DeepPurpose: A Deep Learning Library for Drug-Target Interaction Prediction

Kexin Huang[1], Tianfan Fu[2], Lucas M. Glass[3], Marinka Zitnik[1], Cao Xiao[3], and Jimeng Sun[4]

[1]Harvard University, Boston, MA
[2]Georgia Institute of Technology, Atlanta, GA
[3]IQVIA, Cambridge, MA
[4]University of Illinois at Urbana-Champaign, Urbana, IL

December 11, 2020

Abstract

Accurate prediction of drug-target interactions (DTI) is crucial for drug discovery. Recently, deep learning (DL) models for show promising performance for DTI prediction. However, these models can be difficult to use for both computer scientists entering the biomedical field and bioinformaticians with limited DL experience. We present DeepPurpose, a comprehensive and easy-to-use deep learning library for DTI prediction. DeepPurpose supports training of customized DTI prediction models by implementing 15 compound and protein encoders and over 50 neural architectures, along with providing many other useful features. We demonstrate state-of-the-art performance of DeepPurpose on several benchmark datasets.
Supplementary: Supplementary data are available at Bioinformatics online.

1 Introduction

Drug-target interactions (DTI) characterize the binding of compounds to protein targets [Santos et al., 2017]. Accurate identification of molecular drug targets is fundamental for drug discovery and development [Rutkowska et al., 2016, Zitnik et al., 2019] and is especially important for finding effective and safe treatments for new pathogens, including SARS-CoV-2 [Velavan and Meyer, 2020].

Deep learning (DL) has advanced traditional computational modeling of compounds by offering an increased expressive power in identifying, processing, and extrapolating complex patterns in molecular data [Öztürk et al., 2018, Lee et al., 2019]. There are many DL models designed for DTI prediction [Öztürk et al., 2018, Lee et al., 2019, Nguyen et al., 2020]. However, to generate predictions, deploy DL models in practice, test, and evaluate model performance, one needs considerable programming skills and extensive biochemical knowledge. Prevailing tools are designed for experienced interdisciplinary researchers. They are challenging to use by both computer scientists entering the biomedical field and domain bioinformaticians with limited experience in training and deploying DL models. Furthermore, each open-sourced tool has a

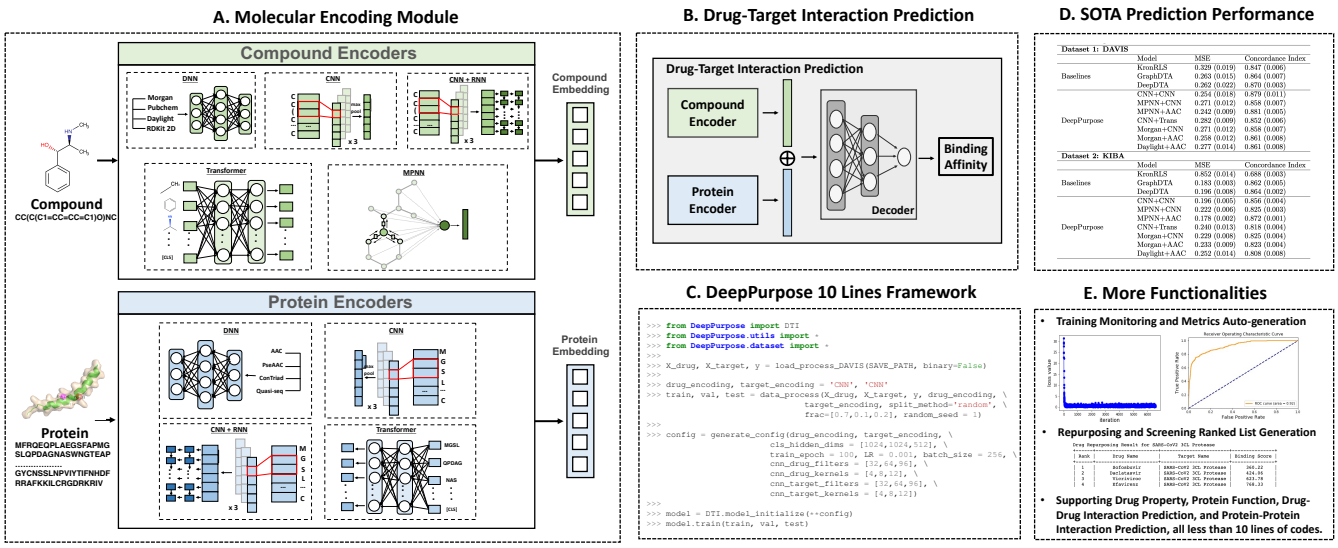

Figure 1: **Overview of DeepPurpose library.** (A) DeepPurpose takes as input the SMILES of a compound and a protein's amino acid sequence and then generates embeddings for them. (B) The learned embeddings are then concatenated and fed into a decoder to predict DTI binding affinity. (C) DeepPurpose provides a simple but flexible programming framework that implements over 50 state-of-the-art deep learning models for DTI prediction. (D) DeepPurpose models achieve comparable performance with three other DTI prediction algorithms on two benchmark datasets. (E) Finally, DeepPurpose has many functionalities, including monitoring the training process, debugging, and generation ranked lists for repurposing and screening. Further, DeepPurpose supports other downstream prediction tasks (e.g., drug-drug interaction prediction, compound property prediction).

different programming interface and is coded differently, which prevents easy integration of outputs from various methods for model ensembles [Yang et al., 2019].

Here, we introduce DeepPurpose, a deep learning library for encoding and downstream prediction of proteins and compounds. DeepPurpose allows rapid prototyping via a programming framework that implements over 50 deep learning models, seven protein encoders, and eight compound encoders. Empirically, we find that models implemented in DeepPurpose achieve state-of-the-art prediction performance on DTI benchmark datasets.

2 DeepPurpose Library

Deep learning models for DTI prediction can be formulated as an encoder-decoder architectures [Cho et al., 2014a]. DeepPurpose library implements a unifying encoder-decoder framework, which makes the library uniquely flexible. By merely specifying an encoder's name, the user can automatically connect a encoder of interest with the relevant decoder. DeepPurpose then trains the corresponding encoder-decoder model in an end-to-end manner. Finally, the user accesses the trained model either programmatically or via a visual interface and uses the model for DTI prediction.

2.1 Module for Encoding Proteins and Compounds

DeepPurpose takes the compound's simplified molecular-input line-entry system (SMILES) string and protein amino acid sequence pair as input. Then, they are fed into molecular encoders which specifies a deep transfor-

mation function that maps compounds and proteins to a vector representation. In particular, for compounds, DeepPurpose provides eight encoders using different modalities of compounds: Multi-Layer Perceptrons (MLP) on Morgan, PubChem, Daylight and RDKit 2D Fingerprint; Convolutional Neural Network (CNN) on SMILES strings; Recurrent Neural Network (RNN) on top of CNN; transformer encoders on substructure fingerprints; message passing graph neural network on molecular graph. For proteins, DeepPurpose provides seven encoders for the input amino acid sequence: MLP on Amino Acid Composition (AAC), Pseudo AAC, Conjoint Triad, Quasi-Sequence descriptors; CNN on amino acid sequences; RNN on top of CNN; transformer encoder on substructure fingerprints. Note that alternative input features may not work for a specific encoder architecture. The detailed encoder specifications and references are described in Supplementary.

2.2 Module for DTI Prediction

DeepPurpose feeds the learned protein and compound embeddings into an MLP decoder to generate predictions. Output scores include both continuous binding scores, such as the median inhibitory concentration (IC_{50}), as well as binary outputs indicating whether a protein binds to a compound. The library detects whether the task is regression or classification and switches to the correct loss function and evaluation metrics. In the case of regression, we use the Mean Square Error (MSE) as the loss function and MSE, Concordance Index, and Pearson Correlation as performance metrics. In the classification case, we use Binary Cross Entropy as the loss function and Area Under the Receiver Operating Characteristics (AUROC), Area Under Precision-Recall (AUPRC), and F-1 score as performance metrics. At inference, given new proteins and new compounds, DeepPurpose returns prediction scores representing predicted probabilities of binding between compounds and proteins.

2.3 Modules for Other Downstream Prediction Tasks

DeepPurpose includes `repurposing` and `virtual_screening` functions. Using only a few lines of codes that specify a list of compounds library to be screened upon and an optional set of training dataset, DeepPurpose trains five deep learning models, aggregates prediction results, and generates a descriptive ranked list in which compound candidates with the highest predicted binding scores are placed at the top. If the user does not specify a training dataset, DeepPurpose uses a pre-trained deep model for prediction. This list can then be examined to identify promising compound candidates for further experiments. Second, DeepPurpose also supports user-friendly programming frameworks for other modeling tasks, including drug and protein property prediction, drug-drug interaction prediction, and protein-protein interaction prediction (See Supplementary). Third, DeepPurpose provides an interface to many types of data, including public large binding affinity dataset [Liu et al., 2007], bioassay data [Kim et al., 2019], and a drug repurposing library [Corsello et al., 2017].

2.4 Programming Framework and Implementation Details

The functionality of DeepPurpose is modularized into six key steps where a single line of code can invoke each step: a) Load the dataset from a local file or load a DeepPurpose benchmark dataset. b) Specify the names of compound and protein encoders. c) Split the dataset into training, validation and testing sets using `data_process` function, which implements a variety of data-split strategies. d) Create a configuration file and specify model parameters. If needed, DeepPurpose can automatically search for optimal values of hyper-parameters. e) Initialize a model using the configuration file. Alternatively, the user can load a pre-trained model or a previously saved model. f) Finally, train the model using `train` function and monitor the progress of training and performance metrics. DeepPurpose is OS-agnostic and uses the Jupyter Notebook interface. It

can be run in the cloud or locally. All datasets, models, documentation, installation instructions, and tutorials are provided at https://github.com/kexinhuang12345/DeepPurpose.

3 Using DeepPurpose for DTI Prediction

To demonstrate the use of DeepPurpose, we compare DeepPurpose with KronRLS [Pahikkala et al., 2015], a popular DTI method, and GraphDTA [Nguyen et al., 2020] and DeepDTA [Öztürk et al., 2018], state-of-the-art DL methods. We find that many DeepPurpose models achieve comparable prediction performance on two benchmark datasets, DAVIS [Davis et al., 2011] and KIBA [He et al., 2017] (Figure 1D). A complete script to generate the results is provided in Supplementary Information.

4 DeepPurpose with Interactive Web Interface

In addition to rapid model prototyping, DeepPurpose also provides utility functions to load a pre-trained model and make predictions for a new drug and target inputs. This functionality allows domain scientists to examine predictions quickly, modify the inputs based on predictions, and iterate on the process until finding a drug or target with desired properties. We leverage Gradio [Abid et al., 2019] to create a web interface programmatically. We use a user-trained DeepPurpose model in the backend and create a custom web interface in fewer than ten code lines. This web interface takes the SMILES and amino acid sequence as the input and returns prediction scores with less than 1-second latency. We provide examples in the Supplementary.

References

Rita Santos, Oleg Ursu, Anna Gaulton, A Patrícia Bento, Ramesh S Donadi, Cristian G Bologa, Anneli Karlsson, Bissan Al-Lazikani, Anne Hersey, Tudor I Oprea, et al. A comprehensive map of molecular drug targets. *Nature reviews Drug discovery*, 16(1):19–34, 2017.

Anna Rutkowska, Douglas W. Thomson, Johanna Vappiani, Thilo Werner, Katrin M. Mueller, Lars Dittus, Jana Krause, Marcel Muelbaier, Giovanna Bergamini, and Marcus Bantscheff. A Modular Probe Strategy for Drug Localization, Target Identification and Target Occupancy Measurement on Single Cell Level. *ACS Chemical Biology*, 11(9):2541–2550, September 2016. ISSN 1554-8929. doi: 10.1021/acschembio.6b00346. URL https://doi.org/10.1021/acschembio.6b00346.

Marinka Zitnik, Francis Nguyen, Bo Wang, Jure Leskovec, Anna Goldenberg, and Michael M. Hoffman. Machine learning for integrating data in biology and medicine: Principles, practice, and opportunities. *Information Fusion*, 50:71–91, October 2019. ISSN 1566-2535. doi: 10.1016/j.inffus.2018.09.012. URL http://www.sciencedirect.com/science/article/pii/S1566253518304482.

Thirumalaisamy P. Velavan and Christian G. Meyer. The COVID19 epidemic. *Tropical Medicine & International Health*, 25(3):278–280, March 2020. ISSN 1360-2276. doi: 10.1111/tmi.13383. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7169770/.

Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. DeepDTA: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829, September 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty593. URL https://academic.oup.com/bioinformatics/article/34/17/i821/5093245.

Ingoo Lee, Jongsoo Keum, and Hojung Nam. DeepConv-DTI: Prediction of drug-target interactions via deep learning with convolution on protein sequences. *PLOS Computational Biology*, 15(6), June 2019. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1007129. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007129.

Thin Nguyen, Hang Le, Thomas P. Quinn, Thuc Le, and Svetha Venkatesh. GraphDTA: Predicting drug–target binding affinity with graph neural networks. *bioRxiv*, April 2020.

Kevin Yang, Kyle Swanson, Wengong Jin, Connor Coley, Philipp Eiden, Hua Gao, Angel Guzman-Perez, Timothy Hopper, Brian Kelley, Miriam Mathea, Andrew Palmer, Volker Settels, Tommi Jaakkola, Klavs Jensen, and Regina Barzilay. Analyzing Learned Molecular Representations for Property Prediction. *Journal of Chemical Information and Modeling*, 59(8):3370–3388, August 2019. ISSN 1549-9596. doi: 10.1021/acs.jcim.9b00237. URL https://doi.org/10.1021/acs.jcim.9b00237.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation at ACL*, pages 103–111, Doha, Qatar, October 2014a. doi: 10.3115/v1/W14-4012. URL https://www.aclweb.org/anthology/W14-4012.

Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N. Jorissen, and Michael K. Gilson. BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic Acids Research*, 35: D198–D201, January 2007.

Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, et al. Pubchem 2019 update: improved access to chemical data. *Nucleic Acids Research*, 47(D1):D1102–D1109, 2019.

Steven M Corsello, Joshua A Bittker, Zihan Liu, Joshua Gould, Patrick McCarren, Jodi E Hirschman, Stephen E Johnston, Anita Vrcic, Bang Wong, Mariya Khan, et al. The drug repurposing hub: a next-generation drug library and information resource. *Nature medicine*, 23(4):405–408, 2017.

Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug–target interaction predictions. *Briefings in Bioinformatics*, 16(2): 325–337, March 2015. ISSN 1467-5463. doi: 10.1093/bib/bbu010. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4364066/.

Mindy I. Davis, Jeremy P. Hunt, Sanna Herrgard, Pietro Ciceri, Lisa M. Wodicka, Gabriel Pallares, Michael Hocker, Daniel K. Treiber, and Patrick P. Zarrinkar. Comprehensive analysis of kinase inhibitor selectivity. *Nature Biotechnology*, 29(11):1046–1051, October 2011. ISSN 1546-1696. doi: 10.1038/nbt.1990.

Tong He, Marten Heidemeyer, Fuqiang Ban, Artem Cherkasov, and Martin Ester. SimBoost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines. *Journal of Cheminformatics*, 9, April 2017. ISSN 1758-2946. doi: 10.1186/s13321-017-0209-z. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5395521/.

Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.

David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, 2014b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

Kexin Huang, Cao Xiao, Trong Nghia Hoang, Lucas M Glass, and Jimeng Sun. Caster: Predicting drug interactions with chemical substructure representation. *AAAI*, 2020.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, pages 1263–1272, 2017.

M Reczko and H Bohr. The def data base of sequence based protein fold class predictions. *Nucleic acids research*, 22(17):3616, 1994.

Kuo-Chen Chou. Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. *Bioinformatics*, 21(1):10–19, 2005.

Juwen Shen, Jian Zhang, Xiaomin Luo, Weiliang Zhu, Kunqian Yu, Kaixian Chen, Yixue Li, and Hualiang Jiang. Predicting protein–protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences*, 104(11):4337–4341, 2007.

Kuo-Chen Chou. Prediction of protein subcellular locations by incorporating quasi-sequence-order effect. *Biochemical and Biophysical Research Communications*, 278(2):477–483, 2000.

A Further Details on DeepPurpose

Overview of DeepPurpose Framework. SkipGNN uses an encoder-decoder framework for DTI prediction. The input of SkipGNN is a compound SMILES string and protein amino acid sequence pair. The output of SkipGNN is a score that measures the binding activity of the input compound protein pair. The power of deep learning models comes from its ability to create predictive feature vectors also called *embeddings*. In particular, SkipGNN encodes both input compound and protein through various deep learning encoders to obtain their deep embeddings, then concatenates and feed them into a decoder, which is another deep neural network that aims to classify whether the input compound and target protein bind.

Encoders Implemented in DeepPurpose. SkipGNN provides 8 compound encoders and 7 protein encoders with numerous variants, ranging from classic chemical informatics fingerprints to various deep neural networks. SkipGNN feed two latent vectors generated from compound and protein encoders into the decoder to produce the final prediction score. With such a pipeline design, switching encoders is very simple in SkipGNN. By configuring a different encoder name SkipGNN will automatically switch to the required encoder model and connect them with the decoder for prediction.

Ritonavir: CC(C)C1=NC(=CS1)CN(C)C(=O)NC(C(C)C)C(=O)NC(CC2=CC=CC=C2)CC(C(CC3=CC=CC=C3)NC(=O)OCC4=CN=CS4)O

SARS-CoV2-3CL Protease: SGFRKMAFPSGKVEGCMVQVTCGTTTLNGLWLDDVVYCPRHVICTSEDMLNPNYEDLLIRKSNHNFLVQA GNVQLRVIGHSMQNCVLKLKVDTANPKTPKYKFVRIQPGQTFSVLACYNGSPSGVYQCAMRPNFTIKGSFLNGSCGSVGFNIDYDCVSFCYMHH MELPTGVHAGTDLEGNFYGPFVDRQTAQAAGTDTTITVNVLAWLYAAVINGDRWFLNRFTTTLNDFNLVAMKYNYEPLTQDHVDILGPLSAQTG IAVLDMCASLKELLQNGMNGRTILGSALLEDEFTPFDVVRQCSGVTFQ

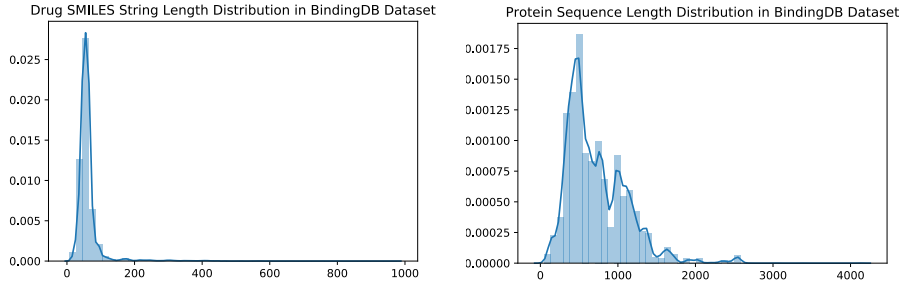

Figure 2: Examples of input representation. Compound is represented as SMILES string and protein is represented as amino acid sequence. The length distribution of the input are provided.

Compound Encoders. The input compound is represented by SMILES strings corresponding to molecule graphs (Figure. 2).

1. **Morgan Fingerprint** Rogers and Hahn [2010] is a 1024-length bits vector that encodes circular radius-2 substructures. A multi-layer perceptron is then applied on the binary fingerprint vector.

2. **Pubchem** Kim et al. [2019] is a 881-length bits vector, where each bit corrresponds to a hand-crafted important substructures. A multi-layer perceptron is then applied on top of the vector.

3. **Daylight** [1] is a 2048-length vector that encodes path-based substructures. A multi-layer perceptron is then applied on top of the vector.

4. **RDKit-2D** [2] is a 200-length vector that describes global pharmacophore descriptor. It is normalized to make the range of the features in the same scale using cumulative density function fit given a sample of the molecules.

5. **CNN** Krizhevsky et al. [2012] is a multi-layer 1D convolutional neural network. The SMILES characters are first encoded with an embedding layer and then fed into the CNN convolutions. A global max pooling layer is then attached and a latent vector describe the compound is generated.

6. **CNN+RNN** Cho et al. [2014b], Hochreiter and Schmidhuber [1997] attaches a bidirectional recurrent neural network (GRU or LSTM) on top of the 1D CNN output to leverage the more global temporal dimension of compound. The input is also the SMILES character embedding.

7. **Transformer** Vaswani et al. [2017] uses a self-attention based transformer encoder that operates on the sub-structure partition fingerprint Huang et al. [2020].

8. **MPNN** Gilmer et al. [2017] is a message-passing graph neural network that operate on the compound molecular graph. It transmits latent information among the atoms and edges, where the input features incorporate atom/edge level chemical descriptors and the connection message. After obtaining embedding vector for each atom and edge, a readout function (mean/sum) is used to obtain a (molecular) graph-level embedding vector.

[1] Daylight chemical information systems: https://www.daylight.com/
[2] https: //github.com/bp-kelley/descriptastorus

Protein Encoders. The input targets are proteins represented as sequences of 20 different kinds of amino acids (Figure. 2).

1. **AAC** Reczko and Bohr [1994] is a 8,420-length vector where each position correpsonds to an amino acid k-mers and k is up to 3.
2. **PseAAC** Chou [2005] includes the protein hydrophobicity and hydrophilicity patterns information in addition to the composition.
3. **Conjoint Triad** Shen et al. [2007] uses the continuous three amino acids frequency distribution from a hand-crafted 7-letter alphabet.
4. **Quasi Sequence** Chou [2000] takes account for the sequence order effect using a set of sequence-order-coupling numbers.
5. **CNN** Krizhevsky et al. [2012] is a multi-layer 1D convolutional neural network. The target amino acid is decomposed to each individual character and is encoded with an embedding layer and then fed into the CNN convolutions. It follows a global max pooling layer.
6. **CNN+RNN** Cho et al. [2014b], Hochreiter and Schmidhuber [1997] attaches a bidirectional recurrent neural network (GRU or LSTM) on top of the 1D CNN output to leverage the sequence order information.
7. **Transformer** Vaswani et al. [2017] uses a self-attention based transformer encoder that operates on the sub-structure partition fingerprint Huang et al. [2020] of proteins. Since transformer's computation time and memory is quadratic on the input size, it is computational infeasible to treat each amino acid symbol as a token. The partition fingerprint decomposes amino acid sequence into protein substructures of moderate sized such as motifs and then each of the partition is considered as a token and fed into the model.

Note on Feature-Architecture Combinations During implementation, we notice that all of the architectures require specific input features, which means other input features are incompatible with the architecture. DeepPurpose currently supports five architectures: MLP, CNN, CNN+RNN, Transformer, and MPNN, and it supports five types of features: fingerprints, SMILES string, ESPF fingerprint, and molecular graph. Notice that these architectures are not suitable for alternative features. MLP can take in fingerprints vectors, but one-hot matrices of SMILES or graphs are incompatible. CNN/CNN+RNN expect a matrix where each row is a one-hot vector for an entity in the SMILES/Amino Acids, but cannot take a single fingerprint vector or graphs. Transformer is quadratic to the input length, which makes the long fingerprints and SMILES/Amino acids computationally expensive. MPNN operates on graphs, thus, it is incompatible with fingerprints, SMILES, ESPF. In addition, the current included features & encoders combinations are not randomly assembled but are previously included in the literature and have shown strong performance for molecular modeling with compounds and targets.

Programming Framework The functionality of DeepPurpose is modularized into six key steps where a single line of code can invoke each step: a) Load the dataset from a local file or load a DeepPurpose benchmark dataset. b) Specify the names of compound and protein encoders. c) Split the dataset into training, validation and testing sets using `data_process` function, which implements a variety of data-split strategies. d) Create a configuration file and specify model parameters. If needed, DeepPurpose can automatically search for optimal values of hyper-parameters. e) Initialize a model using the configuration file. Alternatively, the user can load a pre-trained model or a previously saved model. f) Finally, train the model using `train` function and monitor the progress of training and performance metrics.

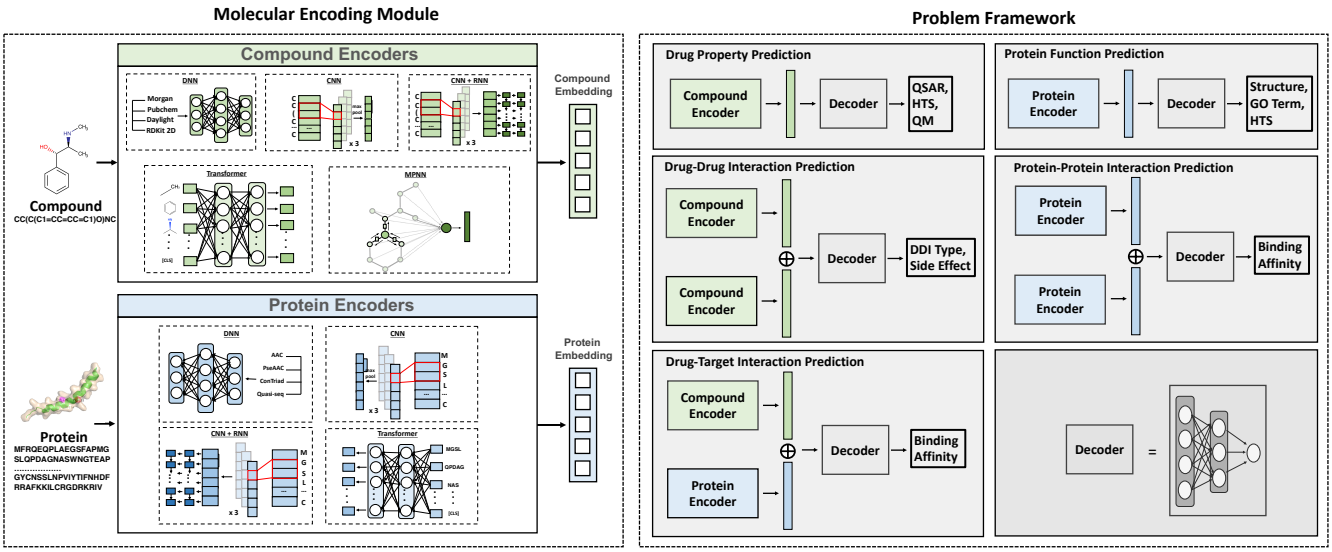

Figure 3: DeepPurpose supports DTI, DDI, PPI, Drug Property and Protein Function Prediction tasks.

Downstream Prediction, Objective Function, and Inference. After SkipGNN obtains latent compound and protein embedding, both are fed into a multi-layer perceptron decoder. There are two classes of tasks/datasets in drug target interaction prediction. One's label is binding score such as Kd, IC50, and they are continuous values while the other one's label is binary, whether or not they can bind. SkipGNN is able to automatically detect whether the task is regression for continuous label or classification for binary label by counting the number of unique labels in the data. For binding affinity score prediction, it uses mean squared error (MSE) loss (Eq. 1). For binary interaction prediction, it uses binary cross entropy (BCE) loss (Eq. 2).

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

$$\mathscr{L}_{BCE} = \frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i), \tag{2}$$

where y_i is the true label and $\hat{y}_i$ is the predicted label for i-th compound-protein pair. For evaluation metrics, we use MSE, Concordance Index, and Pearson Correlation for continuous regression and Receiver Operating Characteristics-Area Under the Curve (ROC-AUC), Precision Recall-Area Under the Curve (PR-AUC) and F1 score at threshold 0.5 for binary classification. During inference, given new proteins or new compounds, the model prediction is used as the predicted binding score/interaction probability.

B DeepPurpose for Other Tasks Related to Compound and Protein

In addition to DTI prediction, DeepPurpose also supports user-friendly programming frameworks for other molecular modeling tasks, namely, drug/protein property prediction, drug-drug interaction prediction, protein-protein interaction prediction tasks. We provide a framework illustration in Figure. 3.