Advanced Graph and Sequence Neural Networks for Molecular Property Prediction and Drug Discovery

Zhengyang Wang[1,+]**, Meng Liu**[1,+]**, Youzhi Luo**[1,+]**, Zhao Xu**[1,+]**, Yaochen Xie**[1,+]**, Limei Wang**[1,+]**, Lei Cai**[1,2+]**, and Shuiwang Ji**[1,*]

[1]Texas A&M University, Department of Computer Science and Engineering, College Station, TX 77843, USA
[2]Currently with Microsoft, Bellevue, WA 98004, USA
[*]Correspondence should be addressed to: sji@tamu.edu
[+]These authors contributed equally to this work.

ABSTRACT

Properties of molecules are indicative of their functions and thus are useful in many applications. With the advances of deep learning methods, computational approaches for predicting molecular properties are gaining increasing momentum. However, there lacks customized and advanced methods and comprehensive tools for this task currently. Here we develop the MoleculeKit, a suite of comprehensive machine learning methods and tools spanning different computational models and molecular representations for molecular property prediction and drug discovery. Specifically, MoleculeKit represents molecules as both graphs and sequences. Built on these representations, MoleculeKit includes novel deep models for learning from molecular graphs and sequences. Therefore, MoleculeKit not only serves as a comprehensive tool, but also contributes towards developing novel and advanced graph and sequence learning methodologies. Results on both online and offline antibiotics discovery and molecular property prediction tasks show that MoleculeKit achieves consistent improvements over prior methods.

1 Introduction

Molecular property prediction is one of the key tasks in cheminformatics and has applications in many fields, including quantum mechanics, physical chemistry, biophysics and physiology[1]. With the rapid advances of machine learning methods, there is a growing trend and success to apply computational methods to predict molecular properties, which saves a huge amount of time-consuming and expensive chemical processes[2–12]. With emerging global challenges like the COVID-19 pandemic, it is even more urgent to develop powerful machine learning methods for molecular property prediction, thereby accelerating the speed of drug discovery[13–16].

In the literature, there are two common ways to represent molecules *in silico*, *i.e.*, molecular graphs and simplified molecular-input line-entry system (SMILES) sequences[17]. Molecular graphs are intuitive to humans and informative to machine learning models. In particular, molecular graphs can keep the structure information of molecules by clearly showing the connections between atoms. On the other hand, SMILES sequences have a longer history in cheminformatics and have been widely used. SMILES is a linguistic construct to represent molecules as single-line sequences using certain simple grammar rules and a vocabulary composed of characters representing atoms and bonds. The SMILES sequences also stores essential chemical information of molecules. Various machine learning methods have been proposed on either molecular graphs[1,7–9,12,18–21] or SMILES sequences[22–27]. Setting aside the difference in data types, these machine learning methods can also be divided by the method type. That is, we can categorize them into deep learning models and non-deep learning models. Deep learning methods[28] have set new records in most machine learning tasks. However, non-deep learning methods still show advantages in certain cases, especially when the amount of available labeled data is small. Nevertheless, there is no clear evidence on which data type or method type leads to better performances for molecular property prediction[1,12,25]. In addition, the development of deep learning models for molecular property prediction is still in early stage.

In this work, we propose MoleculeKit, an advanced machine learning tool for molecular property prediction and drug discovery. MoleculeKit is unique in three aspects. First, MoleculeKit consists of a suite of comprehensive machine learning methods across different data types and method types. We expect them to provide complementary information for molecular property prediction and yield better performance. Second, we propose a new graph-based deep learning method named multi-level message passing neural network (ML-MPNN), which is included in MoleculeKit. ML-MPNN is able to aggregate information from a full range of levels in molecular graphs, including nodes, edges, subgraphs, and the entire graph and makes full use of richly informative molecular graphs (Section 4.2.1). Third, MoleculeKit is equipped with a novel sequence-based deep learning method, namely contrastive-BERT. Contrastive-BERT is based on the state-of-the-art BERT[29] and includes the masked embedding recovery task, a novel self-supervised pre-training task via contrastive learning (Section 4.3.1). With these three features, MoleculeKit not only serves as a comprehensive tool for molecular property prediction, but also contributes towards developing novel and advanced techniques. Concretely, MoleculeKit has four modules covering deep and non-deep methods based on both molecular graphs and SMILES sequences; those are, ML-MPNN, Weisfeiler-Lehman subtree (WL-subtree) kernel[18], contrastive-BERT and subsequence kernel[22] (Fig. 1a).
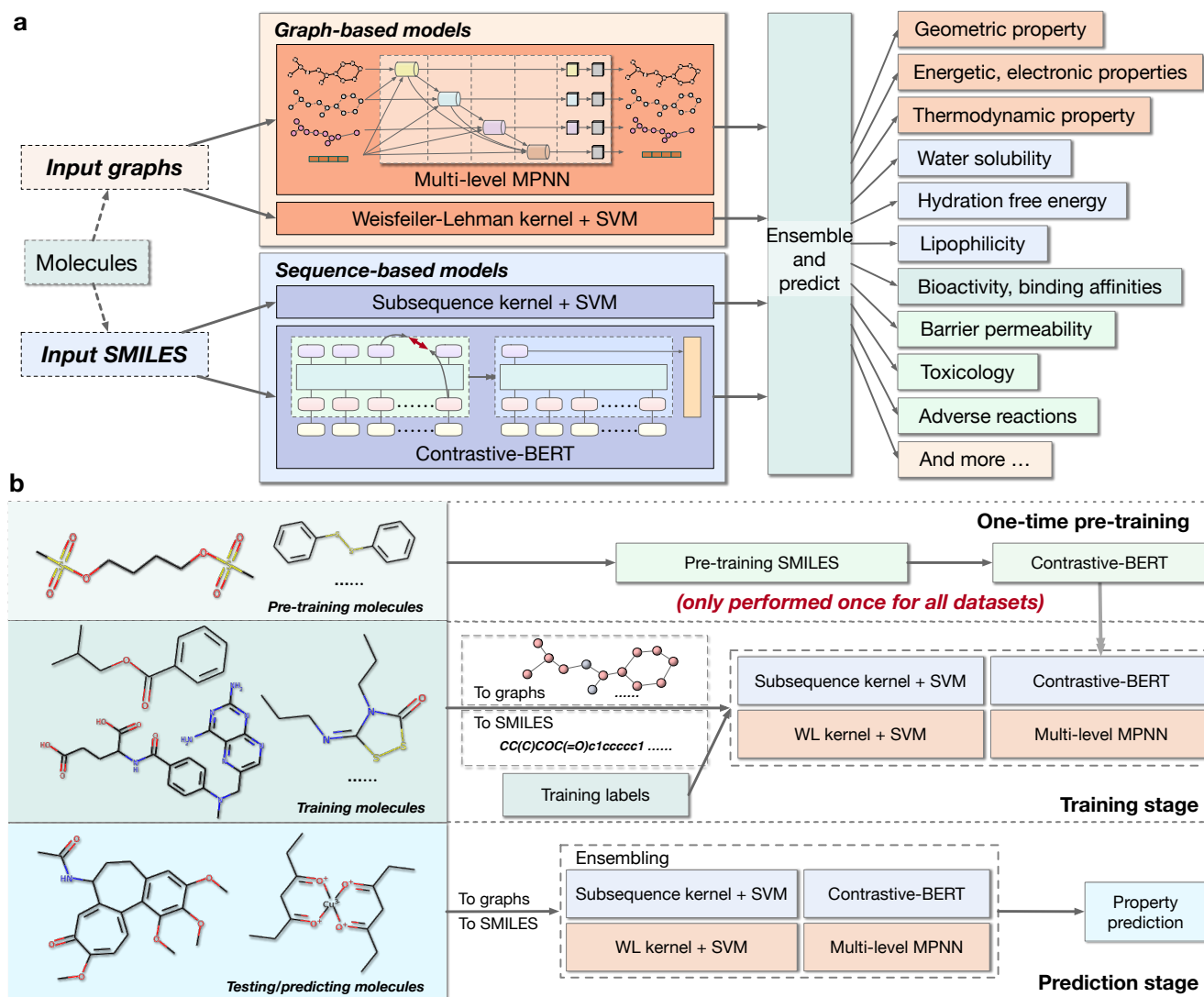
Figure 1. MoleculeKit overview, training and inference. a, MoleculeKit is composed of four modules, including two graph-based models, multi-level message passing neural network (ML-MPNN) and Weisfeiler-Lehman subtree (WL-subtree) kernel, and two SMILES sequence-based models, contrastive-BERT and subsequence kernel. We represent molecules as molecular graphs and SMILES sequences, and feed into the corresponding modules. The final prediction result of MoleculeKit is obtained by ensembling the predicted scores of the four modules. **b**, Given a training dataset with labeled molecules, each module is trained individually under the standard supervised setting. In particular, contrastive-BERT initializes the training parameters differently from the other three modules who apply the random initialization. The initialization of contrastive-BERT comes from an extra one-time pre-training phase on a large pre-training dataset with unlabeled molecules. In the prediction stage, MoleculeKit predicts the property of molecules by ensembling scores predicted by the four modules.

In the following, we (1) demonstrate the overall effectiveness of MoleculeKit in the AI Cures open challenge task[16], (2) show the advantages of MoleculeKit and each module of it in drug discovery with an off-line dataset, (3) analyze the improvements brought by MoleculeKit as well as our proposed ML-MPNN and contrastive-BERT on a wide range of molecular property prediction tasks from MoleculeNet benchmarks[1]. In addition, we discuss the key differences of ML-MPNN and contrastive-BERT from related works, and perform comprehensive ablation studies to support the novel designs in ML-MPNN and contrastive-BERT.

2 Results

2.1 Pipeline: training and inference

The proposed MoleculeKit consists of four modules; namely, multi-level message passing neural network (ML-MPNN), Weisfeiler-Lehman subtree (WL-subtree) kernel[18], contrastive-BERT and subsequence kernel[22] (Fig. 1a). Despite the differences in data types and model architectures among these four modules, the training and inference procedures are unified (Fig. 1b). First, representing molecules as molecular graphs and SMILES sequences[17] is effortless. Next, in the training stage, we assume a training dataset for each molecular property prediction task, where molecules are labeled by the property of interest. Each module is then trained under the standard supervised learning setting, just with different initialization methods. Specifically, contrastive-BERT initializes training parameters from an extra one-time pre-training phase, while the other three modules apply the random initialization. The pre-training of contrastive-BERT is performed under a self-supervised contrastive learning setting on 2 million unlabeled molecules from the ZINC[30] database. It is worth emphasized that this pre-training phase is conducted only once for all tasks. Finally, during inference, we use the ensemble of the predicted scores from the four modules as the final prediction result of our MoleculeKit. Users of MoleculeKit are also given the freedom of employing fewer modules.

2.2 AI Cures open challenge task

The COVID-19 pandemic has posed a great challenge to public health, and we first demonstrate how our MoleculeKit can be used to accelerate drug discovery for emerging global threats like the COVID-19. To this end, we use our MoleculeKit to participate in an open challenge on drug discovery, known as the AI Cures[16], which aims at discovering new antibiotics to cure the secondary lungs infections caused by COVID-19. The entire dataset consists of 2,335 chemical molecules, which is split into a training set of 2,097 molecules and a test set of 238 molecules. The prediction target is whether a molecule has the antibacterial activity, or inhibition, to *Pseudomonas aeruginosa*, which is a bacterium leading to secondary lungs infections of COVID-19 patients. The ground truth binary labels of all molecules are obtained from *in-vitro* screening assay. The class labels of the training data have been made available to challenge participants, while the labels of test data have been hold out for the challenge organizers to evaluate the performance of participants. The prediction performance is evaluated by ROC-AUC and PRC-AUC (Section 4.5) on the hold-out test set.

We use all the four modules of MoleculeKit in this task with both graph and sequence representations of molecules. Our final prediction is obtained with the ensemble of these four modules. Concretely, for each molecule of the test set, we use the

Table 1. Results on the AI Cures open challenge task in terms of PRC-AUC and ROC-AUC. Our model name is MoleculeKit, and our team name is DIVE@TAMU.

(a) Test PRC-AUC

Rank	Model	Author	Submissions	Test PRC-AUC
1	MolecularG	AIDrug@PA	7	0.725
2	-	AGL Team	20	0.702
3	MoleculeKit	DIVE@TAMU	7	0.677
4	GB	BI	6	0.67
5	Chemprop ++	AICures@MIT	4	0.662
6	-	Mingjun Liu	3	0.657
7	Pre-trained OGB-GIN (ensemble)	Weihua Hu@Stanford	2	0.651
8	RF + fingerprint	Cyrus Maher@Vir Bio	1	0.649
9	Graph Self-supervised Learning	SJTU_NRC_Mila	3	0.622
10	-	Congjie He	10	0.611

(b) Test ROC-AUC

Rank	Model	Author	Submissions	Test ROC-AUC
1	MoleculeKit	DIVE@TAMU	7	0.928
2	Chemprop ++	AICures@MIT	4	0.877
3	-	Gianluca Bontempi	7	0.848
4	-	Apoorv Umang	1	0.84
5	Pre-trained OGB-GIN (ensemble)	Weihua Hu@Stanford	2	0.837
6	-	Kexin Huang	1	0.824
7	Chemprop	Rajat Gupta	7	0.818
8	MLP	IITM	7	0.807
9	Graph Self-supervised Learning	SJTU_NRC_Mila	3	0.8
10	-	Congjie He	10	0.8

average of four predicted inhibition probabilities from four modules as its final predicted inhibition probability.

There are a total of 27 teams participating in this open challenge. The prediction performance of our MoleculeKit and those of other top teams are reported in Table 1. In summary, our MoleculeKit achieves the highest test ROC-AUC of 0.928 and ranks the third on test PRC-AUC with a value of 0.677. Note that the top two teams (AIDrug@PA and AGL Team) with higher Test PRC-AUC (0.725 and 0.702) than us achieve very low ROC-AUC (0.7 and 0.675). Our MoleculeKit outperforms them on average by 35% on test ROC-AUC, while they outperform us on average by 5% on test PRC-AUC. These results indicate that our MoleculeKit achieves very stable and consistent performance across two different evaluation metrics. Our promising performance and turnkey solution to this open challenge on drug discovery demonstrate that MoleculeKit can be used for drug screening and accelerating drug discovery.

2.3 Antibiotic discovery from Drug Repurposing Hub

In addition to the AI Cures open challenge task, we apply our MoleculeKit on another dataset to further demonstrate its advantages on accelerating drug discovery. In particular, we focus on predicting the capability of inhibiting the growth of *Escherichia coli*. It will greatly help finding new antibiotics to alleviate the bacteria's drug-resistance problem. The training dataset is composed of 2,335 molecules with binary labels obtained through screening. The testing dataset, provided by Stoke

Table 2. Results on the Drug Repurposing Hub (DRH) testing dataset in terms of PRC-AUC and ROC-AUC.

Model	Test PRC-AUC	Test ROC-AUC
Baseline	0.777	0.888
ML-MPNN	0.828	0.900
WL-subtree kernel	0.857	0.913
Contrastive-BERT	0.804	0.877
Subsequence kernel	0.778	0.878
MoleculeKit	0.867	0.919

et al.[15], contains 162 molecules form Drug Repurposing Hub (DRH), among which 53 are found inhibitory against *Escherichia coli*.

The baseline model in this experiment comes from Stoke et al.[15] as well; that is, the directed message passing neural network (D-MPNN)[12] is applied. Specifically, Stoke et al. randomly split the 2,335 training molecules into a new training dataset and a validation dataset with the ratio of 9:1 with 20 different random seeds, generating 20 different train-validation splits. Then they train D-MPNN on these 20 splits independently, yielding 20 D-MPNN models. During prediction on the testing dataset, the average of predicted scores from these 20 models is provided as the final prediction score. The performance is evaluated by PRC-AUC and ROC-AUC (Section 4.5).

We apply all the four modules of MoleculeKit on this task. In order to show the superiority of MoleculeKit, we do not perform the ensemble over different train-validation splits. Instead, we randomly split the 2,335 training molecules into a new training dataset and a validation dataset with the ratio of 8.5:1.5 once, and use this train-validation split only. In addition to the ensemble of all four modules, we report the performance of each module as ablation studies, indicating that the better performance of MoleculeKit does not merely come from the ensemble of multiple methods. The results are provided in Table 2.

Surprisingly, the subsequence kernel, without the ensemble over different train-validation splits, can already achieve the competitive performance with the baseline by itself. The sequence-based deep learning method, contrastive-BERT, outperforms the baseline by a large margin in terms of PRC-AUC while achieving a similar ROC-AUC. The slight decrease in ROC-AUC is complemented by the graph-based methods. Both graph-based methods are very effective by themselves, especially the WL-subtree kernel. As a result, with the ensemble of four modules, MoleculeKit improves the performance significantly.

The results clearly show three insights regarding MoleculeKit. First, by comparing ML-MPNN and contrastive-BERT with the baseline, we can see that our proposed deep learning methods are effective. Second, the kernel methods are very powerful, revealing why MoleculeKit includes them. Third, different data types and method types provide complementary information, resulting in a further improved performance with the ensemble.

2.4 MoleculeNet benchmarks for molecular property prediction

With two successful applications on drug discovery, we further explore MoleculeKit on a wider range of molecular property prediction tasks. Particularly, we apply MoleculeKit on MoleculeNet benchmarks[1] for molecular property prediction.

We perform experiments on 14 datasets of MoleculeNet benchmarks, focusing on various molecular properties in the
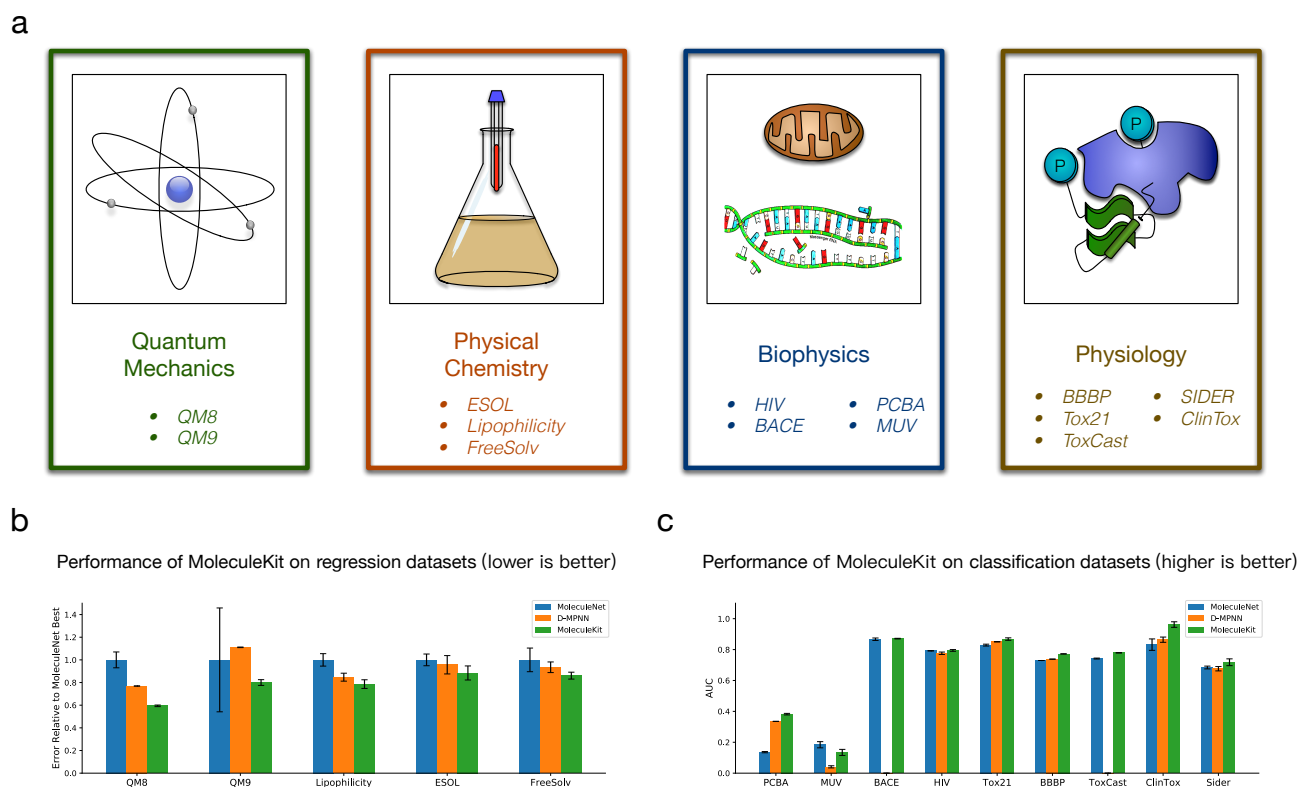
Figure 2. **MoleculeKit on MoleculeNet benchmarks for molecular property prediction. a**, We apply MoleculeKit on MoleculeNet benchmarks for a wide range of molecular property prediction tasks. Experiments are performed on 14 datasets, covering different molecular properties in the field quantum mechanics, physical chemistry, biophysics and physiology. **b**, Prediction performances of MoleculeKit on the testing dataset of 5 regression tasks, in terms of MAE or RMSE (Section 4.5, Supplementary Table 1). We follow Yang et al.[12] to plot the error relative to the MoleculeNet baseline. The confidence intervals are marked by the standard deviations. **c**, Prediction performances of MoleculeKit on the testing dataset of 9 classification tasks, in terms of ROC-AUC or PRC-AUC (Section 4.5, Supplementary Table 1). The absolute performance values are plotted. The confidence intervals are marked by the standard deviations. For b&c, the mean and standard deviation results are computed from 3 independent runs over 3 different train/validation/test splits on each dataset. The same 3 splits are used for baselines and our MoleculeKit for fair comparisons. Results of MoleculeNet[1] and D-MPNN[12] are directly copied from their papers. Note that D-MPNN is not evaluated on BACE and ToxCast datasets. All the corresponding quantitative results are provided in Supplementary Table 2. These results show significant and consistent performance gains brought by our MoleculeKit.

field of quantum mechanics, physical chemistry, biophysics and physiology (Fig. 2a). These properties can be either binary labels, like those in drug discovery, or continuous values. Correspondingly, they are formulated as binary classification tasks or regression tasks. Typically, binary classification tasks are evaluated by PRC-AUC or ROC-AUC, while regression tasks are evaluated by MAE or RMSE (Section 4.5). In order to serve as benchmarks and enable fair comparisons, Wu et al.[1] provide the split and evaluation methods for all the datasets. We report these key details of datasets in Supplementary Table 1. Further information can be found in Wu et al.[1].

Two baseline models are chosen for comparisons with our MoleculeKit. One is a collection of previously proposed molecular featurization and learning algorithms provided together with MoleculeNet benchmarks[1]. It includes both graph-based and sequence-based methods, covering both deep and non-deep models. Without loss of clarity, we denote this baseline as

MoleculeNet and always report the best performance from the collection. The other model is the directed message passing neural network (D-MPNN)[12], which incorporates latest techniques in graph neural networks and achieves the state-of-the-art performances on most datasets of MoleculeNet benchmarks. These two models form strong baselines in terms of both breadth and depth.

All the results in this section, including baselines and our MoleculeKit, are obtained by following the same settings provided in MoleculeNet benchmarks[1]. The performances of our MoleculeKit and the two baselines are compared in Fig. 2b&c. All the corresponding quantitative results are provided in Supplementary Table 2. Note that when showing performances on regression datasets (Fig. 2b), we follow Yang et al.[12] to plot the error relative to the MoleculeNet baseline, because the scale of performance values differ significantly across different datasets. On the other hand, when showing performances on classification datasets (Fig. 2c), the absolute performance values in $[0, 1]$ are plotted. Note that D-MPNN is not evaluated on BACE and ToxCast datasets[12]. It is also worth noting that, in this set of experiments, we do not always use the ensemble of all four modules for MoleculeKit. Instead, we choose modules for the final ensemble based on their validation performances. Besides, the two kernel methods are not used on QM9, PCBA, MUV, ToxCast datasets, as they are too large in terms of the number of molecules or tasks (Supplementary Table 1). In these cases, the kernel methods are infeasible due to lack of memory or exceeding long computation time. Overall, our MoleculeKit outperforms both baselines on 13 out of 14 datasets. On the MUV dataset, MoleculeKit is beaten by MoleculeNet while improved D-MPNN significantly. These results demonstrate the power of MoleculeKit on molecular property prediction.

Additionally, in order to demonstrate the effectiveness of our proposed novel deep learning methods, ML-MPNN and contrastive-BERT, we also perform ablation studies by comparing them individually with the baselines in the following.

Ablation studies on ML-MPNN. The comparisons between our proposed ML-MPNN and the two baselines, MoleculeNet and D-MPNN, are provided in Fig. 3a&b. The plots follow the same settings as Fig. 2b&c. The corresponding quantitative results can be found in Supplementary Table 2.

On 9 out of 14 datasets, ML-MPNN outperforms both baselines. Concretely, ML-MPNN achieves better performances than MoleculeNet[1] on 11 out of 14 datasets and D-MPNN[12] on 9 out of 12 datasets, respectively. In particular, we notice that MoleculeNet utilizes extra 3D coordinate information on QM8 and QM9 datasets, which D-MPNN and our ML-MPNN do not use. D-MPNN barely outperforms MoleculeNet on QM8 and underperforms MoleculeNet on QM9, while our ML-MPNN yields a performance boost.

The increased performances on a majority of the datasets show the effectiveness of our proposed ML-MPNN. Meanwhile, the comparisons between D-MPNN and ML-MPNN demonstrate that our proposed ML-MPNN can serve as the new state-of-the-art graph-based deep learning method for molecular property prediction.

Ablation studies on contrastive-BERT. The comparisons between our proposed contrastive-BERT and the two baselines are provided in Fig. 3c&d. The corresponding quantitative results can be found in Supplementary Table 2.

On half of 14 datasets, contrastive-BERT shows performances equal to or better than both baselines. Respectively,
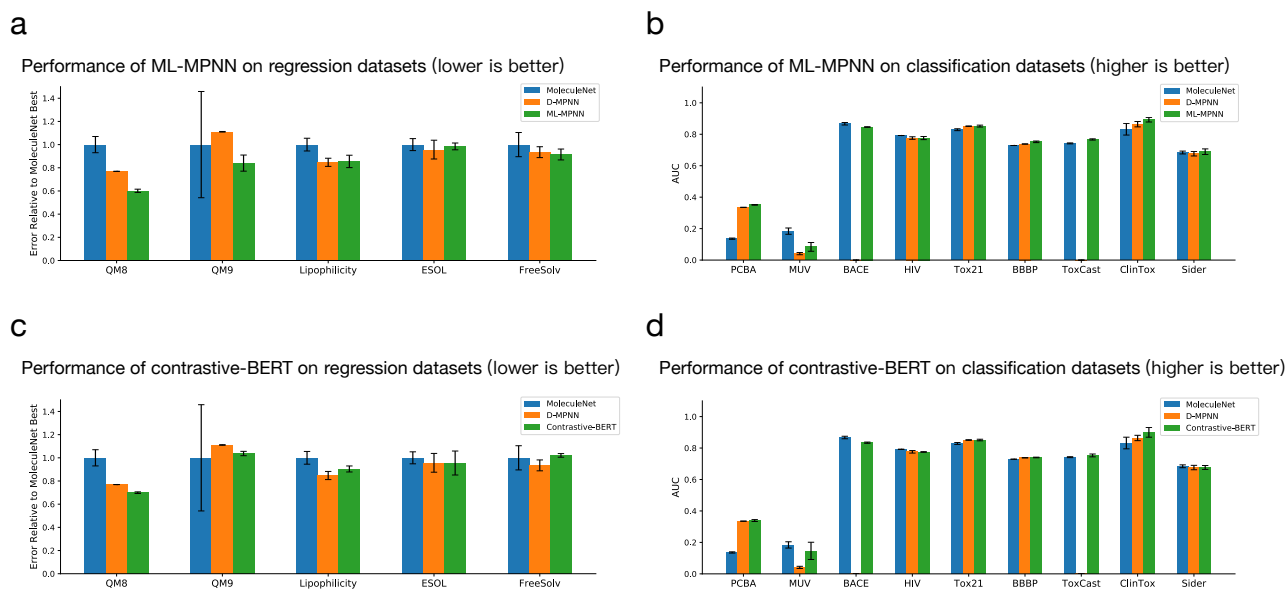
Figure 3. **ML-MPNN and contrastive-BERT on MoleculeNet benchmarks for molecular property prediction. a**, Prediction performances of ML-MPNN on the testing dataset of 5 regression tasks, in terms of MAE or RMSE (Section 4.5, Table 1). **b**, Prediction performances of ML-MPNN on the testing dataset of 9 classification tasks, in terms of ROC-AUC or PRC-AUC (Section 4.5, Supplementary Table 1). **c**, Prediction performances of contrastive-BERT on the testing dataset of 5 regression tasks, in terms of MAE or RMSE (Section 4.5, Supplementary Table 1). **d**, Prediction performances of contrastive-BERT on the testing dataset of 9 classification tasks, in terms of ROC-AUC or PRC-AUC (Section 4.5, Supplementary Table 1). All the figures follow the same settings as Fig. 2b&c. All the corresponding quantitative results are provided in Supplementary Table 2. These results demonstrate the advantages of our proposed novel deep learning methods, ML-MPNN and contrastive-BERT.

contrastive-BERT outperforms MoleculeNet[1] on 8 out of 14 datasets and D-MPNN[12] on 7 out of 12 datasets. Note that, both D-MPNN and ML-MPNN use additional global molecular features obtained with the open-source package RDKit[31] (Section 4.6), and MoleculeNet has access to extra 3D coordinate information on QM8 and QM9 datasets, as mentioned above. However, contrastive-BERT only uses SMILES sequences, without additional features nor extra information. Taking these into consideration, contrastive-BERT serves as a powerful sequence-based model for molecular property prediction.

3 Discussion

We propose MoleculeKit, an advanced tool composed of graph-based and sequence-based machine learning methods, for various molecular property prediction tasks including drug discovery. Specifically, MoleculeKit consists of four modules, including two graph-based methods, multi-level message passing neural network (ML-MPNN) and Weisfeiler-Lehman subtree (WL-subtree) kernel, and two sequence-based methods, contrastive-BERT and subsequence kernel. Among the four modules of MoleculeKit, ML-MPNN and contrastive-BERT are novel deep learning methods proposed in this work. In order to demonstrate the advantages of MoleculeKit as well as the two novel methods, we perform thorough experimental studies on two drug discovery applications and a wide range of molecular property prediction tasks.

We anticipate that our work would have potential impacts in multiple aspects. First, both our proposed ML-MPNN

and contrastive-BERT achieve success in various molecular property prediction applications by themselves. They stand for advanced techniques for graph-based and sequence-based machine learning method, respectively. We expect them to not only push forward the development of new models for molecular property prediction, but also inspire the proposal of more powerful machine learning methods in general. Second, we have shown that different data types and method types both provide complementary information and boost the performances with an ensemble as MoleculeKit. This serves as a useful insight when applying machine learning methods for molecular property prediction in practice. In particular, our MoleculeKit would serve as a reliable open-source tool. Third, our work may have a larger impact beyond the academic community. For example, we achieve top performances in the AI Cures open challenge, which aims at accelerating drug discovery for emerging global threats like the COVID-19.

Regarding the two novel deep learning methods proposed in this work, ML-MPNN and contrastive-BERT, we discuss the key differences from previous studies and illustrate the superiority of our methods. In terms of ML-MPNN, performing molecular property prediction by learning from molecular graphs has drawn a lot of attention in the community[1, 7–9, 12, 19–21]. Typically, existing works learn representations for molecules through message passing. The original MPNN[8] updates node representations by aggregating messages from its neighboring nodes and connected edges in each layer. Instead of updating representations for nodes, D-MPNN[12] learns representations for each directed edge. The recent proposed HIMP[21] incorporates representations associated with substructures during message passing. While these existing methods are shown to be effective, they only leverage limited information contained in the molecular graphs during message passing. Intuitively, information related to the molecular properties can come from different levels, such as edges, nodes, and subgraphs. Based on these insights, our ML-MPNN is designed to be capable of aggregating information from a full range of levels in molecular graphs.

Specifically, ML-MPNN can be regarded as an extension of the unified graph neural network framework[32]. The key difference lies in taking subgraph-level representations into consideration. ML-MPNN incorporates this level of information effectively, leading to a more powerful message passing framework. In order to show that the improvements of ML-MPNN do come from subgraph-level representations, we perform ablation studies to compare the performances between ML-MPNN and ML-MPNN without subgraph-level representations on MoleculeNet benchmarks. The comparison results are provided in Supplementary Table 3 and Supplementary Fig. 6. With subgraph-level representations, improvements can be observed on 12 out of 14 datasets. Another novel aspect of ML-MPNN is the normalization methods, We extend the the graph size normalization[33] to SizeNorm that normalizes multi-level representations before applying BatchNorm (Section 4.2.1). In order to verify the effectiveness of the SizeNorm and BatchNorm, we conduct ablation studies as well and show results in Supplementary Table 4 and Supplementary Fig. 7. Compared to the model without any normalization, SizeNorm can improve the performance on 5 datasets out of 14 datasets and BatchNorm can improve the performance on 11 datasets. Applying SizeNorm and BatchNorm together can improve performance on 12 datasets.

In terms of contrastive-BERT, there exists multiple studies on learning from sequential data[29, 34–36]. Recurrent neural networks (RNNs), like long short-term memory (LSTM)[34] and gated recurrent unit (GRU)[35], are the most popular sequence-

based machine learning models before the proposal of Transformer[36]. Based on Transformer, BERT[29] applies pre-training and achieves great success in various tasks. Transformer is theoretically more powerful than RNNs and shows improved performance in tasks like machine translation[36]. However, we find it necessary to perform pre-training as in BERT to achieve good performance on molecular property prediction, due to the small sizes of available datasets. To verity this, we conduct experiments comparing performances on MoleculeNet benchmarks between GRU, Transformer, and our contrastive-BERT, and report results in Supplementary Table 5 and Supplementary Fig. 8. We can see that GRU outperforms Transformer significantly on small datasets, while contrastive-BERT achieves the best performances on 13 out of 14 datasets. Contrastive-BERT has a similar architecture as Transformer but is featured by a pre-training phase via contrastive learning[37–40] (Section 4.3.1), indicating the necessity of effective pre-training.

Contrastive-BERT is not the first study that applies BERT[29] on molecular property prediction. SMILES-BERT[26] directly employs BERT on SMILES sequences with the masked language model pre-training task from BERT. This pre-training tasks is a classification task, where the model is asked to predict what the masked characters are. On the contrary, our contrastive-BERT proposes a novel pre-training task via contrastive learning, named masked embedding recovery task, where the model is asked to predict the embeddings of the masked characters. This contrastive learning task is substantially harder than the original classification task and is supposed to encourage learning better representations of SMILES sequences for down-streaming tasks. In order to demonstrate the advantages of our proposed contrastive-BERT and the masked embedding recovery task, we perform ablation studies comparing our contrastive-BERT with SMILES-BERT model. Results are provided in Supplementary Table 6 and Supplementary Fig. 9. Our contrastive-BERT model outperforms the SMILES-BERT model on 11 out of 14 datasets.

One limitation of our MoleculeKit is that it does not consider the 3D coordinate information of molecules when available. While we have shown that MoleculeKit is able to outperform models with 3D coordinate information into consideration, we expect incorporating such information into MoleculeKit may further improve the performances. Intuitively, molecules are 3D structures. Location information of atoms can be important for predicting molecular properties. Although there are several attempts[11,41,42] to consider such 3D coordinate information, how to utilize such information effectively is still an open challenge in the community. We leave incorporating 3D coordinate information into MoleculeKit as future work.

References

1. Wu, Z. *et al.* MoleculeNet: a benchmark for molecular machine learning. *Chem. science* **9**, 513–530 (2018).

2. Schneider, G. & Wrede, P. Artificial neural networks for computer-based molecular design. *Prog. biophysics molecular biology* **70**, 175–222 (1998).

3. Behler, J. & Parrinello, M. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. review letters* **98**, 146401 (2007).

4. Bartók, A. P., Payne, M. C., Kondor, R. & Csányi, G. Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons. *Phys. review letters* **104**, 136403 (2010).

5. Varnek, A. & Baskin, I. Machine learning methods for property prediction in chemoinformatics: quo vadis? *J. chemical information modeling* **52**, 1413–1437 (2012).

6. Bartók, A. P., Kondor, R. & Csányi, G. On representing chemical environments. *Phys. review B* **87**, 184115 (2013).

7. Duvenaud, D. K. *et al.* Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, 2224–2232 (2015).

8. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th international conference on machine learning*, 1263–1272 (2017).

9. Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R. & Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nat. communications* **8**, 1–8 (2017).

10. Hy, T. S., Trivedi, S., Pan, H., Anderson, B. M. & Kondor, R. Predicting molecular properties with covariant compositional networks. *The journal chemical physics* **148**, 241745 (2018).

11. Unke, O. T. & Meuwly, M. PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges. *J. chemical theory computation* **15**, 3678–3693 (2019).

12. Yang, K. *et al.* Analyzing learned molecular representations for property prediction. *J. chemical information modeling* **59**, 3370–3388 (2019).

13. Unterthiner, T. *et al.* Deep learning as an opportunity in virtual screening. In *Proceedings of the deep learning workshop at NIPS*, vol. 27, 1–9 (2014).

14. Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E. & Svetnik, V. Deep neural nets as a method for quantitative structure–activity relationships. *J. chemical information modeling* **55**, 263–274 (2015).

15. Stokes, J. M. *et al.* A deep learning approach to antibiotic discovery. *Cell* **180**, 688–702 (2020).

16. AI Cures. https://www.aicures.mit.edu/tasks. Accessed:2020-11-25.

17. Weininger, D. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. chemical information computer sciences* **28**, 31–36 (1988).

18. Shervashidze, N., Schweitzer, P., van Leeuwen, E. J., Mehlhorn, K. & Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *J. machine learning research* **12**, 2539–2561 (2011).

19. Kearnes, S., McCloskey, K., Berndl, M., Pande, V. & Riley, P. Molecular graph convolutions: moving beyond fingerprints. *J. computer-aided molecular design* **30**, 595–608 (2016).

20. Coley, C. W., Barzilay, R., Green, W. H., Jaakkola, T. S. & Jensen, K. F. Convolutional embedding of attributed molecular graphs for physical property prediction. *J. chemical information modeling* **57**, 1757–1772 (2017).

21. Fey, M., Yuen, J. G. & Weichert, F. Hierarchical inter-message passing for learning on molecular graphs. In *international conference on machine learning, graph representation learning and beyond workshop* (2020).

22. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. & Watkins, C. Text classification using string kernels. *J. machine learning research* **2**, 419–444 (2002).

23. Cao, D.-S. *et al.* In silico toxicity prediction by support vector machine and smiles representation-based string kernel. *SAR QSAR environmental research* **23**, 141–153 (2012).

24. Bjerrum, E. J. SMILES enumeration as data augmentation for neural network modeling of molecules. *arXiv preprint arXiv:1703.07076* (2017).

25. Mayr, A. *et al.* Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chem. science* **9**, 5441–5451 (2018).

26. Wang, S., Guo, Y., Wang, Y., Sun, H. & Huang, J. SMILES-BERT: large scale unsupervised pre-training for molecular property prediction. In *Proceedings of the 10th ACM international conference on bioinformatics, computational biology and health informatics*, 429–436 (2019).

27. Honda, S., Shi, S. & Ueda, H. R. SMILES Transformer: pre-trained molecular fingerprint for low data drug discovery. *arXiv preprint arXiv:1911.04738* (2019).

28. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

29. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186, DOI: 10.18653/v1/N19-1423 (Association for Computational Linguistics, Minneapolis, Minnesota, 2019).

30. Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S. & Coleman, R. G. ZINC: a free tool to discover chemistry for biology. *J. chemical information modeling* **52**, 1757–1768 (2012).

31. Landrum, G. RDKit: Open-source cheminformatics. http://www.rdkit.org. Accessed:2020-10-01.

32. Battaglia, P. W. *et al.* Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

33. Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y. & Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982* (2020).

34. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

35. Cho, K. *et al.* Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1724–1734, DOI: 10.3115/v1/D14-1179 (Association for Computational Linguistics, Doha, Qatar, 2014).

36. Vaswani, A. *et al.* Attention is all you need. In *Advances in neural information processing systems*, 5998–6008 (2017).

37. Hadsell, R., Chopra, S. & LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, vol. 2, 1735–1742 (IEEE, 2006).

38. Baevski, A., Zhou, Y., Mohamed, A.-r. & Auli, M. wav2vec 2.0: a framework for self-supervised learning of speech representations. *Adv. neural information processing systems* **33** (2020).

39. He, K., Fan, H., Wu, Y., Xie, S. & Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738 (2020).

40. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. A simple framework for contrastive learning of visual representations. *Proceddings 37th international conference on machine learning* (2020).

41. Schütt, K. *et al.* SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Advances in neural information processing systems*, 991–1001 (2017).

42. Klicpera, J., Groß, J. & Günnemann, S. Directional message passing for molecular graphs. In *Proceddings of the 8th international conference on learning representations* (2019).

Acknowledgements

Author contributions

S.J. conceived and initiated the research. Z.W., M.L., Y.L., Z.X., Y.X., L.W., L.C. and S.J. designed the methods. M.L., Y.L., Z.X., Y.X. and L.W. implemented the training and validation methods and designed and developed the software package. Z.W., L.C. and S.J. supervised the project. Z.W., M.L., Y.L., Z.X., Y.X., L.W. and S.J. wrote the manuscript.

Competing Interests

The authors declare no competing interests.

4 Methods

4.1 Notations and problem definitions

Our proposed MoleculeKit aims at making accurate predictions on certain properties of a molecule through graph-based and sequence-based machine learning models. Correspondingly, we represent a molecule as the molecular graph and the simplified molecular-input line-entry system (SMILES) sequence[17]. Here, we give basic notations and define the problems.

As a molecule is composed of chemical atoms and bonds between them, it is natural to represent it as a molecular graph, where nodes and edges correspond to atoms and bonds, respectively. In particular, a bond could be represented as a single undirected edge or two directed edges with opposite directions. Formally, a molecular graph is denoted as $G = (V, E) \in \mathscr{G}$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. $|V|$ and $|E|$ are the numbers of nodes and edges, respectively. Moreover, we consider attributed molecular graphs. Specifically, there are feature vectors associated with each node $v \in V$ and edge $e \in E$. These feature vectors are represented as $x(v) \in \mathbb{R}^p$ and $w(e) \in \mathbb{R}^q$, given by the pre-defined node attribute mapping $x : V \mapsto \mathbb{R}^p$ and edge attribute mapping $w : E \mapsto \mathbb{R}^q$, respectively.

On the other hand, a molecule can be represented as a SMILES sequence as well. Basically, a SMILES sequence is simply a string, where each character denotes a chemical atom or an indicator of structures like a bond, a ring, etc. Formally, we denote a SMILES sequence as $S \in \mathscr{S}$ and its length as $|S|$. And $S[i]$, $i = 1, \ldots, |S|$ refers to the i-th character of S. All the characters come from a vocabulary Σ.

A property of a molecule can be represented by a scalar number y. A continuous $y \in \mathbb{R}$ can denote the water solubility of the molecule. A binary $y \in \{-1, 1\}$ can indicate whether a molecule is antibacterial or not. Depending on y, the problem we focus on can be formulated as either a regression ($y \in \mathbb{R}$) or a binary classification ($y \in \{-1, 1\}$) task. Specifically, given a molecule represented by the molecular graph G or the SMILES sequence S as the input, a machine learning model is asked to predict y.

In the following, we describe our proposed MoleculeKit ensembling both graph-based and sequence-based machine learning methods.

4.2 Graph-based machine learning methods in MoleculeKit

With the molecular graph G as the input, MoleculeKit consists of two graph-based machine learning models; those are, our novel multi-level message passing neural network (ML-MPNN) and the Weisfeiler-Lehman subtree kernel (WL-subtree)[18].

4.2.1 Multi-level message passing neural network

Our proposed multi-level message passing neural network (ML-MPNN) is featured by the ability of aggregating information from a full range of levels in molecular graphs, including nodes, edges, subgraphs, and the entire graph. As a result, multi-level representations are initialized and updated through our ML-MPNN.

Multi-level Representations. We first introduce the multi-level representations and their initialization. In particular, we consider node-level representations, edge-level representations, subgraph-level representations, and a graph-level representation.

As discussed in Section 4.1, the molecular graph $G = (V, E)$ is attributed, with feature vectors $x(v) \in \mathbb{R}^p$ and $w(e) \in \mathbb{R}^q$

associated with each node $v \in V$ and edge $e \in E$. The node-level representations and edge-level representations in our ML-MPNN are then initialized as $x(v) \in \mathbb{R}^p$ and $w(e) \in \mathbb{R}^q$, respectively. Concretely, we follow Yang et al.[12] to define $x(\cdot)$ and $w(\cdot)$. The node attribute mapping $x(\cdot)$ encodes important attributes of a chemical atom, such as the type of atom, number of related bonds, formal charge, and atomic mass. And the edge attribute mapping $w(\cdot)$ encodes useful properties of a bond, including the bond type, the stereo, whether the bond is part of a ring, and whether the bond is conjugated. In the following, we directly use $x(v) \in \mathbb{R}^p$ and $w(e) \in \mathbb{R}^q$ to denote the node-level representations and edge-level representations for simplicity.

ML-MPNN utilizes the junction tree to obtain subgraphs and initialize the subgraph-level representations. The junction tree represents a molecular graph as a tree, where each node in the tree corresponds to a subgraph in the original molecular graph. Specifically, we follow the tree decomposition algorithm[21,43,44] to build the junction tree. Nodes in the resulted junction tree represent rings, bonds, bridged compounds, or singletons in the original molecular graph, as illustrated in Supplementary Fig. 1. Afterwards, in order to initialize subgraph-level representations for the original molecular graph, we simply generate node-level representations for the junction tree. Here, for each node in the junction tree, we encode the concrete subgraph type as its node-level representation. Formally, given a molecular graph G, we denote the node set of the junction tree as C, which is also the set of subgraphs. The subgraph-level representations are denoted by $r(c) \in \mathbb{R}^f$ for each subgraph $c \in C$.

The graph-level representation in our ML-MPNN is empirically initialized as the average of edge-level representations in the corresponding graph. After ML-MPNN, the final graph-level representation obtained is used for predicting the molecular property. Formally, given a molecular graph G, we denote the graph-level representation as $z(G) \in \mathbb{R}^g$.

As a summary, given a molecular graph $G = (V, E)$ and the set of corresponding subgraphs C, the multi-level representations involved in our ML-MPNN are node-level representations $x(v) \in \mathbb{R}^p$ for each node $v \in V$, edge-level representations $w(e) \in \mathbb{R}^q$ for each edge $e \in E$, subgraph-level representations $r(c) \in \mathbb{R}^f$ for each subgraph $c \in C$, and a graph-level representation $z(G) \in \mathbb{R}^g$. Note that ML-MPNN handles molecular graphs with directed edges. For each edge $e \in E$, we use $s_e \in V$ and $t_e \in V$ to denote the source node and target node. In addition, without loss of generality, we consider the nodes and subgraphs are indexed from 1 to $|V|$ and from 1 to $|C|$, respectively. Then the assignment of nodes to subgraphs can be denoted by an assignment matrix $R \in \{0,1\}^{|V| \times |C|}$, where $R_{[v,c]} = 1$ if node v belongs to subgraph c, and $R_{[v,c]} = 0$ otherwise.

ML-MPNN. Next, we introduce how multi-level representations are updated through our ML-MPNN, where message passing is performed hierarchically. ML-MPNN can be viewed as an extension of the general graph neural network framework[32]. ML-MPNN stacks several ML-MPNN layers followed by a final output layer for classification or regression. Each ML-MPNN layer updates the representations for each level in five steps, as illustrated in Supplementary Fig. 2.

Step 1: Update the edge-level representations. We update the edge-level representations by aggregating information from node-level representations of the source node and target node as well as the graph-level representation. Specifically, we concatenate these representations with the previous edge-level representation as the inputs to a multilayer perceptron (MLP) to update the edge-level representation. Formally, we have

$$w'(e) = \text{MLP}\left([w(e), x(s_e), x(t_e), z(G)]\right), \quad \forall e \in E, \tag{1}$$

where $[\cdot]$ denote concatenation and $w'(e)$ represents the updated edge-level representation. Note that, as the graph-level representation contains the global information of the molecular graph, it is supposed to guide the updating process and thus used in each updating step below.

Step 2: Update the node-level representations. For each node, we use updated edge-level representations of incoming edges and node-level representations of corresponding source nodes to compute messages it receives. The node-level representations is then updated with the messages and the graph-level representation. The updating process is formulated as

$$M^{e2n}(v) = \mathrm{MLP}\left(\sum_{\{e:\, t_e=v\}} \left[w'(e), x(s_e) \right] \right), \quad \forall v \in V, \tag{2}$$

$$x'(v) = \mathrm{MLP}\left(\left[x(v), M^{e2n}(v), z(G) \right] \right), \quad \forall v \in V. \tag{3}$$

Here, $x'(v)$ is the updated node-level representation of node v. And $M^{e2n}(\cdot)$ denotes the messages passed from edge-level representations to node-level representations.

Step 3: Update subgraph-level representations. The messages for updating the subgraph-level representations are from nodes that are assigned to the subgraph as well as neighboring subgraphs, *i.e.*, neighboring nodes in the junction tree. The concrete updating procedure is given by

$$M^{n2s}(c) = \mathrm{MLP}\left(\sum_{\{v:\, R_{[v,c]}=1\}} x'(v) \right), \quad \forall c \in C, \tag{4}$$

$$r'(c) = \mathrm{MLP}\left(\left[r(c), M^{n2s}(c), \mathrm{MLP}\left(\sum_{u \in \mathcal{N}(c)} r(u) \right), z(G) \right] \right), \quad \forall c \in C, \tag{5}$$

where $\mathcal{N}(c)$ denotes the set of neighboring subgraphs of subgraph c and $r'(c)$ is the updated subgraph-level representation of subgraph c. And $M^{n2s}(\cdot)$ represents the messages passed from node-level representations to subgraph-level representations.

Step 4: Update the graph-level representation. Finally, the graph-level representation gets updated after receiving messages from updated representations of each level. We have

$$M^{e2g}(G) = \frac{1}{|E|} \sum_{e \in E} w'(e), \tag{6}$$

$$M^{n2g}(G) = \frac{1}{|V|} \sum_{v \in V} x'(v), \tag{7}$$

$$M^{s2g}(G) = \frac{1}{|C|} \sum_{c \in C} r'(c), \tag{8}$$

$$z'(G) = \mathrm{MLP}\left(\left[z(G), M^{e2g}(G), M^{n2g}(G), M^{s2g}(G) \right] \right), \tag{9}$$

where $z'(G)$ is the updated graph-level representation of graph G. And $M^{e2g}(\cdot)$, $M^{n2g}(\cdot)$, and $M^{s2g}(\cdot)$ denote the messages passed from edge-level representations, node-level representations, and subgraph-level representations, respectively.

Step 5: Normalization for multi-level representations. Like modern deep learning models, our ML-MPNN uses batch training. Adding BatchNorm[45] can stabilize the training process and usually achieve better performance. However, due to the various sizes of graphs, batching graphs will lead to representations at different scales, making it difficult to learn the optimal statistics for BatchNorm[33]. As a result, certain normalization methods on sizes must be applied. In ML-MPNN, graphs of various sizes refer to graphs with a variable number of edges, nodes, and subgraphs. Correspondingly, we apply the EdgeSizeNorm, NodeSizeNorm, and SubgraphSizeNorm for edge-level representations, node-level representations, and subgraph-level representations, respectively, given by

$$\bar{w}'(e) = \text{BatchNorm}\left(\text{EdgeSizeNorm}\left(w'(e)\right)\right), \quad \text{EdgeSizeNorm}\left(w'(e)\right) = \frac{w'(e)}{\sqrt{|E|}}, \quad \forall e \in E, \tag{10}$$

$$\bar{x}'(v) = \text{BatchNorm}\left(\text{NodeSizeNorm}\left(x'(e)\right)\right), \quad \text{NodeSizeNorm}\left(x'(v)\right) = \frac{x'(e)}{\sqrt{|V|}}, \quad \forall v \in V, \tag{11}$$

$$\bar{r}'(c) = \text{BatchNorm}\left(\text{SubgraphSizeNorm}\left(r'(c)\right)\right), \quad \text{SubgraphSizeNorm}\left(r'(c)\right) = \frac{r'(c)}{\sqrt{|C|}}, \quad \forall c \in C, \tag{12}$$

$$\bar{z}'(G) = \text{BatchNorm}\left(z'(G)\right). \tag{13}$$

4.2.2 Weisfeiler-Lehman subtree kernel

MoleculeKit applies the Weisfeiler-Lehman subtree kernel (WL-subtree)[18] as a traditional graph-based machine learning component. WL-subtree is able to capture the similarity among molecular graphs regarding the isomorphism. In WL-subtree, we consider molecular graphs with undirected edges. In addition, the node attribute mapping $x : V \mapsto \mathbb{R}^p$ becomes a node label mapping $\ell : V \mapsto \mathbb{Z}^+$ that assigns to each node v the atomic number of the corresponding atom. And there is no edge attribute.

In general, given any two graphs $G_1, G_2 \in \mathcal{G}$, a graph kernel computes the similarity between them as $k_g(G_1, G_2)$, where $k_g : \mathcal{G}^2 \to \mathbb{R}$ is a kernel function. Suppose we have a training dataset $\{(G_i, y_i)\}_{i=1}^N$ with N samples. We train a kernel Support Vector Machine (SVM)[46,47] by optimizing

$$\min_{\{\alpha_i\}_{i=1}^n} C\left[\sum_{i=1}^N L\left(\sum_{j=1}^N \alpha_j y_j K_{ji}, y_i\right)\right] + \frac{1}{2}\sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j K_{ji}, \tag{14}$$

where $K_{ij} = k_g(G_i, G_j)$, C is a hyper-parameter, and $\{\alpha_i\}_{i=1}^N$ are training parameters. For a regression task, L is the ε-insensitive loss[48,49]. After training, the prediction result of a graph G is given by $\sum_{i=1}^N \alpha_i y_i k_g(G_i, G)$. For a binary classification task, L is the hinge loss[46] and the prediction result is given by $\text{sign}\left(\sum_{i=1}^N \alpha_i y_i k_g(G_i, G)\right)$.

In particular, WL-subtree defines a kernel function k_g that computes the similarity based on common subtrees in two graphs. Specifically, let $\{s_1, s_2, \cdots, s_M\}$ be the set of all possible subtrees within a certain depth. For a graph G, we compute

$$\phi(G) = [n(s_1, G), n(s_2, G), \cdots, n(s_M, G)]^T, \tag{15}$$

where $n(s_i, G)$, $i = 1, 2, \ldots, M$ is the number of the subtree s_i in G. With $\phi(\cdot)$, the kernel function between G_1 and G_2 is given

by $k_g(G_1, G_2) = \phi(G_1)^T \phi(G_2)$.

The set $\{s_1, s_2, \cdots, s_M\}$ is obtained by an iterative process of relabeling each node. Concretely, for the t-th iteration, we denote the label of each node v in a graph G as $\ell^{(t)}(v)$. At iteration 0, all the nodes in G are labelled by the node label mapping $\ell^{(0)}(\cdot) = \ell(\cdot)$. At iteration t ($t \geq 1$), we update labels in two steps. First, for each node v, we build a multiset $S(v) = \{\ell^{(t-1)}(u) : u \in \mathcal{N}(v)\}$, where $\mathcal{N}(v)$ is the set of neighboring nodes of v in G. Basically, $S(v)$ collects the current labels of neighboring nodes of v. Note that a multiset is allowed to include repeated elements. Meanwhile, labels in $S(v)$ are sorted. Second, we hash $\{\ell^{(t-1)}(v), S(v)\}$ to a new label s_v. Then we have $\ell^{(t)}(v) = s_v$.

This iterative process is performed on all graphs in the training dataset. Note that, for any pair of different nodes v_1 and v_2, no matter whether they are in the same graph or not, $\ell^{(t)}(v_1) = \ell^{(t)}(v_2)$ if and only if the two t-hop subtrees rooted at v_1 and v_2 are identical. As a result, each label denotes a unique subtree. After T iterations, we collect all the labels that ever appear in any graph during the T iterations and put them into $\{s_1, s_2, \cdots, s_M\}$, a set of all possible subtrees within T hops. Finally, to obtain $n(s_i, G)$, $i = 1, 2, \ldots, M$ for any graph G, we simply perform the T iterations on G and count the times of s_i appearing.

4.3 Sequence-based machine learning methods in MoleculeKit

With the SMILES sequence S as the input, MoleculeKit consists of two sequence-based machine learning models; those are, our proposed contrastive-BERT and the subsequence kernel[22].

4.3.1 Contrastive-BERT

Our contrastive-BERT is characterized by a novel self-supervised pre-training task via contrastive learning[37–40], namely masked embedding recovery task. It results in better performances on downstream applications after fine-tuning. Note that the pre-training phase is performed only once on a large unlabeled dataset and we have provided the pre-trained model. The fine-tuning phase is the same as end-to-end supervised training on labeled datasets, except for using the pre-trained model to initialize training parameters instead of random initialization. In the following, we first describe the network architecture of our contrastive-BERT and then focus on the proposed masked embedding recovery task.

Network architecture. Similar to the original BERT[29], the network architecture of our contrastive-BERT mainly follows Transformer[36]. Given a SMILES sequence S with the vocabulary Σ, a trainable embedding layer first is applied to transform each character $S[i]$, $i = 1, 2, \ldots, |S|$ into an embedding vector $h_i \in \mathbb{R}^d$. The embedding layer actually maintains an embedding matrix of dimension $|\Sigma| \times d$, where each row corresponds to the embedding vector of a character in Σ. The embedding vector of a character is supposed to capture chemical information about the corresponding atom or structure. In addition, since the order matters in SMILES sequences, the position embedding from Vaswani et al.[36] is added to h_i. Afterwards, h_i serves as the inputs to Transformer and gets updated.

Transformer is composed of a stack of Transformer layers. A Transformer layer consists of a self-attention mechanism followed by a multilayer perceptron (MLP). Specifically, with h_i, $i = 1, 2, \ldots, |S|$ as inputs, the self-attention mechanism allows direct interactions among them and updates each h_i by aggregating information according to the interaction results. Formally,

by writing h_i, $i = 1, 2, \ldots, |S|$ into a matrix $H = [h_1, h_2, \ldots, h_{|S|}]^T \in \mathbb{R}^{|S| \times d}$, we have

$$H' = \text{Softmax} \left(\frac{(HW^Q)(HW^K)^T}{\sqrt{d}} \right) HW^V, \tag{16}$$

where $H' = [h'_1, h'_2, \ldots, h'_{|S|}]^T \in \mathbb{R}^{|S| \times d}$ represents the updated embeddings and $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ are training parameters. In particular, the multi-head technique[36] is also used. The self-attention mechanism is similar to a global message passing in the sense that each character can receive information from any other character. The following MLP has two layers with the Gaussian error linear unit (GELU)[50] as the activation function. In order to facilitate the training, residual connections[51] and the layer normalization[52] are applied on both the self-attention mechanism and the MLP.

Consequently, the outputs of our contrastive-BERT are updated embedding vectors for each character of a SMILES sequence S, denoted as o_i, $i = 1, 2, \ldots, |S|$. In practice, a special <TASK> token is inserted to S as the first character $S[1]$. The output embedding vector of <TASK> will be used for prediction, as explained below.

Masked embedding recovery task. As described above, the training of the contrastive-BERT consists of a self-supervised pre-training phase followed by a supervised fine-tuning phase, as shown in Supplementary Fig. 4. Our contrastive-BERT mainly contributes to the self-supervised pre-training phase by proposing the masked embedding recovery task, a novel self-supervised task via contrastive learning.

Basically, the masked embedding recovery task asks the model to predict the embeddings of masked characters based on embeddings of unmasked characters. The masking strategy in our contrastive-BERT follows the one in the original BERT[29], in order to fairly demonstrate the advantages of the new objective. Concretely, in a SMILES sequence, 15% of the characters are randomly masked. For each character that is selected to be masked, it has 80% probability to be replaced by a special <MASK> token, 10% probability to be replaced by another random character, and 10% probability to remain unchanged. For each SMILES sequence, at least one character is masked.

As a result, if a character $S[i]$ is masked, the corresponding input embedding h_i is replaced and will not be fed into the network. The masked embedding recovery task is to predict the output embedding o_i without h_i. However, there is no ground truth for o_i, which means a common supervised objective cannot be used. In order to perform the pre-training, our contrastive-BERT designs a self-supervised objective via contrastive learning. In particular, for a masked character $S[i]$, we force the predicted o_i to be similar to h_i, as they correspond to the same character. In contrast, o_i needs to be less similar to $h_j, \forall j \neq i$. Note that, while h_i is replaced from the inputs to the network, it can still be retrieved from the embedding matrix. Given a function to measure the similarity, the pre-training can be achieved using a contrastive loss, as illustrated in Supplementary Fig. 5.

After the pre-training phase with the masked embedding recovery task, our contrastive-BERT is fine-tuned on labeled datasets for downstream applications. Specifically, a predictor is added on top of the embedding vector of <TASK> to predict the molecular property. The fine-tuning is an end-to-end training process with a supervised loss. In both training phases, we use a data augmentation technique based on a characteristic of SMILES sequences. That is, one molecule may be represented by

multiple different SMILES sequences, although each SMILES sequence only corresponds to a certain molecule. For example, *CCN*, *NCC*, and *C(N)C* are all valid SMILE sequences of the organic compound ethylamine with the formula $CH_3CH_2NH_2$ (Supplementary Fig. 3). Each time we perform training on a molecule, we randomly choose a valid SMILES sequence as inputs to our contrastive-BERT. Such data augmentation helps the model learn embeddings that capture intrinsic chemical information irrelevant to the representation formats. During testing, we use the canonicalization algorithm[53] to generate a unique canonical SMILES sequence for each molecule.

4.3.2 Subsequence kernel

As a traditional sequence-based machine learning component, our MoleculeKit uses the subsequence kernel[22]. As indicated by its name, the subsequence kernel computes the similarity between SMILES sequences based on their common subsequences. Formally, for a SMILES sequence $S \in \mathscr{S}$, a sequence U is called a subsequence of S if there exists a list of indices $\boldsymbol{i} = \{i_1, i_2, \cdots, i_{|U|} : 1 \leq i_1 < i_2 < \cdots < i_{|U|} \leq |S|\}$ such that $U[j] = S[i_j]$, $j = 1, \cdots, |U|$, *i.e.*, $U = S[\boldsymbol{i}]$. Note that the subsequence U is not necessarily composed of consecutive characters from S.

Similar to the graph kernel described in Section 4.2.2, a kernel function $k_s : \mathscr{S}^2 \to \mathbb{R}$ is defined to compute the similarity between two sequences $S_1, S_2 \in \mathscr{S}$ as $k_s(S_1, S_2)$. Given a training dataset $\{(S_i, y_i)\}_{i=1}^N$, the training process solves the same optimization problem in Equation (14) with $K_{ij} = k_s(S_i, S_j)$. With the learned parameters $\{\alpha_i\}_{i=1}^N$, the prediction result of an sequence S is given by $\sum_{i=1}^N \alpha_i y_i k_s(S_i, S)$ and sign $\left(\sum_{i=1}^N \alpha_i y_i k_s(S_i, S)\right)$ for regression and binary classification, respectively.

Concretely, the subsequence kernel focuses on subsequences with a pre-defined length D. For SMILES sequences with the vocabulary Σ, the set of all possible subsequences of length D is $\Sigma^D = \{U_1, U_2, \ldots, U_{|\Sigma^D|}\}$. For a sequence S, we compute

$$n(U, S) = \sum_{\boldsymbol{i}:U=S[\boldsymbol{i}]} \lambda^{i_D - i_1 + 1}, \tag{17}$$

$$\phi(S) = \left[n(U_1, S), n(U_2, S), \cdots, n(U_{|\Sigma^D|}, S) \right]^T, \tag{18}$$

where $n(U_i, G)$, $i = 1, 2, \ldots, |\Sigma^D|$ is a weighted count of U_i in S. Specifically, λ with $0 < \lambda < 1$ is introduced as a decay factor, so that a subsequence is given a smaller weight if its characters are further away from each other in the sequence S. With $\phi(\cdot)$, the kernel function k_s between two SMILES sequences S_1 and S_2 is given by $k_s(S_1, S_2) = \phi(S_1)^T \phi(S_2)$. In practice, it is common to normalize it by $\frac{k_s(S_1, S_2)}{\sqrt{k_s(S_1, S_1) \cdot k_s(S_2, S_2)}}$.

4.4 Training losses

We introduce the loss functions to train the deep learning components of MoleculeKit, *i.e.*, our proposed ML-MPNN and contrastive-BERT. Note that the contrastive-BERT has an extra one-time self-supervised pre-training phase with a contrastive loss, as described in Section 4.3.1. The fine-tuning phase of the contrastive-BERT uses the same supervised losses as ML-MPNN. For all the training of deep learning models, we employ the batch training, where we randomly select a batch of B samples from the training dataset in each training iteration. The Adam[54] optimizer is used to minimize the loss.

4.4.1 Supervised loss for regression tasks

In regression tasks, we use the mean squared error (MSE) loss:

$$L_{MSE}(\{y_i\}_{i=1}^B, \{\hat{y}_i\}_{i=1}^B) = \frac{1}{B}\sum_{i=1}^B (y_i - \hat{y}_i)^2, \tag{19}$$

where $y_i, \hat{y}_i \in \mathbb{R}$ are the ground-truth label and prediction result of the i-th training sample in the batch, respectively.

4.4.2 Supervised loss for binary classification tasks

In binary classification tasks, we use the binary cross entropy (BCE) loss:

$$L_{BCE}(\{y_i\}_{i=1}^B, \{\hat{y}_i\}_{i=1}^B) = -\frac{1}{B}\sum_{i=1}^B \left(\mathbb{1}_{\{y=1\}}(y_i)\log(\hat{y}_i) + \mathbb{1}_{\{y=-1\}}(y_i)\log(1-\hat{y}_i) \right), \tag{20}$$

where $y_i \in \{-1, 1\}$ is the ground-truth label of the i-th training sample in the batch, $\hat{y}_i \in (0, 1)$ is the corresponding predicted classification score, and $\mathbb{1}_{\{y=1\}}(\cdot)$ and $\mathbb{1}_{\{y=-1\}}(\cdot)$ are indicator functions.

4.4.3 Self-supervised loss for the masked embedding recovery task

The masked embedding recovery task uses a self-supervised contrastive loss. Given a SMILES sequence S, the input and output embeddings of the contrastive-BERT are h_i and o_i, $i = 1, 2, \ldots, |S|$, respectively. If $S[i]$ is masked, the corresponding contrastive loss is given by

$$L_{contrast}(o_i, h_1, h_2, \ldots, h_{|S|}) = -\log \frac{\exp(sim(o_i, h_i)/\tau)}{\sum_{j=1}^{|S|} \exp(sim(o_i, h_j)/\tau)}, \tag{21}$$

where τ is a temperature parameter and the similarity function $sim(\cdot, \cdot)$ computes the cosine similarity, defined by

$$sim(o, h) = \frac{o^T h}{|o| \cdot |h|}. \tag{22}$$

If a SMILES sequence has multiple masked characters, the final loss is the averaged contrastive loss.

4.5 Evaluation metrics

As the molecular property prediction problem can be formulated as with a regression or a binary classification task, we use different evaluation metrics for different datasets.

4.5.1 Regression tasks

Given a testing dataset of N samples for a regression task, we denote the ground-truth labels as $\{y_i\}_{i=1}^n$ and the corresponding prediction results as $\{\hat{y}_i\}_{i=1}^N$, where $y_i, \hat{y}_i \in \mathbb{R}$. The evaluation metric is mean absolute error (MAE)[55]:

$$MAE(\{y_i\}_{i=1}^N, \{\hat{y}_i\}_{i=1}^N) = \frac{1}{N}\sum_{i=1}^N |y_i - \hat{y}_i|, \tag{23}$$

or root-mean-square error (RMSE):

$$RMSE(\{y_i\}_{i=1}^N, \{\hat{y}_i\}_{i=1}^N) = \sqrt{\frac{1}{N}\sum_{i=1}^N (y_i - \hat{y}_i)^2}. \tag{24}$$

A smaller MAE or RMSE indicates a better performance.

4.5.2 Binary classification tasks

Given a testing dataset of N samples for a binary classification task, we denote the ground-truth labels as $\{y_i\}_{i=1}^n$ and the corresponding predicted classification scores as $\{\hat{y}_i\}_{i=1}^N$, where $y_i \in \{-1, 1\}$ and $\hat{y}_i \in [0, 1]$. The predicted classification score $\hat{y}_i$ can be transformed into the predicted label $\tilde{y}_i \in \{-1, 1\}$ with a threshold γ, where $\tilde{y}_i = 1$ if $\hat{y}_i \geq \gamma$ and $\tilde{y}_i = -1$ if $\hat{y}_i < \gamma$.

If $\tilde{y}_i = y_i$ and $\tilde{y}_i = 1$ (or $\tilde{y}_i = -1$), we say the i-th prediction is a true positive (or true negative); if $\tilde{y}_i \neq y_i$ and $\tilde{y}_i = 1$ (or $\tilde{y}_i = -1$), we say the i-th prediction is a false positive (or false negative). With the threshold γ, we use $TP_\gamma, TN_\gamma, FP_\gamma, FN_\gamma$ to represent the number of true positives, true negatives, false positives and false negatives, respectively. And we can compute the true positive rate (TPR), false positive rate (FPR), and positive predictive value (PPV) under γ as:

$$TPR_\gamma = \frac{TP_\gamma}{TP_\gamma + FN_\gamma}, \tag{25}$$

$$FPR_\gamma = \frac{FP_\gamma}{FP_\gamma + TN_\gamma}, \tag{26}$$

$$PPV_\gamma = \frac{TP_\gamma}{TP_\gamma + FP_\gamma}. \tag{27}$$

Note that TPR and PPV are also known as recall and precision.

The evaluation metric for binary classification tasks is the area under curve (AUC) of the receiver operating characteristic (ROC) curve[56] or the precision-recall curve (PRC)[57], denoted by ROC-AUC and PRC-AUC. Concretely, by varying $\gamma \in [0, 1]$, the ROC curve and PRC plot how (FPR, TPR) and (recall, precision) change in a *xy*-coordinate plane, respectively. A larger ROC-AUC or PRC-AUC indicates a better performance.

4.6 Configurations

The settings of our device are - GPU: Nvidia GeForce RTX 2080 Ti 11GB; CPU: Intel Xeon Silver 4116 2.10GHz; OS: Ubuntu 16.04.3 LTS.

We implement our proposed ML-MPNN in PyTorch[58] and Pytorch Geometric[59]. Following D-MPNN[12], we incorporate additional global molecular features that can be easily obtained with the open-source package RDKit[31]. Each molecule has a 200-dimensional global molecular feature, which is concatenated with the final graph-level representation learned by our ML-MPNN as inputs to a 2-layer MLP to make the prediction. The task-specific configurations include the settings of initial learning rate, learning rate decay factor, weight decay, hidden dimension, dropout rate, and batch size, as listed in Supplementary Table 7.

Our contrastive-BERT model is also implemented in PyTorch. The network is composed of 6 Transformer layers with 1024 hidden units and 4 attention heads. For the one-time pre-training phase, we train the model for 10 epochs, with the initial learning rate set to 0.0001 and decay with the cosine annealing strategy. The batch size is set to 128 and the dropout rate is 0.1. In the fine-tuning phase, the settings are task-specific. Specifically, we tune batch size, the number of training epochs and learning rate for each dataset, as listed in Supplementary Table 8.

As for kernel methods, the number of iterations of graph kernels, the subsequence length and the decay factor of string kernels, and whether kernel normalization is used are task-specific. The setting of these hyperparameters for each dataset is summarized in Supplementary Table 9.

Reporting summary

Further information on research design can be found in the Nature Research Reporting Summary linked to this article.

Data availability

The dataset for AI Cures open challenge task[16] is available at `https://www.aicures.mit.edu/data`. The dataset for antibiotic discovery, provided by Stokes et al.[15], can be found at `https://www.sciencedirect.com/science/article/pii/S0092867420301021`. The datasets and splits of MoleculeNets benchmarks[1] for molecular property prediction can be downloaded from `http://deepchem.io/datasets/MolNet_bkp.zip` and `http://deepchem.io/trained_models/Hyperparameter_MoleculeNetv3.tar.gz`. The dataset of 2 million molecules from ZINC[30], prepared by Hu et al.[60], for pretraining our contrastive-Bert can be downloaded from `http://snap.stanford.edu/gnn-pretrain/data/chem_dataset.zip`.

We also collect datasets and corresponding splits of MoleculeNet benchmarks and the dataset of 2 million molecules from ZINC at `https://github.com/divelab/MoleculeKit/tree/master/moleculekit/datasets` in our MoleculeKit tool. We recommend our users to use this collection for convenience.

Code availability

The code for MoleculeKit training, prediction and evaluation (in Python/PyTorch), is publicly available at `https://github.com/divelab/MoleculeKit`.

References

43. Rarey, M. & Dixon, J. S. Feature trees: a new molecular similarity measure based on tree matching. *J. computer-aided molecular design* **12**, 471–490 (1998).

44. Jin, W., Barzilay, R. & Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th international conference on machine learning*, 2323–2332 (2018).

45. Ioffe, S. & Szegedy, C. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32th international conference on machine learning*, 448–456 (2015).

46. Cortes, C. & Vapnik, V. Support-vector networks. *Mach. learning* **20**, 273–297 (1995).

47. Chang, C.-C. & Lin, C.-J. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems technology (TIST)* **2**, 1–27 (2011).

48. Vapnik, V. N. *The Nature of Statistical Learning Theory* (Springer-Verlag, Berlin, Heidelberg, 1995).

49. Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J. & Vapnik, V. Support vector regression machines. In *Advances in neural information processing systems*, 155–161 (1997).

50. Hendrycks, D. & Gimpel, K. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415* (2016).

51. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).

52. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

53. Neglur, G., Grossman, R. L. & Liu, B. Assigning unique keys to chemical compounds for data integration: some interesting counter examples. In *International workshop on data integration in the life sciences*, 145–157 (Springer, 2005).

54. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. In *Procceddings of the 3rd international conference on learning representations* (2015).

55. Willmott, C. J. & Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Clim. research* **30**, 79–82 (2005).

56. Fawcett, T. An introduction to ROC analysis. *Pattern recognition letters* **27**, 861–874 (2006).

57. Davis, J. & Goadrich, M. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on machine learning*, 233–240 (2006).

58. Paszke, A. *et al.* PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, 8026–8037 (2019).

59. Fey, M. & Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *international conference on learning representations, workshop on representation learning on graphs and manifolds* (2019).

60. Hu, W. *et al.* Strategies for pre-training graph neural networks. In *Procceddings of the 7th international conference on learning representations* (2019).