Supplementary matrial for VISAR: an interactive tool for dissecting chemical features learned by deep neural network QSAR models

Qingyang, Ding; Siyu Hou; Songpeng, Zu; Yonghui Zhang; Shao Li

March 29, 2020

Contents

S1 Supplementary Methods

S1.1 Datasets

The compound-protein interactions (CPIs) were extracted from the ChEMBL database Version 23 [1]. ChEMBL is one of the biggest open databases of drug-like bioactive compounds, with the data manually or automatically annotated from a large number of scientific literature, presenting in a unified format and confidence scores of the data quality. We filtered the dataset for "target type" as "SINGLE PROTEIN", "organism" as "Homo sapiens", "assay type" as "B" (for binding), "standard type" as "Ki" or "IC50", and "confidence score" larger than 7 (with 9 the highest, 7 means the data is reliable).

The CPI datasets of family A GPCRs and protein kinases were used in this study. Both of them are classic drug targets with many previous pharmacological studies, sufficiently large CPI recording entries with targets testing against structural diverse compounds, making it possible for us to evaluate and interpret the trained models by referring to related chemoinformatics, medicinal chemistry, and structural biology works. The protein classification table of ChEMBL was used for the selection of the single protein targets belonging to "small molecule receptor (family A GPCR)" and "Protein Kinases" on level 3, corresponding to the subclass of family A GPCR and kinases respectively. Dataset of size less than 100 were dropped, leaving 54 GPCRs with more than 70000 CPIs and 29 kinases with more than 7000 CPIs as the training and validation dataset used in this study (See the MT assay table for the detailed summary of datasets).

In the case of CPIs with multiple recordings (same target-ligand pair tested in different experiment batches), a further filtering was applied – if the standard deviation of the affinities in multiple recording was bigger than the cutoff (in this study we set it to 0.5), these recordings were considered inconsistent and discarded, while for the consistent ones, the mean of them was used to represent the CPI strength.

Before training QSAR models, the binding affinities of CPIs in ChEMBL were transformed. The negative logarithm of the original values (unit nM) times $1e^{-9}$ were lower clip to 4 and then shifted to $-\epsilon$.

To further validate the performance of our model, external datasets from DUDE [2] was also used in our study. DUDE is a famous database of decoys for the benchmark of virtual screening studies, and in this study, our trained models were tested by 4 datasets for GPCRs and 11 datasets for kinases extracted from DUDE, with half of the compounds in testing sets as decoys.

S1.2 Compounds and Descriptors

To obtain valid and clean compound data for model training, compounds with the molecular weight larger than 1000 were filtered out. Salt removal and charge neutralization were also carried out for each compound. The Deepchem package [3], an open-source toolchain for deep-learning in drug discovery, was adopted for dataset management and feature extraction. The featurizer 'circular 2048' and 'circular 1024' of Deepchem were used as the descriptors of compounds for both baseline models and neural network models. The open-source software RDKit, one of the chemoinformatics libraries and toolkits [4], was then used for the interpretation of circular fingerprints (built by applying Morgan algorithm thus was called Morgan fingerprints [5] in RDKit). Circular 2048 and Circular 1024 corresponds to radius 2 and 1 for Morgan fingerprint settings respectively. RDKit.Chem.Draw functions were also applied for the compound visualization. PandaTools along with the descriptor mapping function of RDKit was applied to calculate the physiological properties of the compounds.

S1.3 Model

Suppose we have J proteins and for each protein j, $j = 1, 2, \ldots, J$, we have the data $D_j = \{\mathbf{X}_j, \mathbf{y}_j\}$, $\mathbf{X}_j \in \mathcal{R}^{n_j \times p}$, and $\mathbf{y}_j \in \mathcal{R}^{n_j \times 1}$. Each row of $\mathbf{X}_j$ represents one compound with the chemical fingerprint features of p dimensions. y_{ji} records transformed value of the binding affinity (unit as nM) of the compound i against the protein j, $i = 1, 2, \ldots, n_j$.

S1.3.1 Ridge regression model optimized by cross-validation

For each protein j, the ridge model coefficients minimize a penalized residual sum of squares, and leave-one-out cross-validation was applied to determine the alpha for each model:

$$\min \|X_j\omega_j - y\|^2 + \alpha_j\|\omega_j\|^2 \tag{1}$$

In this work, we applied RidgeCV from sklearn [6] linear model package, with 20 α to try, starting from -1 and ending at 2.

S1.3.2 Support vector regression

Linear support vector regression(SVR) was also used to train quantitative predictive models for each protein j respectively. Given $X_j \in \mathcal{R}^{N_j \times p}$ as the fingerprint of ligands that tested against the protein j, and y_j as the corresponding $log(IC_{50})$ value, we have linear function in the form of:

$$y_j = X_j\omega_j + b \tag{2}$$

with $\omega \in \chi, b \in \mathcal{R}$, where χ denotes the space of the input space patterns. Thus it equals to the convex optimization problem:

$$\text{minimize } \frac{1}{2}\|\omega\|^2 + \mathbf{C}\sum_{i=1}^{N_j}(\xi_i + \xi_i^*) \tag{3}$$

$$\text{subject to } \begin{cases} y_j^i - X_j^i\omega_j - b \leq \epsilon + \xi_i \\ X_j^i\omega_j + b - y_j^i \leq \epsilon\xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{4}$$

In this work, we applied LinearSVR from sklearn.svm package, with $\mathbf{C} = 1.0, \epsilon = 0.2$.

S1.3.3 Single task neural network

In this study we use the feed-forward (artificial) neural network (ANN) as the basis of our approach. For the input descriptors $X_j = [x_1, \ldots, x_{N_j}]^T$, they are fed to neurons associated with weights $W^{[1]}$ of layer 1, a bias term b and activation function $f(z)$. After goes through a neuron, the output is:

$$A_j^{[1]} = f(W^{[1]}X_j + b^{[1]}) \tag{5}$$

Then for subsequent layers:

$$A_j^{[l]} = f(W^{[l]}A_j^{[l-1]} + b^{[l]}) \tag{6}$$

The rectified linear unit (ReLU) function was used as the activation function in this study. The typical cost function of root mean squared error (RMSE) was the training object. A series of hyperparameters regarding the network architecture and training strategies were tested, including:

- the number of hidden layers
- the number of neurons in each hidden layer

- the percentage of neurons to drop-out during training

Each dataset for a specific target was trained separately using back-propagation. The model was implemented by Tensorflow [7], Keras [8] and DeepChem package. The Adam optimizer was applied with standard settings, the default learning rate was set to 0.001, and the default size of each batch was 128. HyperparamOpt function of DeepChem package was used for hyperparameter screening.

S1.3.4 Multitask neural network

Compared to the single neural network, multitask models are trained for multiple objects simultaneously. The architecture of the multitask model could be diverse [9], and the selections of the objects could also be delicate in order to balance the data sparsity and useful information for sharing.

Here we adopted a straight forward implementation of multitask neural network, appending the logP value and measurement for compound structure complexity (the descriptor BertzCT in RDKit) to the last layer as additional output nodes, and keeping the rest of the model architecture the same. For this model architecture, there's no missing value for all objects in the dataset. This implementation was only meant for illustration of activity landscape analysis, rather than optimizing predictive performance.

Additionally, the hybrid bypass architecture proposed by Ransundar et al. [9] was also adopted, referred to as the 'RobustMT' training mode.

S1.4 Model evaluation

When evaluating the model performance, we randomly partition the data into 100 samples validation set and the rest as training set. For quantitative prediction, we applied the root-mean-square error (RMSE) to compare the predicted results to real data.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y_i}^2)}{n}} \tag{7}$$

In which, n was the test sample number, y_i was the true value of the sample i, and $\hat{y_i}$ was the predicted value. We also use Pearson correlation as evaluation endpoint:

$$\rho_{Y,\hat{Y}} = \frac{\mathbf{cov}(Y, \hat{Y})}{\sigma_Y \sigma_{\hat{Y}}} \tag{8}$$

Where Y is the actual value of y as vector, and $\hat{Y}$ is the predicted one.

For external dataset performance evaluation, the area under the receiver operating characteristic curve (AUC) was used as the evaluation metrics, since the DUDE dataset doesn't have quantitative binding values but instead a label of active or decoy.

Data processing was carried out in python, while figure drawn using R package ggplot2 [10]. The RMSE, pearson correlation, and AUC calculation was done by sklearn.metrics package.

S1.5 Visualization of SAR

Transformed chemical features (i.e. the output of a specified layer in DNN model) were calculated, and the value dimensions were further reduced using PCA (down to 20) and then tSNE, thus we can visualize the activity landscape on 2D plots. The calculation of PCA and tSNE were also done by sklearn packages.

For the SAR visualization of each compound, instead of simply listing substructures with significant large weights, VISAR uses the derivative of loss function for each fingerprint bit as the weight, which represents the 'importance' of a feature for the corresponding assay [11].

More specifically, in order to obtain the assay-specific weights $L^c_{atom} \in R^{u \times v}$ in general architectures, we firstly compute the gradient of y_c with respect to fingerprint bits x_k:

$$\alpha^c_k = \frac{\partial y_c}{\partial x_k}, k = 1, 2, ..., p, c = 1, 2, ..., J \tag{9}$$

This weight α^c_k represents a partial linearization of the deep network downstream from the fingerprints, and captures the 'importance' of the fingerprint for a target class. Since each atom may be involved in multiple fingerprint bits, a linear combination of all the related weights for an atom is carried out as the final SAR coefficient L^c_{atom} for the atom:

$$L^c_{atom} = \sum_k I(k)\alpha^c_k \tag{10}$$

where $I(k)$ is 1 if the atom is involved in the fingerprint bit k, else 0.

SAR coefficients are min-max normalized between 0 and 1 for visualization. By applying a color map to the SAR coefficients, we could visualize color-coded atom contributions directly on each atom with the help of RDKit drawing functions and bokeh package [12].

S1.6 Pharamcophore building

The pharmacophore-based tool align-it [13] was applied for our pharmacophore related analysis. The open-source tool of align-it is capable of representing pharmacophoric features of the compounds as Gaussian 3D volumes. The analysis involves several steps as follows:

- generate the 3D conformation of the compounds with RDkit using MMFF94 force field.
- extract pharmacophoric features of the compounds using align-it. The features include aromatic and lipophilic, hydrogen bond donor and acceptor, as well as charge centers.
- align previously generated pharmocophores and calculate similarities also using align-it. The similarity was calculated based on the volume of overlap between pharmacophores, and TANIMOTO measure was adopted,

$$\text{TANIMO} = \frac{V_O}{V_A + V_B - V_O} \tag{11}$$

S2 Supplementary Results

S2.1 The rationale of empowering CPI analysis with a deeper understanding of neural network

Firstly we argued that CPI prediction is not just about the predictive power, but also to obtain a better understanding of the 'binding mode' between ligands and their targets – for the latter, deep neural network model with circular fingerprints could be practical and yet helpful.

As showed in Fig S1A, the 4 ways of molecule representation – the circular fingerprints, pharmacophores, 3D molecular conformation in the binding pocket and the transformed features of intermediate neural network layer – depicted the 'binding mode' from different perspectives. Morgan fingerprint (Fig S1a) is one of the most widely used molecular descriptors in virtual screening. Through iterative algorithms, local substructures of compounds

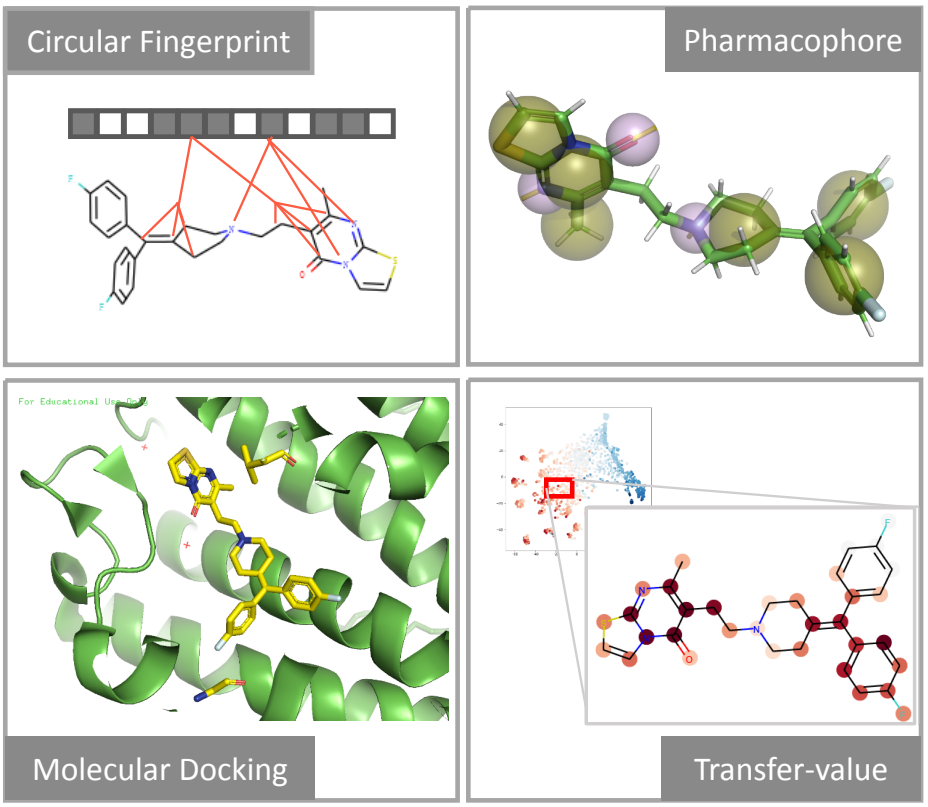

Figure S1: Illustration of different perspectives in understanding the binding mode. Ritanserin, which is a 5HT2c specific inhibitor, served as a representative of the compounds. Fig1a is a schematic diagram of the substructure embedded as Morgan Fingerprint bits. Fig1b is the pharmacophore, with yellow and pink spheres representing aromatic and lipophilic sites and hydrogen bond acceptors respectively. Fig1c is the stucture of 5HT2c - ritanserin complex (6BQH solved by Peng et al. [14]). The interactions between RIT and $G218^{5.42}$ and $V354^{7.39}$ were identified as key for selectivity. Fig1d is an example of the activity landscape and the zoomed-in SAR of ritanserin regarding a specific target. The red color in landscape indicates higher binding affinity while blue color lower activity; the red box on the landscape shows the position of ritanserin; the dark red color mapping on the chemical structure is related to stronger contribution to the binding affinity of ritanserin.

are encoded and folded to bit positions of the fixed length of 1024 or 2048. Pharmacophores (Fig S1b) are defined as steric and electronic features with a specific geometry that is known as important tools in de novo drug design [15]. In the framework of align-it, each pharmacophore is modeled as a 3D spherical Gaussian volume (with center position, the spread and the information on geometry orientation). For 3D molecular conformation in the binding pocket (Fig S1c), the interaction surface of target binding pocket and ligands were showed, allowing people to identify the specific residue and functional group that contribute to the binding. These types of features are not always available for any compound or target of interest, and embedded with the assumption that the crystal structures of the target are the same as that of the physiological states. While pharmacophore and molecular docking manage to provide rigid 3D shape description inferred from target structure and part of the active ligands, circular fingerprints as binary vectors are more suitable as the input for similarity calculation and predictive model without further limitation.

As illustrated in Fig S1d, the trained QSAR models are dissected on two levels: Firstly on the macro-level, the transformed features out of intermediate neural network (i.e., a given DNN layer output) are reduced to 2 dimensions to visualize the distribution of all compounds in the training dataset (the applicability domain). As showed in the upper left figure, compounds in the dataset appear to be arranged according to their binding affinity and structure similarity simultaneously, forming a smoothed activity landscape based on the engineered features. Through building the landscape, we could view clusters of active compounds and grasp the complexity of the already tested activity space against the target of interest from a global perspective.

On the micro-level, we zoom in for a specific compound and calculate the gradient of each fingerprint and map the gradient back to each atom. Since for Morgan fingerprint, the same bit might refer to more than one substructures due to the hashing mechanism [16], the mapping may include noises but sufficient in serving the purpose of visualizing SAR and indicating key functional groups. In the lower-right figure of ritanserin, one of the 4-fluorophenyl and thiazolopyrimidine were labeled as positively contributing to the binding, which is consistent with the finding in structural studies showed in Fig S1d. This kind of SAR didn't involve any information from the target, yet succeeded in narrowing down the key functional groups participating in the interaction.

In summary, we showed in this section that Morgan fingerprint, pharmacophore and 3D molecular conformation in binding pocket are in fact different ways of depicting the binding between compounds and targets of interest. Transformed-feature based activity landscape and visualized SAR could serve as bridges that translate what have been learned from predictive neural network models into human interpretable insights.

S2.2 The model performances

Next, we set out to build neural network models and showed that they have plausible predictive power and generalization ability. Since as indicated by previous work, fairly shallow neural network structure would be rather sufficient for single task QSAR model [17], here we took a deep look at only the 2 or 3-layer model and focussed specifically on various pyramid architectures. Based on hyperparameter screening (Fig S2 and Table S1), we chose the neural network model architecture for the subsequent training and analysis.

For the next step a series of CPI datasets extracted from CHEMBL targeting proteins from GPCR family A and kinases were trained, and the performance of these models was compared to baseline methods support vector regression (SVR) and ridge regression (Rig). Figure S2 shows that our single task (ST) and multi-task (MT) neural network models were performing marginally better than SVR models and compatible with Ridge models on most of the test sets. The results are summarised in Table S3. Further external validation was carried out using DUDE datasets and pre-trained predictive models for kinases. The

Table S1: Summary of dataset information

Task	sample_size	full_name	pref_name
T107	2951	Serotonin 2a (5-HT2a) receptor	5-HT2a
T108	2063	Serotonin 2c (5-HT2c) receptor	5-HT2c
T51	3202	Serotonin 1a (5-HT1a) receptor	5-HT1a
T106	808	Serotonin 1b (5-HT1b) receptor	5-HT1b
T105	883	Serotonin 1d (5-HT1d) receptor	5-HT1d
T10618	109	Serotonin 1e (5-HT1e) receptor	5-HT1e
T227	1064	Serotonin 2b (5-HT2b) receptor	5-HT2b
T168	396	Serotonin 4 (5-HT4) receptor	5-HT4
T10624	313	Serotonin 5a (5-HT5a) receptor	5-HT5a
T10627	2523	Serotonin 6 (5-HT6) receptor	5-HT6
T10209	1560	Serotonin 7 (5-HT7) receptor	5-HT7
T8	671	Tyrosine-protein kinase ABL	ABL
T9	201	Epidermal growth factor receptor erbB1	EGFR
T10188	249	MAP kinase p38 alpha	MK14
T10434	383	Tyrosine-protein kinase SRC	SRC
T10980	240	Vascular endothelial growth factor receptor 2	VEGFR2
T11408	117	c-Jun N-terminal kinase 3	MK10
T11451	356	Hepatocyte growth factor receptor	MET
T11636	262	Protein kinase C beta	KPCB
T11638	139	MAP kinase ERK2	ERK2
T10938	385	Tyrosine-protein kinase JAK2	JAK2
T11678	289	Cyclin-dependent kinase 2	CDK2

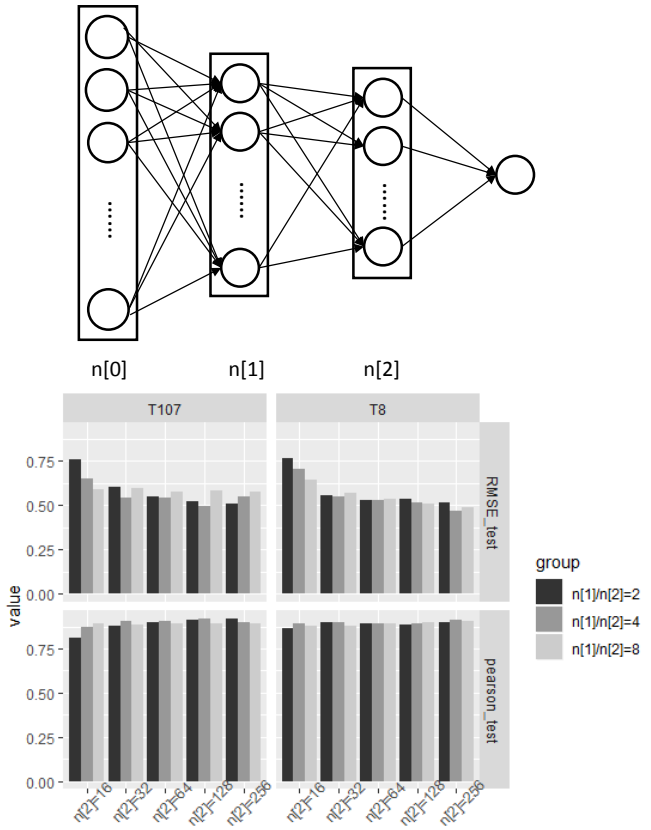

Figure S2: Structure of the neural network model and the test of structural robustness.

Table S2: Parameters for neural network model training

Task	**layers**	**dropout ratio**
T107	(512,64)	0.4
T108	(512,128)	0.2
T10209	(512,64)	0.4
T105	(512,128)	0.2
T106	(512,64)	0.4
T10618	(512,128)	0.4
T10624	(512,128)	0.2
T10627	(512,64)	0.2
T168	(512,128)	0.2
T227	(512,64)	0.4
T51	(512,128,64)	0.2

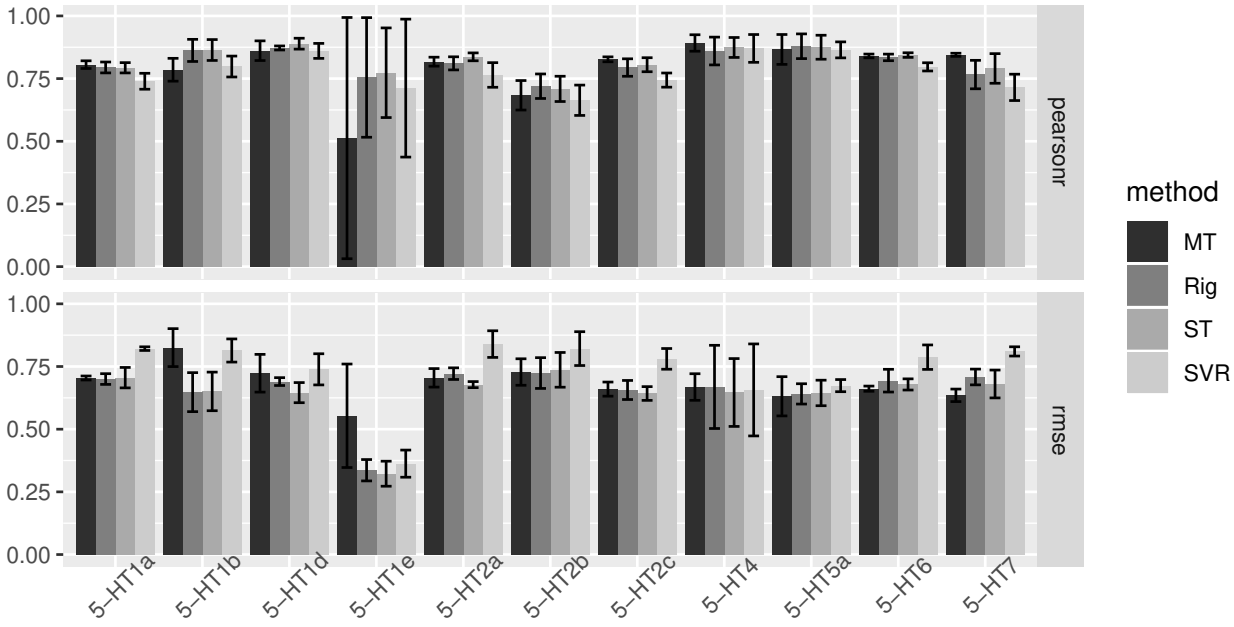

Figure S3: Predictive performance of the neural network model compared with baseline methods.

Table S3: Performance on internal dataset of serotonin receptors

Task	Target	pearsonr				rmse			
		MT	ST	SVR	Rig	MT	ST	SVR	Rig
T51	5-HT1a	**0.81**	0.79	0.74	0.79	**0.70**	0.71	0.82	**0.70**
T106	5-HT1b	0.79	**0.86**	0.80	**0.86**	0.83	**0.65**	0.81	**0.65**
T105	5-HT1d	0.86	**0.89**	0.86	0.87	0.72	**0.65**	0.74	0.69
T10618	5-HT1e	0.51	0.77	0.71	**0.75**	0.55	**0.32**	0.36	0.34
T107	5-HT2a	0.82	**0.84**	0.76	0.81	0.70	**0.68**	0.84	0.72
T227	5-HT2b	0.68	0.71	0.66	**0.72**	0.73	0.74	0.82	**0.72**
T108	5-HT2c	**0.83**	0.81	0.74	0.79	0.66	**0.64**	0.78	0.66
T168	5-HT4	**0.89**	0.87	0.87	0.86	0.67	**0.65**	0.66	0.67
T10624	5-HT5a	0.87	**0.88**	0.86	**0.88**	**0.63**	0.64	0.67	0.64
T10627	5-HT6	**0.84**	**0.84**	0.80	0.83	**0.66**	0.68	0.79	0.69
T10209	5-HT7	**0.84**	0.79	0.71	0.77	**0.64**	0.68	0.81	0.71

Table S4: Performance on external dataset of kinases

Task	Assay	ST	SVR	Rig
T8	Tyrosine-protein kinase ABL	0.63	**0.66**	0.63
T11678	Cyclin-dependent kinase 2	**0.67**	0.66	0.63
T9	Epidermal growth factor receptor erbB1	0.90	0.90	**0.91**
T10938	Tyrosine-protein kinase JAK2	**0.87**	0.77	0.75
T11636	Protein kinase C beta	**0.63**	0.54	0.61
T11451	Hepatocyte growth factor receptor	**0.84**	0.74	0.80
T11638	MAP kinase ERK2	**0.93**	0.90	0.87
T11408	c-Jun N-terminal kinase 3	**0.82**	**0.82**	0.73
T10188	MAP kinase p38 alpha	**0.85**	0.78	0.80
T10434	Tyrosine-protein kinase SRC	**0.89**	0.83	**0.89**
T10980	Vascular endothelial growth factor receptor 2	**0.82**	0.65	0.59

resulting AUC scores for single task neural network models, SVR and Ridge are listed in Table S4. Here we found that ST models had the highest AUC score for most external datasets, and for those that were not the best, ST models didn't fall behind much. This suggests a fair generalization ability of our neural network models.

Similarly with the findings reported by Liu et al. [18], the predictive power of neural networks didn't appear to be far better than baseline models. Rather, the applicability domain, as well as the limitation of performance, seems to be set by the training datasets. We argued that though it's probably true that most predictive models overfit and 'reward memorization rather than generation' [19], it's still worth the effort if we could take the full advantage the 'memory' instead of just shooting for better metrics. And that's the reason why we need the strategy to dissect the learned features from QSAR models.

S2.3 Neural network gradient-based visualization of SAR showed consistency with key binding sites validated by structural studies

As illustrated in S2.1, the color-coded atom contribution pattern is one of the helping tools for us to validate what has been learned in neural network models. In Figure S4 we explored three compounds – ergotamine (ERG), ritanserin(RIT) and lysergic acid diethylamide (LSD) – targeting a series of serotonin receptors, and showed that the neural network gradient-based atom contribution of these compounds were consistent with the reported structural

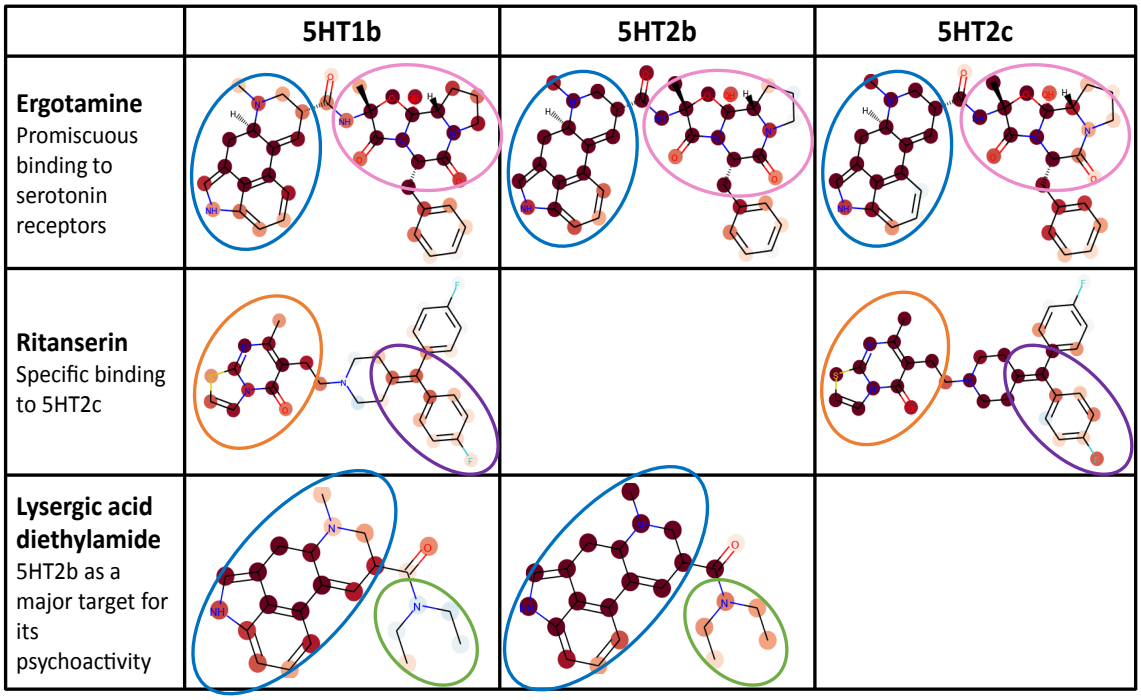

Figure S4: The color-coded atom contributions of ERG, RIT and LSD against serotonin receptors. Blue circle: ergoline core; Pink circle: tripeptide site; Purple circle: 4-fluorophenyl group; Orange circle: thiazolopyrimidine group; Green circle: diethylamide group

core binding sites.

ERG is a promiscuous inhibitor of serotonin receptors, and several structural studies have shown that its binding mode is conserved through 5HT1b, 5HT2b and 5HT2c [14, 20]. The ergoline core of ERG is recognized by nine key residues from protein binding pockets, serving as the key interaction sites; the ergoline structure in our SAR results was identified as a highly positive contribution. As for cyclic tripeptide and benzyl substituents, only non-specific side-chain contacts are identified, allowing them with more flexibility in interactions; in SAR analysis tripeptide site was also labeled as a positive contribution but with differences in detail for 3 targets, and benzyl site is weaker compared with ergoline core and tripeptide sites.

For RIT, the thiazolopyrimidine were marked red for both 5HT1b and 5HT2c, while the aromatic ring, C6 ring and its linker were stronger contributors in 5HT2c binding compared to that of 5HT1b. According to Peng et. al., RIT's 4-fluorophenyl and thiazolopyrimidine interacting with respective residues are indeed the primary reason for RIT's 5-HT2 selectivity, suggesting that our SAR succeeded in finding both the key points in RIT-5HT2 interactions.

Similar to ERG, LSD has the ergoline moiety, and the occupation of the ergoline core in the same orthosteric pocket is a well-established binding site [20]. Additionally, LSD's diethylamide group binds extended binding pockets and its conformation is identified to be important the potency and activity of LSD at 5HT2b receptors [21], while in the SAR of LSD against 5HT2b, diethylamide group stands out as responsible for the binding.

In summary, Figure S4 shows that the atom contributions of the compounds gave a clear and insightful mark of the core structure, and could be validated by related structural studies. Baseline models, on the other hand, failed at picking key binding sites (Figure S6 and S7). Without information from the specific binding pocket, visualized SAR derivated from trained neural network models that ultimately learned from quantitative binding affinities has shown their potential in translating what has been learned from the predictive model to useful understanding of the binding modes.

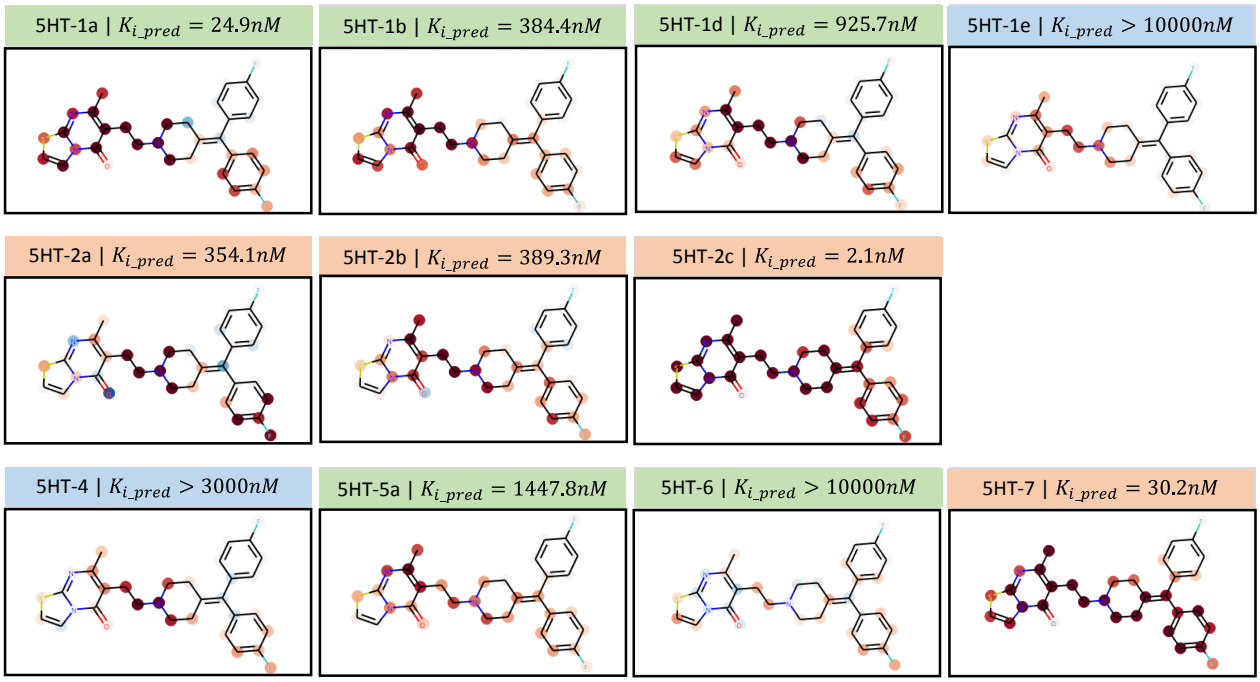

Figure S5: Visualized SAR of RIT based on trained weights of deep neural network.

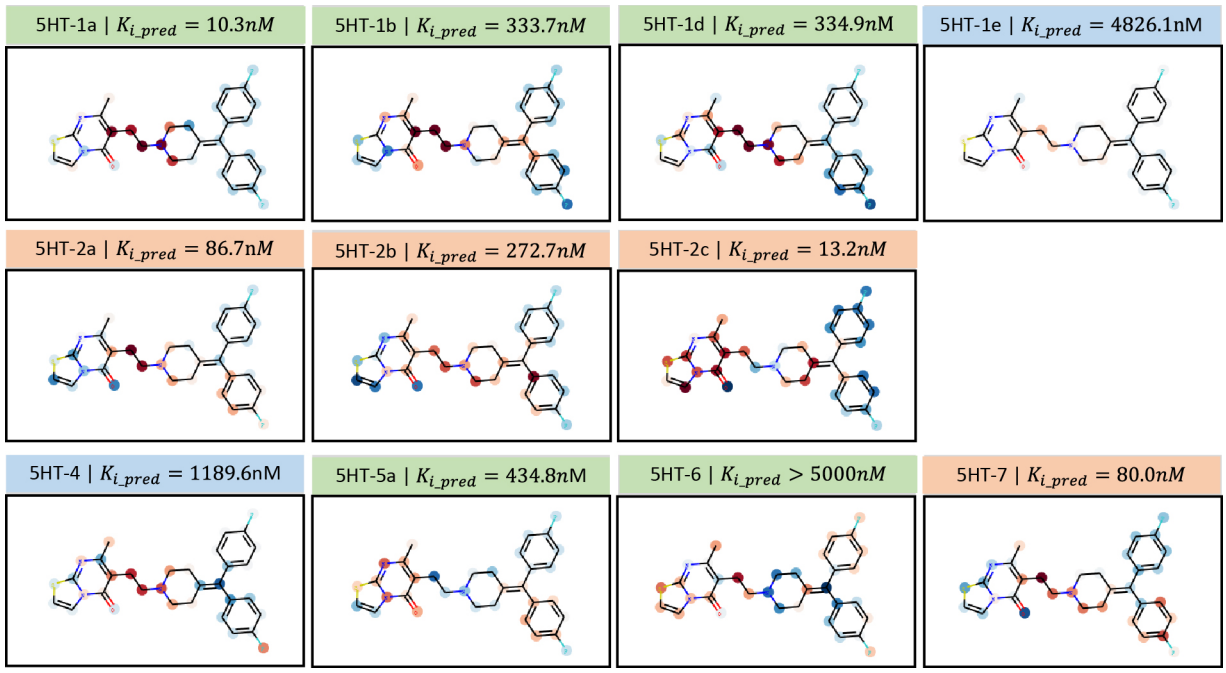

Figure S6: Visualized SAR of RIT based on RidgeCV model coefficients.

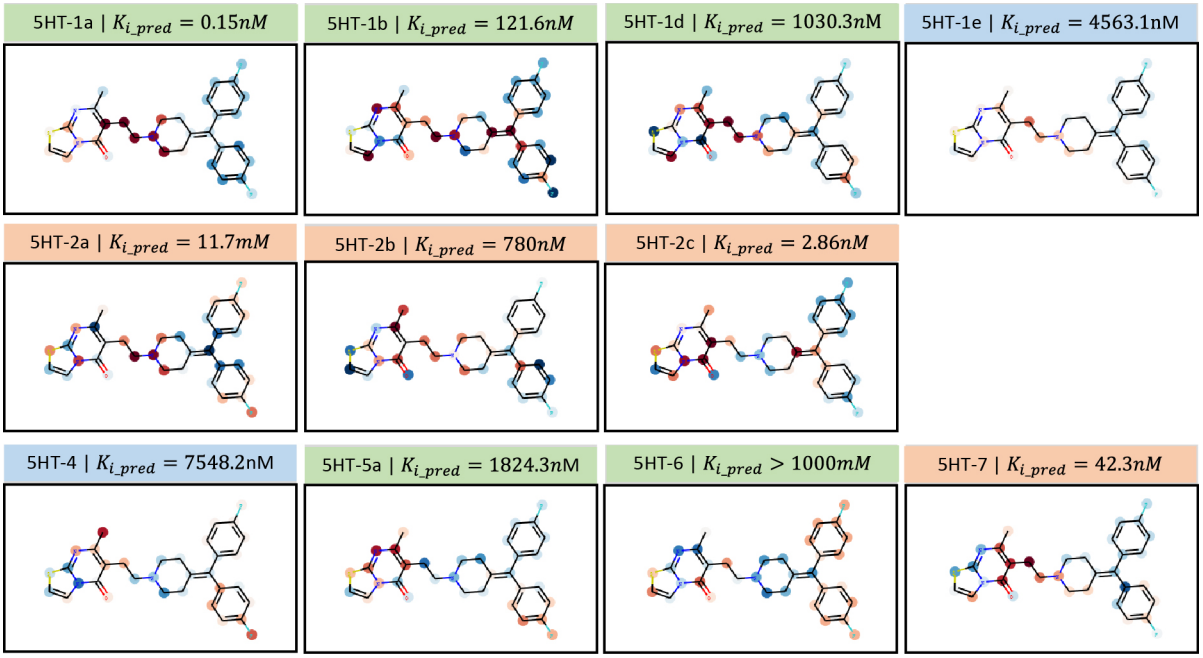

Figure S7: Visualized SAR of RIT based on SVR model coefficients.

S2.4 The activity landscape based on transformed features of the neural network provide better understanding of the training process and active pharmacophore

Next, we seek to illustrate that the activity landscape based on transformed features (i.e. the outputs of a given DNN layer in the model) of the trained QSAR models provide a global perspective of the SAR, and a better understanding of the active pharmacophore. We firstly showed the activity landscape constructed by transformed features of the pre-trained neural network, and then zoomed in and inspect the clusters and the active pharmacophore on the landscape.

The formation of a smoothed activity landscape resembles a semi-supervised clustering process, but with quantitative data points as the label to learn from, more detailed. An example of the activity landscape using transformed features of the neural network model (Fig S8a) and Morgan Fingerprint (Fig S8b) trained for target 5HT-2a was showed, with pink shades of the points indicating strong binding affinity and light blue shades weak binding affinity. Since for the Morgan Fingerprint landscape, the clustering of compounds was purely dependent on structure similarity, clusters may be mixed with both active and inactive compounds of similar structure (forming 'activity cliffs'), and the clusters are far apart from each other.

As the learning steps increases, the similarities between compounds would no longer depend only on the original fingerprint but on weighted new features, and the distance between the compounds was re-defined based on engineered features learned from the quantitative predictive model. Therefore the landscape would be smoothed where the chemical features correlate better with the biological activity [22] (showcased in Fig S9). The exact same cluster on the Morgan Fingerprint landscape (labeled as blue points in Fig S8a) was split into two on transformed-feature landscape, one with higher binding affinity, another one branching towards other direction.

The distribution of compounds is schematically showed in Fig S8c and S8d, with blue color representing low binding affinity clusters of compounds, and pink/orange/red color correlated with stronger binding affinity. While on the Morgan Fingerprint landscape the clusters were more independent of each other (discretely arranged after dimension reduction by tSNE), on transformed-feature landscape compounds were arranged according to their

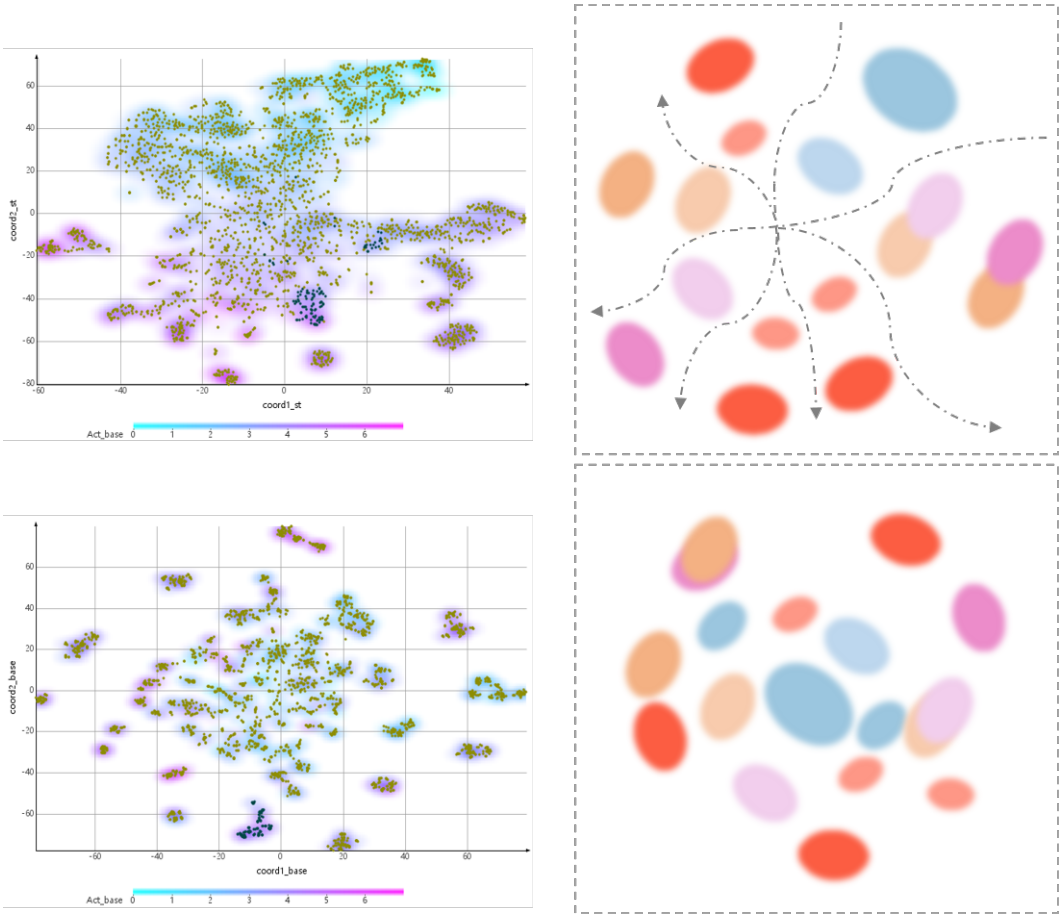

Figure S8: Example of activity landscapes of 5HT-2a. For scatter plots on the left, pink shading indicate high binding affinity, while cyan shadings indicate low binding affinity. The highlighted blue dots on the two figures are the same group of compounds. Schematic plots on the right correspond to transformed-feature landscape and original Morgan fingerprint landscape respectively, with blue color indicating low binding affinity and other colors indicating higher binding affinity with possibly various binding modes (could be depicted as various kinds of active pharmacophore model).

binding affinity, shaping a smoothed landscape along which structures 'evolved from inactive to active' based on rules learned from neural network models.

The clusters of active compounds could be used to build pharmacophore models in a classic process, with the resulting models are with relative weights on each pharmacophoric feature. Instead of manually picking rules for 'active' [15], now with the landscape and SAR in mind, we know that each pharmacophore model is strictly defined within a local applicability domain, and across the landscape, an accurate number of such model could be identified and used for guiding de novo drug design.

Figure S10 shows two examples of the local pharmacophore model. In the upper panel, two groups of the aromatic and lipophilic sites were identified as common pharmacophoric features of the selected cluster of compounds, but according to SAR visualization of these compounds, we would notice that the two groups were not equal in contributing to the binding. Benzoisothiazol or benzoisoxazol groups (left part of the pharmacophore) together with the piperazine linker structure (hydrogen bond donor in the middle) were labeled as the common positive contributor, while the quinoline structure and other substitute groups (right part of the pharmacophore) were weaker contributors. For the lower panel, a similar pattern exists, with the methyl-pyrazole group along with methoxyphenylurea recognized as responsible for the binding affinity (corresponding to the left part of the pharmacophore). Other functional groups, though commonly seen in this cluster, were not as core as them.

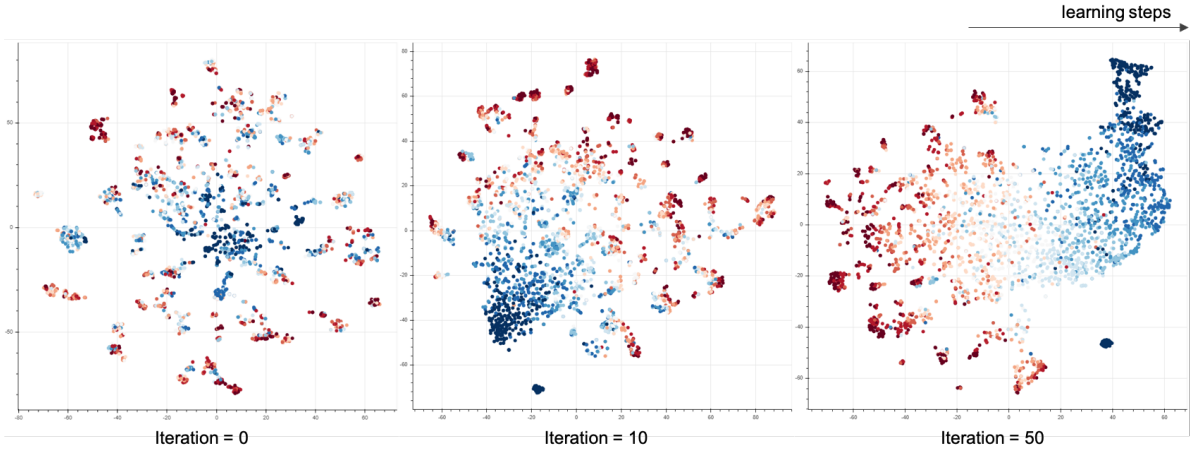

Figure S9: The evolving of the activity landscape along learning iterations.

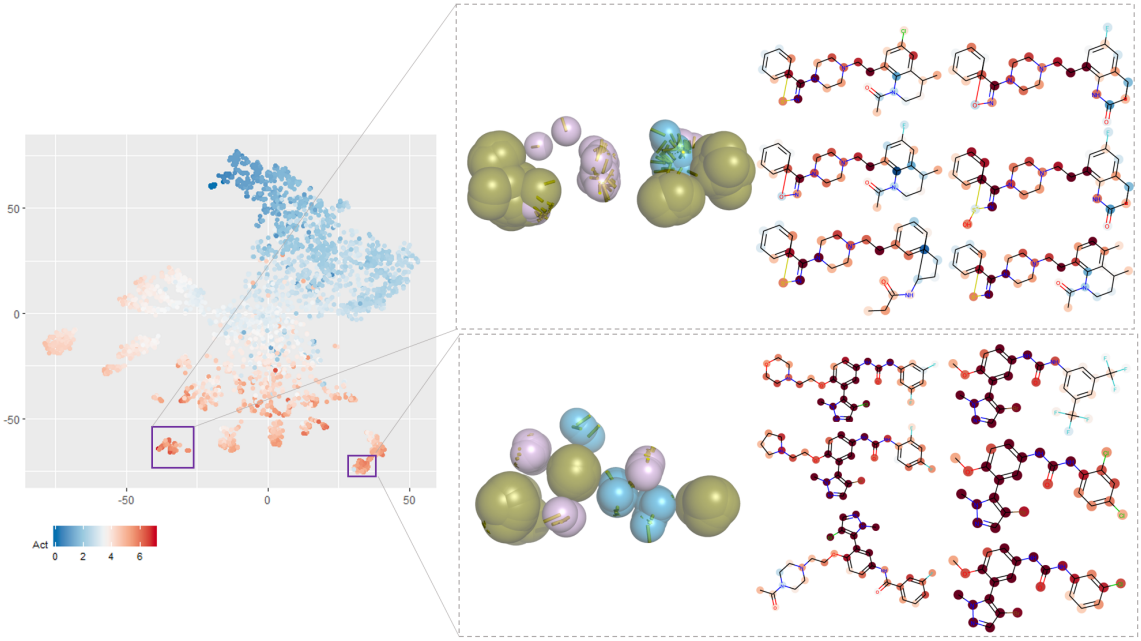

Figure S10: Example of binding modes based on local active area on the activity landscape of 5HT-2a.

These observations provide powerful insights into distinguishing core binding sites and other sites on the compounds that are open to modification.

Since this insight only depended on the quantitative CPI information in CHEMBL database without any information on the protein target, we argued that with relatively small CPI datasets this type of analysis is also applicable, and could intuitively guide the design for selective compounds.

S3 Usage instructions

The source code and usage instructions for VISAR are available on github.

S3.1 Installation

For model training, two environments are optional. Users could install either or both of the two.

- Option1: a working environment with python=3.6 and CUDA 9.0 is recommended, and the environment is depended on Deepchem, Rdkit, Tensorflow, Numpy, Pandas, Sklearn, Scipy.
- Option2: a working environment with python=3.7 and CUDA 10.0 is recommeded, and the environment is depended on pytorch, Rdkit, Tensorflow, Numpy, Pandas, Sklearn, Scipy.

The VISAR python package is available for pip install.

```
## Create an environment for model training
# either deepchem or pytorch is needed, users could use one of them or both
# option1: deepchem
conda create -n deepchem_visar python=3.6
conda activate deepchem_visar
pip install tensorflow-gpu==1.12.0

# option2: pytorch
conda create -n pytorch_visar python=3.7
conda activate pytorch_visar
pip install pytorch==1.3.0

## Install packages
pip install tensorflow-gpu==1.12.0
conda install scipy  # also install six
conda install matplotlib
conda install pandas
conda install seaborn

conda install -c conda-forge scikit-learn  # also install joblib
pip install deepchem==2.1.1.dev353
pip install visar  # also install bokeh

conda install -c rdkit rdkit  # Installs also numpy and pandas

conda install jupyter  # Installs also ipykernel
python -m ipykernel install --user --name deepchem_visar
```

For using VISAR web application in visualising the trained models, the requirements for the environment is fewer, independent of GPU, and could be easily applied on desktops or laptops of either Windows or macOS platforms with chrome browsers. To use the interactive application, users need to firstly get the local copy of the VISAR repository by direct dowloading or

```
git lfs install  # enable large file downloading
git lfs clone https://github.com/Svvord/visar.git
```

The installation steps are as follows:

```
## Create an environment for visualization using VISAR web application
conda create -n visar_viz python=3.6
conda activate visar_viz

## install packages
```

```
conda install -c conda-forge rdkit  # also install numpy and pandas
conda install -c conda-forge scikit-learn  # also install scipy
conda install matplotlib
conda install bokeh
conda install cairosvg
```

and then start the web application by

```
cd /path/of/visar
bokeh serve --show VISAR_webapp
```

S3.2 Model training utilities

VISAR package provides users with preprocessed datasets, which include 3060 biochemical assays originated from CHEMBL database, and more than 100,000 compounds. To train the model, users could setup parameters for model and use hyperparameter screening and training functions provided by VISAR package.

The following code is a demonstration of training a DNN multitask model in pytorch.

```
import os
os.environ['CUDA_VISIBLE_DEVICES']='1'

# step1: model parameter set-up
task_names = ['T8', 'T9']

para_dict_DNN = {
  'model_name': 'VISAR_pytorch_demo',  # user specific
  'task_list': task_names,  # MUST BE A LIST!
  # input data related params:
  'dataset_file': './data/Kinase_tot_4deepchem_processed.csv',
  'feature_type': 'Morgan',
  'id_field': 'molregno',
  'smiles_field': 'cano_smiles',  #
  'add_features': None,
  'frac_train': 0.9,
  'rand_seed': 0,
  'batch_size': 100,
  'normalize': True,
  # model architecture related parameters:
  'layer_nodes': [256, 128, len(task_names)], #
  'dropouts': 0.3,
  # model training related parameters:
  'learning_rate': 0.001,
  'GPU': True,
  'epoch': 100, # training epoch of each round (saving model at the end of
                                        each round)
  'epoch_num': 5, # how many rounds
  'optimizer': 'Adam',
  # viz file processing related parameters:
  'model_architecture': 'RobustMT',
  'hidden_layer': 1
}

# step2: hyperparams screening
# The hyperparam_screening function would return the best parameter dictionary
                                        . Users could also check the
                                        performances recorded in the log file
                                        and manually pick the best param.
from visar.utils.pytorch_functions import hyperparam_screening
```

```
import os
os.chdir('/working/directory/')  # user specified

import copy
from collections import OrderedDict
from visar.dataloader.pytorch_utils import compound_FP_loader
from visar.pytorch_regressor import pytorch_DNN_model

candidate_params_dict = OrderedDict([('layer_nodes', [[256,128,1], [512,64,1],
                                      [512,128,1], [512,265,1]]),
                                ('dropouts', [0.2, 0.4]),
                                ('learning_rate', [0.01, 0.001])])

best_param = hyperparam_screening(pytorch_DNN_model, para_dict_DNN,
                              candidate_params_dict,
                              mode = 'grid_search',
                              epoch = 10, epoch_num = 2)

# step3: model training
from visar.pytorch_regressor import pytorch_DNN_model
from visar.dataloader.pytorch_utils import compound_FP_loader
train_loader, test_loader, train_df, test_df, para_dict_DNN =
                                        compound_FP_loader(para_dict_DNN)

pyDNN_model = pytorch_DNN_model(para_dict_DNN)
pyDNN_model.model_init()
pyDNN_model.model

pyDNN_model.fit(train_loader, test_loader)
```

Users could referred to Multitask model training template and Single task model training template.

Next, for visualization, VISAR provides a function to read trained model files and compose files (also referred to as RUNKEY dataframe files) for web application rendering. Users could also provide a file of compounds they interest in by specifying the name of the file, the column names of compound id, SMILES structure as well as additional properties. VISAR would process the custom file along with the original datasets, mapping custom compounds on the activity landscape of the trained model, and giving activity predictions and SAR plots. By doing this, RUNKEY dataframe files would be generated and subsequently used for the display on the web application by selecting the 'custom data' radio button. The codes for result processing are demonstrated below:

```
# step4: process trained model files for visualization
from visar.dataloader.pytorch_utils import compound_FP_loader
from visar.pytorch_regressor import pytorch_DNN_model

# prepare custom dataloader
custom_para_dict = {
  'task_list': ['activity'],   # a dummy column of float
  # input data related params:
  'dataset_file': './data/binding_mode_notation.csv',
  'feature_type': 'Morgan',
  'id_field': 'cid',   #
  'smiles_field': 'SMILES',  #
  'model_flag':'ST',
  'add_features': None,
  'frac_train': 1,
  'batch_size': 100,
  'normalize': False
}
```

```
custom_loader, custom_df, custom_para_dict = compound_FP_loader(
                                        custom_para_dict)

# load previous model
import json
para_dict_DNN = json.load(open('./logs/VISAR_pytorch_demo/train_parameters.
                                        json','r'))

pyDNN_model = pytorch_DNN_model(para_dict_DNN)
pyDNN_model.model_init()
pyDNN_model.load_model()

# load training data
train_loader, test_loader, train_df, test_df, para_dict_DNN =
                                        compound_FP_loader(para_dict_DNN,
                                max_cutoff = 8000)

pyDNN_model.para_dict['custom_id_field'] = custom_para_dict['id_field']
pyDNN_model.para_dict['custom_smiles_field']=custom_para_dict['smiles_field']
pyDNN_model.para_dict['hidden_layer'] = 2
pyDNN_model.generate_viz_results(train_loader, train_df, 'test_viz_file',
                            custom_loader = custom_loader,
                            custom_df = custom_df)
```

S3.3 Visualization with web application

By default, the web application would use the sample dataframes to render the panel, presenting the interface as Figure S11. The general steps for interactive analysis are:

1. Set the directory (including the prefix) of the pre-composed data frames and the mode of your training. After clicking 'Run' button on the upper panel, the whole interface would update according to your settings.

2. Explore the activity space on the left panel. Several places are allowing for interactive exploring, including: A. color options for the scatter plotting, enabling different color rendering based on different assays; B. number of bi-clusters, which dynamically linked with the arrangement of the heatmap on the bottom panel (through trying out different bi-cluster numbers, users could gain an idea of how the activity profile is distributed on the activity landscape); C. information of the compounds when hovering your mouse on the scatter plot, displaying its ID, batch ID and the color code for the bi-cluster where it belongs; D. information of the batch when hovering your mouse on the heatmap, displaying its ID and color code for the bi-cluster where it belongs.

3. Upon selecting the batch or individual compounds on the left panel, chemical structures along with the color-coded atom contributions are displayed on the right panel. There are two ways for batch selection: one is to directly click on the heatmap, another is to use the drop-down list (E). As for compound selection, use the tap mode of the scatter plot and click on the points. Since for RobustMT mode, multiple tasks are available for the compound; thus by selecting SAR task (F), the atom contributions of the compounds regarding different assays would update accordingly.

4. If a custom file is provided during the generation of RUNKEY dataframe, the predictive results for custom compounds could then be displayed by clicking the 'custom data' radio button. This function would allow the users to quickly explore the SAR of new atoms in their compounds of interests, and the prediction could serve as a reference for the future experiment design (Figure S12).

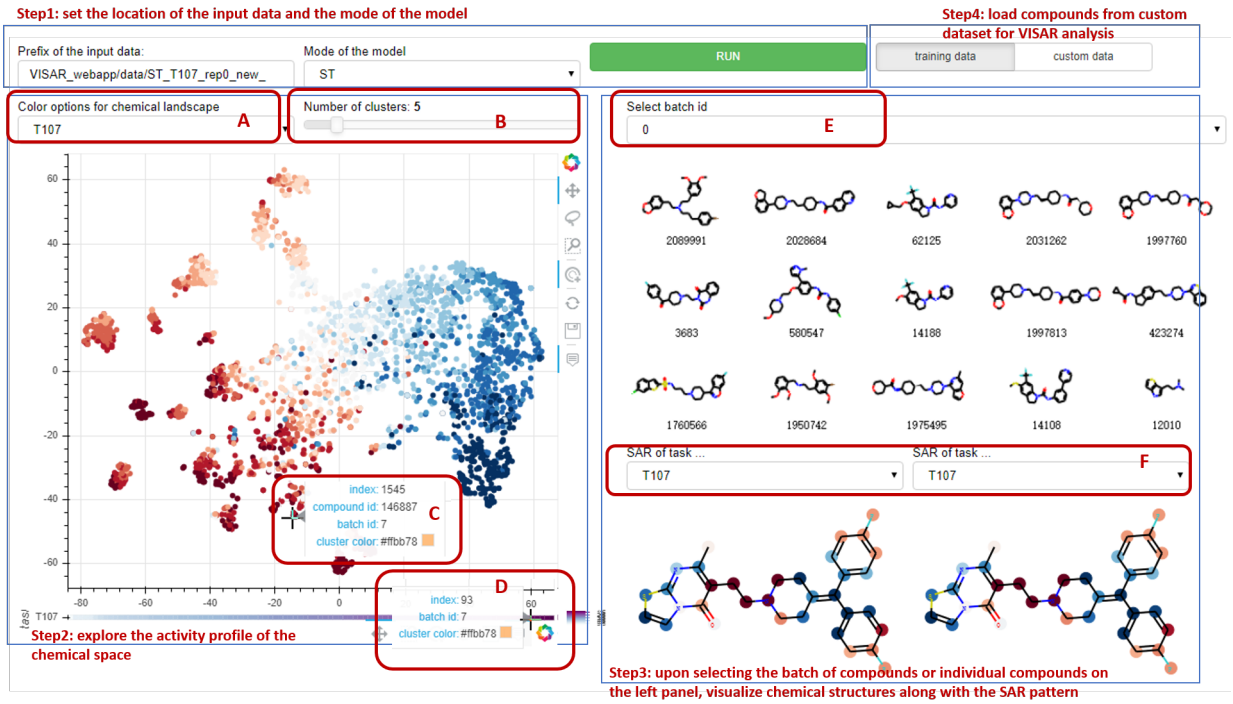

Figure S11: Demonstration of VISAR web application interface.

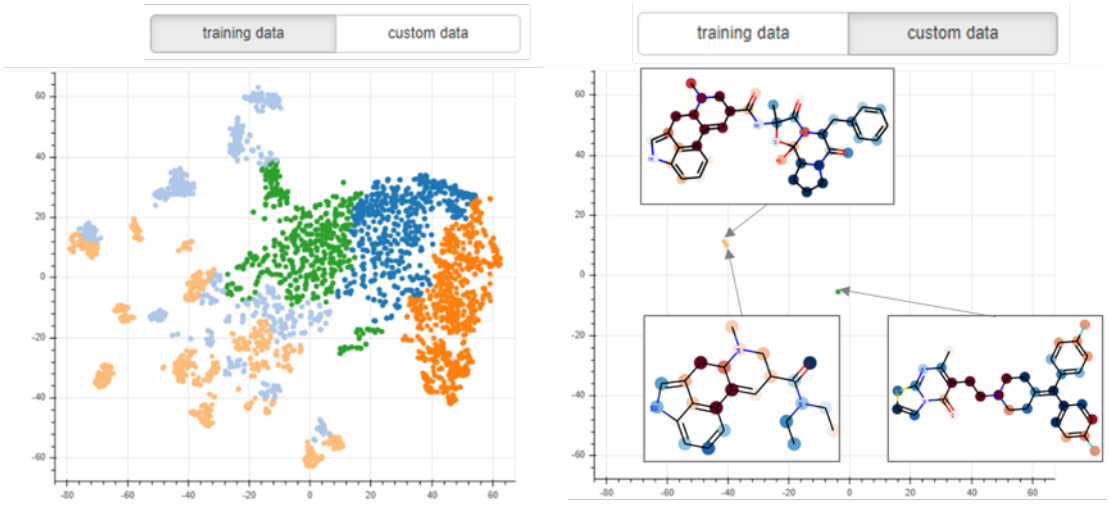

Figure S12: An example showing the activity landscape of the training dataset (left), and the positions of custom compounds after switching for 'custom data' mode (right). The color indicates the clustering label of the compounds, with the same color suggests similar activity; by mapping custom compounds on the landscape of training compounds, we could easily observe that the custom compounds are within the applicability domain. Then the users may click on the custom compounds and see their predicted SAR patterns.

S3.4 Pharmacophore analysis

VISAR provided handy function for converting selected dataframe to SDF files:

```
import os
import numpy as np
import pandas as pd
from visar.utils.visar_utils import df2sdf

compound_df = pd.read_csv('path_to/RUNKEY_compound_df.csv')
selected_batch = 0
select_df = compound_df.loc[compound_df['label'] == selected_batch]

df2sdf(select_df, 'output_prefix_batch0.sdf', selected_batch = 0)
```

With SDF files of compounds of interest, users could adopt other software for further analysis, including pymol, MOE, DataWarrior, Schrodinger, TeachOpenCADD and etc.

References

[1] A. P. Bento, A. Gaulton, A. Hersey, L. J. Bellis, J. Chambers, M. Davies, F. A. Krüger, Y. Light, L. Mak, S. McGlinchey, *et al.*, "The chembl bioactivity database: an update," *Nucleic acids research*, vol. 42, no. D1, pp. D1083–D1090, 2014.

[2] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet, "Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking," *Journal of medicinal chemistry*, vol. 55, no. 14, pp. 6582–6594, 2012.

[3] B. Ramsundar, P. Eastman, P. Walters, V. Pande, K. Leswing, and Z. Wu, *Deep Learning for the Life Sciences*. O'Reilly Media, 2019.

[4] G. Landrum, "Rdkit: Open-source cheminformatics," *Online). http:// www. rdkit. org. Accessed*, vol. 3, no. 04, p. 2012, 2006.

[5] A. Gobbi and D. Poppinger, "Genetic optimization of combinatorial libraries," *Biotechnology and bioengineering*, vol. 61, no. 1, pp. 47–54, 1998.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2015.

[8] F. Chollet, "keras." https://github.com/fchollet/keras, 2015.

[9] B. Ramsundar, B. Liu, Z. Wu, A. Verras, M. Tudor, R. P. Sheridan, and V. Pande, "Is multitask deep learning practical for pharma?," *Journal of chemical information and modeling*, vol. 57, no. 8, pp. 2068–2076, 2017.

[10] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

[11] P. Polishchuk, "Interpretation of quantitative structure–activity relationship models: past, present, and future," *Journal of chemical information and modeling*, vol. 57, no. 11, pp. 2618–2639, 2017.

[12] Bokeh Development Team, *Bokeh: Python library for interactive visualization*, 2019.

[13] J. Taminau, G. Thijs, and H. D. Winter, "Pharao: Pharmacophore alignment and optimization," *Journal of Molecule Graphics and Modelling*, vol. 27, no. 2, pp. 161 – 169, 2008.

[14] Y. Peng, J. D. McCorvy, K. Harpsøe, K. Lansu, S. Yuan, P. Popov, L. Qu, M. Pu, T. Che, L. F. Nikolajsen, *et al.*, "5-ht2c receptor structures reveal the structural basis of gpcr polypharmacology," *Cell*, vol. 172, no. 4, pp. 719–730, 2018.

[15] S.-Y. Yang, "Pharmacophore modeling and applications in drug discovery: challenges and recent advances," *Drug discovery today*, vol. 15, no. 11-12, pp. 444–450, 2010.

[16] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 742–754, 2010.

[17] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl, and V. Svetnik, "Deep neural nets as a method for quantitative structure–activity relationships," *Journal of chemical information and modeling*, vol. 55, no. 2, pp. 263–274, 2015.

[18] R. Liu, H. Wang, K. P. Glover, M. G. Feasel, and A. Wallqvist, "Dissecting machine-learning prediction of molecular activity: Is an applicability domain needed for quantitative structure–activity relationship models based on deep neural networks?," *Journal of Chemical Information and Modeling*, vol. 59, no. 1, pp. 117–126, 2018.

[19] I. Wallach and A. Heifets, "Most ligand-based classification benchmarks reward memorization rather than generalization," *Journal of chemical information and modeling*, vol. 58, no. 5, pp. 916–932, 2018.

[20] C. Wang, Y. Jiang, J. Ma, H. Wu, D. Wacker, V. Katritch, G. W. Han, W. Liu, X.-P. Huang, E. Vardy, *et al.*, "Structural basis for molecular recognition at serotonin receptors," *Science*, vol. 340, no. 6132, pp. 610–614, 2013.

[21] D. Wacker, S. Wang, J. D. McCorvy, R. M. Betz, A. Venkatakrishnan, A. Levit, K. Lansu, Z. L. Schools, T. Che, D. E. Nichols, *et al.*, "Crystal structure of an lsd-bound human serotonin receptor," *Cell*, vol. 168, no. 3, pp. 377–389, 2017.

[22] F. Grisoni, D. Ballabio, R. Todeschini, and V. Consonni, *Molecular Descriptors for Structure–Activity Applications: A Hands-On Approach*, pp. 3–53. New York, NY: Springer New York, 2018.