DeepDTA	0.7317 (±0.0130)	0.7516 (±0.0138)∗	**0.7631 (±0.0051)**∗
GraphDTA(GCN)	0.7210 (±0.0343)	0.7519 (±0.0052)∗	**0.7568 (±0.0042)**∗
GraphDTA(GATGCN)	0.7140 (±0.0210)	0.7520 (±0.0059)∗	**0.7561 (±0.0013)**
MolTrans	0.7392 (±0.0066)	0.7588 (±0.0115)∗	**0.7733 (±0.0060)**∗

Note: The standard deviations (std) are given in parenthesis. ∗ stands for the corresponding value is significantly higher than the value on its left (*P*-value $P < 0.05$) under the T-test with the two-tailed distribution.

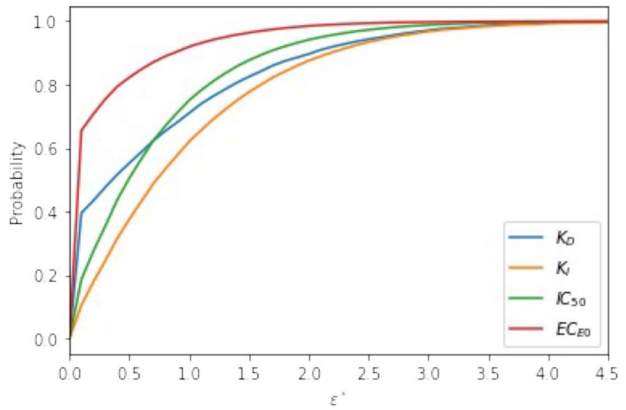


Figure C11. Probability of the compound pairs with affinity gaps smaller than ϵ^*.