Chemprop: Machine Learning Package for Chemical Property Prediction

Esther Heid,[1,2] Kevin P. Greenman,[1] Yunsie Chung,[1] Shih-Cheng Li,[1,3] David E. Graff,[4,1] Florence H. Vermeire,[1,5] Haoyang Wu,[1] William H. Green,[1] and Charles J. McGill[1,6,a)]

[1)] *Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, United States*

[2)] *Institute of Materials Chemistry, TU Wien, 1060 Vienna, Austria*

[3)] *Department of Chemical Engineering, National Taiwan University, Taipei 10617, Taiwan*

[4)] *Department of Chemistry and Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, United States*

[5)] *Department of Chemical Engineering, KU Leuven, Celestijnenlaan 200F, B-3001 Leuven, Belgium*

[6)] *Department of Chemical and Life Science Engineering, Virginia Commonwealth University, Richmond, Virginia 23284, United States*

Deep learning has become a powerful and frequently employed tool for the prediction of molecular properties, thus creating a need for open-source and versatile software solutions that can be operated by non-experts. Among current approaches, directed message-passing neural networks (D-MPNNs) have proven to perform well on a variety of property prediction tasks. The software package Chemprop implements the D-MPNN architecture, and offers simple, easy, and fast access to machine-learned molecular properties. Compared to its initial version, we present a multitude of new Chemprop functionalities such as the support of multi-molecule properties, reactions, atom/bond-level properties, and spectra. Further, we incorporate various uncertainty quantification and calibration methods along with related metrics, as well as pretraining and transfer learning workflows, improved hyperparameter optimization, and other customization options concerning loss functions or atom/bond features. We benchmark D-MPNN models trained using Chemprop with the new reaction, atom-level and spectra functionality on a variety of property prediction datasets, including MoleculeNet and SAMPL, and observe state-of-the-art performance on the prediction of water-octanol partition coefficients, reaction barrier heights, atomic partial charges, and absorption spectra. Chemprop enables out-of-the-box training of D-MPNN models for a variety of problem settings in a fast, user-friendly, and open-source software.

I. INTRODUCTION

Machine learning in general, and especially deep learning, has become a powerful tool in various fields of chemistry. Applications range from the prediction of physico-chemical[1–9] and pharmacological[10] properties of molecules to the design of molecules or materials with certain properties,[11–13] the exploration of chemical synthesis pathways,[14–27] or the prediction of properties important for chemical analysis like IR,[28] UV/VIS[29] or mass spectra.[30–33]

Many combinations of molecular representation and model architecture have been developed to extract features from molecules and predict molecular properties. Molecules can be represented as graphs, strings, precomputed feature vectors, or sets of atomic coordinates and processed using graph-convolutional neural networks, transformers, or feed-forward neural networks to train predictive models. While early works focused on hand-made features or simple fingerprinting methods like circular fingerprints combined with kernel regression methods or neural networks,[34] the current state-of-the-art has shifted to end-to-end trainable models which directly learn to extract their own features.[35] Here, the model complexity can be nearly endless based on the mechanisms of information exchange between parts of the molecule. For example, graph convolutional neural networks (GCNNs) extract local information from the molecular graph for single or small groups of atoms, and use that information to update the immediate neighborhood.[1–3,36] Graph attention transformers allow for a less local information exchange via attention layers, which learn to accumulate the features of atoms both close and far away in the graph.[37,38] Another important line of research comprises the prediction of properties dependent on the three-dimensional conformation of a molecule, such as the prediction of properties obtained from quantum mechanics.[39–42] Finally, transformer models from natural language processing can be trained on string representations such as SMILES or SELFIES, also leading to promising results.[43–46]

In general, larger models with more parameters suffer from less restrictions in what and how they can learn from data, but require larger datasets and pre-training.[47] Smaller models can be very efficient if the right features are known and used,[47] but usually suffer from limitations to their performance or generalization caused by a suboptimal representation or model architecture. For example, models focusing only on the local structure of an atom and its immediate neighborhood usually fail to learn long-range interactions in a detailed fashion.[48] In this work, we focus on GCNNs, which predict many chemical properties accurately[49] at a moderate model size. They offer robust performance if the three-

[a)] Electronic mail: mcgillc2@vcu.edu

dimensional conformation of a molecule is not known or not relevant for a prediction task. They perform best for medium- to large-sized datasets (thousands to hundred thousands of data points). Many flavors of GCNNs have been published, among them our own approach called Chemprop,[36] a directed-message passing algorithm derived from the seminal work of Gilmer et al.[1]

An early version of Chemprop was published in Ref. 36. Since then, the code has substantially evolved, and now includes a vast collection of new features ranging from support of inputs beyond single molecules/targets and scalar molecular properties, to uncertainty estimation, customized atom and bond features, and transfer learning, among others. For example, Chemprop is now able to predict properties for systems containing multiple molecules, such as solute/solvent combinations, or reactions with and without solvent. It can train on molecular targets, spectra, or atom/bond-level targets, and output the latent representation for analysis of the learned feature embedding. Available uncertainty metrics include popular approaches such as ensembling, mean-variance estimation and evidential learning. Chemprop is thus a general and versatile deep learning toolbox, and enjoys a wide user base. Several studies have been published based on Chemprop models ranging from topics such as drug discovery,[10,50,51] to spectroscopic properties,[28,29] physico-chemical properties,[4,52–58] and reaction properties,[59,60] with many of them making use of the new features described in this manuscript, and achieving state-of-the-art performances in their respective tasks.

In this manuscript, we describe and benchmark the new Chemprop features, as well as give usage examples and general advice on using Chemprop. The remainder of the article is structured as follows: The methods section summarizes the architecture of Chemprop, as well as the model details and parameters used in this study. We furthermore describe the data acquisition, preprocessing and splitting of all datasets used in this study. We then discuss a selection of Chemprop features, with a focus on features introduced after the initial release of Chemprop. In the results section, we benchmark Chemprop on a large variety of datasets showcasing its performance for both simple and advanced prediction tasks. We then conclude the article, and provide details on the data and software, which we open-sourced including all scripts to allow for full reproducibility.

II. METHODS

In the following, we describe the general architecture of Chemprop and a selection of the hyperparameters. We then describe the hyperparameter tuning procedure for our benchmark studies. Finally, the source, splitting routines and further information on all benchmark datasets employed in this study are given.

A. Model structure

In general, Chemprop consists of four modules, namely 1) a local features extraction function, 2) a directed message passing neural network (D-MPNN) to learn atomic embeddings from the local features, 3) an aggregation function to join atomic into molecular embeddings, as well as 4) a standard feed-forward neural network (FFN) for the transformation of molecular embeddings to target properties, summarized in Fig. 1. The D-MPNN is a class of graph-convolutional neural network (GCNN), which updates hidden representations of the vertices V and edges E of a graph G based on the local environment. In the following, we will use bold lower case to denote vectors, bold upper case to denote matrices, and italic light font for scalars and objects.

For a molecule, the SMILES string of a molecule is used as input, which is then transformed to a molecular graph using RDKit[61], where atoms correspond to vertices, and bonds to edges. Initial features are constructed based on the identity and topology of each atom and bond. For each vertex v, initial feature vectors $\{\mathbf{x}_v|v \in V\}$ are obtained from a one-hot encoding of the atomic number, number of bonds linked to each atom, formal charge, chirality, number of hydrogens, hybridization and aromaticity of the atom, as well as the scaled atomic mass. These atom features are represented in red on the left side of Fig. 1. For each edge e, initial feature vectors $\{\mathbf{e}_{vw}|\{v,w\} \in E\}$ arise from the bond type, whether the bond is conjugated or in a ring, and whether it contains stereochemical information such as a cis/trans double bond. Since a D-MPNN utilizes directed edges in a graph to pass information, the undirected edge features are then transformed into directed edge features, where each undirected bond in the molecule is represented by two directed edges. Initial directed edge features $\mathbf{e}^{\mathrm{d}}_{vw}$ are obtained via simple concatenation of the atom features of the first atom of a bond $\mathbf{x}_v$ to the respective undirected bond features $\mathbf{e}_{vw}$ (which are the same as $\mathbf{e}_{wv}$)

$$\mathbf{e}^{\mathrm{d}}_{vw} = \mathrm{cat}(\mathbf{x}_v, \mathbf{e}_{vw})) \tag{1}$$

where cat() denotes simple concatenation. Bond features are depicted in blue on the left side of Fig. 1. Chemprop furthermore offers the option to read in custom atom and bond features in addition to or as a replacement of the aforementioned default features, as described later in this manuscript, and thus offers full control of the initial features. In summary, Module 1 of Chemprop constructs atom and directed bond feature vectors $\mathbf{x}_v$ and $\mathbf{e}^{\mathrm{d}}_{vw}$ from the input molecules.

The initial atom and bond features are then passed to a D-MPNN. To construct directed edge features $\mathbf{h}^0_{vw}$ of hidden size h, the initial directed edge features $\mathbf{e}^{\mathrm{d}}_{vw}$ are passed through a single neural network layer with learnable weights $\mathbf{W}_i \in \mathbb{R}^{h \times h_i}$

$$\mathbf{h}^0_{vw} = \tau(\mathbf{W}_i \mathbf{e}^{\mathrm{d}}_{vw}) \tag{2}$$

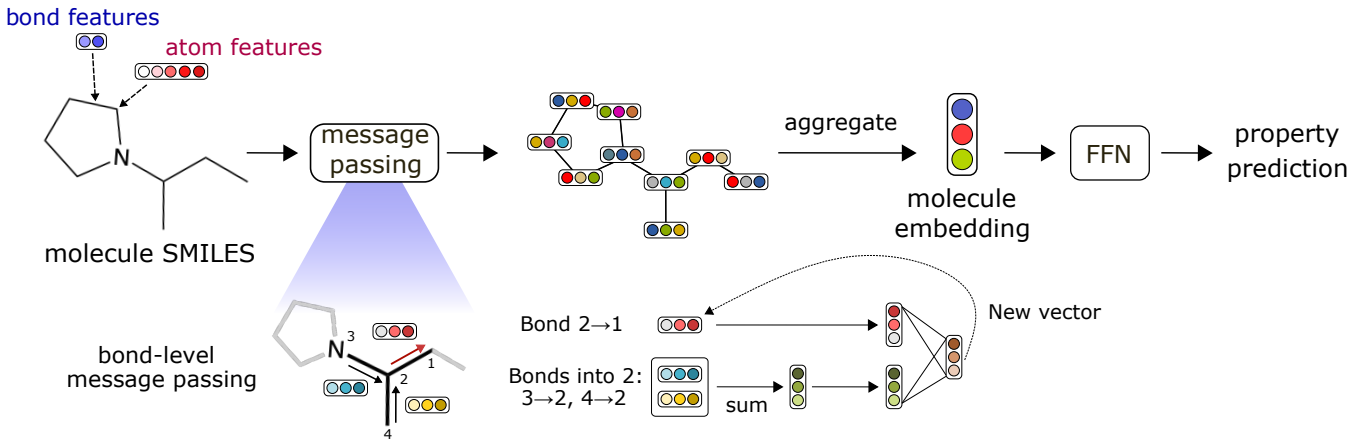

Figure 1. Overview of the architecture of Chemprop.

and a nonlinear activation function τ which can be chosen by the user (default ReLU). The size h of $\mathbf{h}_{vw}^0$ can be chosen by the user (default 300). The size of $\mathbf{e}_{vw}^{\mathrm{d}}$, which we term h_i, is set by the feature extraction routine in Module 1 (default 147). The directed edge features are then iteratively updated based on the local environment via T (default 3) message passing steps

$$\mathbf{h}_{vw}^{t+1} = \tau(\mathbf{h}_{vw}^0 + \mathbf{W}_h \sum_{k \in \{N(v) \backslash w\}} \mathbf{h}_{kv}^t) \tag{3}$$

until $t+1 = T$, where $\mathbf{W}_h \in \mathbb{R}^{h \times h}$ and $N(v)\backslash w$ denotes the neighbors of node v excluding w. Finally, the updated hidden states $\mathbf{h}_{vw}^T$ are aggregated into atomic embeddings via

$$\mathbf{h}_v = \tau(\mathbf{W}_o \mathbf{q}) \tag{4}$$

where $\mathbf{q}$ is a concatenation of the initial atom features $\mathbf{x}_v$ and the sum of all incoming directed edge hidden states

$$\mathbf{q} = \mathrm{cat}\,(\mathbf{x}_v, \sum_{w \in N(v)} \mathbf{h}_{wv}^T) \tag{5}$$

with $\mathbf{W}_o \in \mathbb{R}^{h \times h_o}$. Here, h_o is the size of $\mathbf{q}$, i.e. the sum of the hidden size h and the size of $\mathbf{x}_v$. In summary, in Module 2, the D-MPNN weights $\mathbf{W}_i$, $\mathbf{W}_h$, and $\mathbf{W}_o$ are learned from the training data, outputting learnable atomic embeddings $\mathbf{h}_v$. Customizations and hyperparameter tuning include the choice of activation function τ, the hidden size h, and the number of message passing steps T. Chemprop furthermore offers the option to add bias terms to all neural network layers (defaults to False).

The atomic embeddings $\mathbf{h}_v$ of all atoms in a molecule are then aggregated into a single molecular embedding $\mathbf{h}_m$ via

$$\mathbf{h}_m = \mathrm{cat}\,(\mathbf{h}'_m, \mathbf{x}_m) \tag{6}$$

with

$$\mathbf{h}'_m = \sum_{v \in V} \mathbf{h}_v \tag{7}$$

where $\mathbf{x}_m$ is an optional vector of precomputed molecular features. Chemprop offers different aggregation functions instead of the simple sum over all atoms shown in Eq. (7), namely a scaled sum (called norm within Chemprop) or an average over all atoms (defaults to average). Schweidtmann et al.[62] compared the performance of such aggregation functions for different datasets. Furthermore, various fingerprints $\mathbf{x}_m$ such as binary or count Morgan circular fingerprints[63], or normalized RDKIT[61] features, as implemented in Ref. 64 are directly available in Chemprop. Custom precomputed fingerprints or additional outside information can also be used. Per default, $\mathbf{x}_m$ is empty such that the molecular embedding is simply an aggregation over atomic embeddings, i.e. $\mathbf{h}_m = \mathbf{h}'_m$. In summary, Module 3 produces molecular embeddings $\mathbf{h}_m$ of length h plus the size of $\mathbf{x}_m$. Chemprop furthermore offers the option to circumvent Modules 1-3 and only using $\mathbf{x}_m$ as fixed molecular embedding, so that $\mathbf{h}_m = \mathbf{x}_m$.

Finally, in the last module, molecular target properties are learned from the molecular embeddings $\mathbf{h}_m$ via a feed-forward neural network, where the number of layers (default 2) and the number of hidden neurons (default 300) can be chosen by the user. The number of input neurons is set by the length of $\mathbf{h}_m$, and the number of output neurons by the number of targets (and uncertainty metrics if mean-variance estimation or evidential learning is requested). The activation function is set to be the same as in the D-MPNN, and bias is turned on per default.

Chemprop is fully end-to-end trainable, so that the weights for the D-MPNN and FFN are updated simultaneously. Various loss functions for regression and classification tasks, error and uncertainty metrics, regularization strategies like dropout, learning rate schemes, or batch size can be set by the user, along with options for data splitting, ensembling and cross-validation. By default, a single model is trained on a single random split for 30 epochs. We note that small datasets need a much larger number of epochs to train, and advise to check for

convergence of the learning curve. Chemprop uses the Adam optimizer.[65] The default learning rate schedule increases the learning rate linearly from 10^{-4} to 10^{-3} for the first two warmup epochs and then decreases learning rate exponentially from 10^{-3} to 10^{-4} for the remaining epochs. By default, a batch size of 50 data points is used for each optimizer step. For regression tasks, the loss function defaults to the mean squared error and the evaluation metric to the root mean squared error. For classification tasks, the loss function defaults to the binary cross entropy and the evaluation metric to the area under the curve of the receiver-operator characteristic. For multiclass classification, both loss and metric correspond to the cross entropy. By default, no gradient clipping or dropout is performed. If available, training is performed on a single GPU, else on CPUs (defaults to 8 cores), and data loading is performed in parallel on CPUs with the same number of workers.

For example, to train a default model on the ESOL solubility dataset[86] which is distributed with Chemprop as CSV file, and save the results to the folder"checkpoint", run

```
chemprop_train --data_path data/delaney.csv
--dataset_type regression --save_dir checkpoint
--save_smiles_splits
```

on the command line after installation of Chemprop following the instructions on Github.[87] This splits the data randomly into training, validation and test sets in the ratio 80/10/10, trains a default model and computes the performance on the test set. To compute predictions using an already trained model, run

```
chemprop_predict --checkpoint_dir checkpoint
--test_path checkpoint/fold_0/test_smiles.csv
--preds_path checkpoint/test_preds.csv
```

which takes the previously generated test set, computes predictions using all models in the checkpoint folder and saves them to the indicated path. For the use of Chemprop within a Python script or a graphical web interface, as well as many options to customize the model, data splits, and performance metrics please consult the instructions on Github[87] or the Chemprop documentation.[88] Hyperparameter optimization can be performed with similar commands, as detailed in the Discussion of Features section.

B. Benchmarking hyperparameter tuning

Training of benchmark models was carried out using hyperparameters optimized for each task. Throughout the remainder of this study, we classify datasets as small/large if they contain less/more than 10k data points in total. Models trained on small datasets were optimized for hyperparameters using 100 search iterations, whereas models trained on large datasets were optimized for only 30 iterations. During hyperparameter tuning and final model training, we trained for 200/50 epochs for small/large datasets. All models were trained on a single data split with an ensemble size of 5 during the final training, and without ensembling for hyperparameter tuning. During hyperparameter tuning, we optimized for the number of message passing steps, the hidden size during message passing, the number of layers of the feed forward neural network, as well as its hidden size, and the dropout ratio. For small datasets, furthermore the learning rate (initial, final and maximum), warum-up period and batch size were optimized. For both hyperparameter tuning and model production, scaled sums were used to aggregate atomic into molecular feature vectors. All other parameters were left at their default values.

C. Benchmarking datasets

The benchmarking datasets used in this study are listed in Table I. All datasets are publicly available from the literature as described in the following. Various evaluation metrics are used to assess the performance of the Chemprop models on each dataset and against other models previously reported in the literature:

- ROC-AUC: area under the receiver operating characteristic curve
- PRC-AUC: area under the precision-recall curve
- AP: average precision
- MAE: mean absolute error
- RMSE: root-mean-square error
- R^2: coefficient of determination
- SID: spectral information divergence

1. MoleculeNet & OGB

The HIV and PCBA datasets from MoleculeNet[66] and Open Graph Benchmark (OGB)[67] were selected for classification tasks. Both MoleculeNet and OGB provide a diverse set of benchmark datasets that have been widely used to compare the performance of various machine learning models. They also host public leaderboards that allow us to directly compare our results to other public models. The HIV dataset contains the ability to inhibit HIV replication for 41,127 compounds. The PCBA dataset includes the 128 biological activities selected from PubChem BioAssay[89] for 437,929 compounds. The datasets were evaluated using the random and scaffold splits that were provided by MoleculeNet and OGB. We adopted the training, validation, and test sets of the scaffold-split HIV data and the random-split PCBA data from MoleculeNet. The scaffold-split PCBA data were adopted from OGB as MoleculeNet did not evaluate the

Table I. Summary of the benchmarking datasets.

Dataset/Category	Property/Data type	Task type	N tasks	N data	Metric(s)	Ref.[a]
MoleculeNet & OGB	HIV (HIV replication inhibition)	Classification	1	41,127	ROC-AUC	66
	PCBA (biological activities)	Classification	128	437,929	PRC-AUC, AP	66,67
	QM9 (DFT calculated properties)	Regression	12	133,885	MAE, RMSE	66,68
SAMPL	logP	Regression	1	23,469[b]	RMSE	69,70
Atom/bond-level targets	Quantum mechanical descriptors	Regression	6	136,219	MAE, RMSE	52
	Bond dissociation enthalpy	Regression	1	42,577	MAE	71,72
	Partial charge	Regression	1	130,267	MAE, RMSE	73
Reaction barrier heights	E2	Regression	1	1264	MAE	59,74–76
	S_N2	Regression	1	2361	MAE	59,74–76
	Cycloaddition	Regression	1	5269	MAE	77
	RDB7	Regression	1	23,852[c]	MAE	78
	RGD1-CNHO	Regression	1	353,984[c]	MAE	79
UV/Vis Absorption	UV/Vis peak absorption wavelength	Regression	1	26,395	MAE, RMSE, R^2	80–83
IR Spectra	Spectra between 400 and 4000 cm^{-1}	Spectra	1	8,754	SID	84
PCQM4MV2	HOMO-LUMO gap	Regression	1	3,452,151	MAE, RMSE	85

[a] References for the data and data splits.
[b] The size of the training set. The SAMPL6, SAMPL7, and SAMPL9 data are used as a test set.
[c] Including reverse reactions.

PCBA model on the scaffold split. In all splits, the datasets were split into 80% training, 10% validation, and 10% test sets.

QM9 is a dataset of DFT calculation values commonly used for chemical model benchmarking. The calculations for this dataset were originally carried out by Ramakrishnan et al.[68] and later distributed as part of the MoleculeNet benchmarks.[66] The dataset is made up of 133,885 molecules with properties and structures calculated at the B3LYP/6-31G(2df,p) level of theory. The molecules were chosen as the set of possible molecules containing up to nine heavy atoms of the types C, N, O, and F. Data sources for QM9 provide 3D coordinates for the atoms in the optimized structures, but we only use molecule SMILES as inputs for model training in this work. QM9 provides 12 target values for each molecule, provided in Table II. In the MoleculeNet presentation of the properties, atomized versions of the thermochemical properties U0, U298, H298, and G298 are provided alongside the original versions of the properties. In this work, we will use the atomized thermochemical properties. This dataset was randomly divided into 80% training, 10% validation, and 10% test data.

Table II. Target values present in the QM9 dataset, presented with the target labels used in the dataset.

Label	Units	Description
mu	D	Dipole moment
alpha	α_0^3	Polarizability
HOMO	Ha	Highest occupied molecular orbital energy
LUMO	Ha	Lowest unoccupied molecular orbital energy
gap	Ha	Homo-Lumo gap
r2	α_0^2	Electronic spacial extent
ZPVE	Ha	Zero point vibrational energy
U0	Ha	Internal energy at 0 K
U298	Ha	Internal energy at 298.15 K
H298	Ha	Enthalpy at 298.15 K
G298	Ha	Free energy at 298.15 K
Cv	$\frac{\text{cal}}{\text{mol K}}$	Heat capacity at 298.15 K

2. SAMPL

Experimental logP data of the SAMPL6, SAMPL7 and SAMPL9 challenges was downloaded from the SAMPL GitHub repository.[69] SAMPL runs a series of blind challenges for computational chemistry, providing the identity of test molecules for which predictions of physicochemical properties, among them the water-octanol partition coefficients, can be submitted using quantum-mechanics, molecular mechanics, or empirical models. In this work, we build an empirical model based on Chemprop, which we train on a publicly available dataset of logP measurements from Ref. 70. Molecules present in the SAMPL challenges were removed from the logP training dataset. The remaining 23,469 data points were randomly split into 80% training, 10% validation, and 10% test data. The test data was used to obtain a measure of performance for the literature dataset. We then retrained a production model on the full dataset (no validation or test data) using the best hyperparameters and number of epochs identified earlier, with which we made predictions for the three SAMPL challenges.

3. Atom/bond-level targets

To predict atom-level and bond-level targets, we selected three benchmark datasets. The framework we used to predict atomic and bond properties in Chemprop was based on modifications made to the approach developed by Guan et al.[52] They published a dataset of quantum mechanical descriptors for 136,219 organic molecules with atom types H, C, O, N, F, S, Cl, Br, B, I, P, and Si. This dataset included atomic charges, Fukui indices, NMR shielding constants, bond length, and bond orders. Those molecules were optimized using GFN2-xTB and subjected to population analysis using the B3LYP/def2-SVP level of theory. We used this dataset to evaluate the performance of different implementations, randomly splitting it into 80% training, 10% validation, and 10% test data.

For benchmarking, we also used the BDE-db dataset from St. John et al.[71] This dataset contains bond dissociation enthalpies (BDEs) for 42,577 closed-shell organic molecules with up to 9 heavy atoms of types C, H, O, and N, resulting in 290,664 BDEs. BDEs were calculated using the M06-2X/def2-TZVP level of theory. We used the same data splits as their study,[72] with 40,577 data points as training set and 1000 molecules each in the validation and test sets.

Lastly, we included a dataset of DDEC partial charges, which includes partial charges calculated with different dielectric constants ($\epsilon = 4$ for charges in protein and $\epsilon = 78$ for charges in water).[73] The dataset comprises 130,267 moderate size organic molecules with elements of types C, H, N, O, S, P, F, Cl, Br, and I, curated from ZINC and ChEMBL databases. We randomly split this dataset into 80% training, 10% validation, and 10% test data. Two external test sets of 146 organic liquids and 1081 FDA-approved drugs were used to test the transferability of the models.

4. Reaction barrier heights

To benchmark Chemprop's reaction functionality, four datasets of computational barrier heights were selected to cover a broad range of dataset size, diversity and quality. Since some of the original publications only report model mean absolute errors, we also report mean absolute errors, although we train on mean squared errors similar to all other benchmarks in this study.

First, E2 and S_N2 reactions originally published in Ref. 74 were used with reaction SMILES as provided in Ref. 59. We used the same split sizes as Ref. 75 and Ref. 76, which are 1440 training and 360 validation data points for S_N2, as well as 800 training and 200 validation data points for E2 with random splits. All other data points were used for testing. We then compared the performance of Chemprop to those reported in Ref. 75 and Ref. 76.

Second, cycloaddition reactions from Ref. 77 were used, with random splits into 80% training, 10% validation and 10% test data. We then compared Chemprop against all models reported in Ref. 90.

Third, the RDB7 dataset, which contains 11,926 high-accuracy reaction barrier heights and enthaplies calculated at CCSD(T)-F12/cc-pVDZ-F12 as provided in Ref. 78. In contrast to the E2, S_N2, and cycloaddition datasets that focus on one specific reaction class, this dataset spans a large range of barrier heights and is used to assess Chemprop's performance on substantially more reaction diversity. We randomly split the data into 80% training, 10% validation and 10% test data and then added reverse reactions to each set.

Fourth, the RGD1-CNHO dataset[79] was used, which comprises the largest and most diverse dataset out of the four, and also the most difficult to learn. We again randomly split the data into 80% training, 10% validation and 10% test data and then added reverse reactions to each set.

5. UV/Vis absorption

Multi-molecule models are demonstrated using prediction of UV/Vis peak absorption wavelength, a prediction model that involves both the absorbing molecule and the solvent. Our dataset of the peak wavelength of maximum absorption ($\lambda_{max,abs}$) is a combination of several databases[80–83] that were extracted from the experimental literature. There are 26,395 samples across a variety of dye molecule families and solvents. Each sample consists of a dye molecule SMILES, solvent molecule SMILES, and a peak wavelength value. There are no multi-component species of either dyes or solvents. The train-validation-test splits are in 80/10/10 proportions and are constrained to avoid data leakage of highly-correlated measurements of the same dye in multiple solvents.

6. IR spectra

The dataset used for whole-spectra predictions was collected from infrared absorption spectra made public by NIST.[84] This dataset comprises 8,754 gas-phase spectra, with absorbance magnitudes indicated at 2 cm^{-1} intervals between 400 and 4000 cm^{-1}. The spectra for different molecules have different ranges of collected absorbance and may have regions of missing or excluded values. We randomly split this dataset into 80% training, 10% validation, and 10% test data.

7. HOMO-LUMO gaps

The PCQM4MV2 dataset is a collection of DFT-calculated molecular HOMO-LUMO gaps, originally col-

lected as part of the PubChemQC project[85] and now curated as part of the Open Graph Benchmark.[67] This dataset contains HOMO-LUMO gaps measured in units of eV for 3,452,151 molecules. A further 294,470 molecules have targets privately held by the Open Graph Benchmark for blinded testing purposes and are not included in the benchmarks performed in this work. For benchmark training, the data we had available was randomly divided into 80% training, 10% validation, and 10% test data. The Open Graph Benchmark provides 3D coordinates for training data used in the dataset, but we only use molecule SMILES as inputs for model training in this work.

III. DISCUSSION OF FEATURES

1. Ease of use

Chemprop comes with quick-start guidelines on GitHub[87], an extensive documentation including tutorials,[88] and a video workshop,[97] among other resources. The GitHub repository is actively maintained, and user requests through GitHub Issues are welcomed. Models can be trained and tested with a single line on the command line (or a few lines of python code) and a user-supplied CSV file. This makes the package very easy to use, and Chemprop models are used frequently in literature. Table III lists a non-exhaustive selection of studies based on Chemprop, showcasing its versatility and applicability for the prediction of a large variety of chemical properties.

2. GPU support

The PyTorch backend of Chemprop enables seamless GPU acceleration of both model training and inference. The acceleration of training and inference processes when used with a GPU can be significant, as shown later in Section IV C. This feature is enabled by default on machines possessing a CUDA-enabled GPU, but it may be turned off via the `--no-cuda` argument on the command line. For machines with multiple GPUs, Chemprop will use the default GPU as PyTorch (typically the GPU at index 0). It is possible to specify the k^{th} GPU by either passing `--gpu <k>` on the command or setting the `CUDA_VISIBLE_DEVICES` environment to show only device `k` (i.e., executing `CUDA_VISIBLE_DEVICES=k` prior to running Chemprop).

3. Regularization

Chemprop has two builtin forms of regularization, intended to help reduce overfitting in trained models. These two regularization techniques were present in the initial release of Chemprop and remain an important contributor to model quality. The first form of regularization is called early stopping. With early stopping, the performance of the model on the validation set is calculated at the end of each epoch. The version of the model that is stored at the end of training is the one saved at the end of the best scoring epoch. This has the effect of discarding later epochs of training where the model would be overfitting to the training data, continuing to improve the training loss at the cost of hurting performance on the validation and test sets. Contrary to what the name implies, early stopping as implemented in Chemprop does not shorten the amount of time needed for training.

The second form of regularization is called dropout. During training, dropout regularization will randomly zero out a fraction of the latent variables for that forward pass. This practice has been shown to reduce overfitting and lead to higher quality latent variables.[98] The level of dropout regularization can be specified using the option `--dropout <p>` where p is the dropout probability. By default, dropout is inactive. We have observed dropout to be a helpful addition to models in a variety of contexts and recommend that users include it in their choices of hyperparameters.

4. Additional features

Chemprop can take additional features at the molecule-level or at the atom- and bond-level as optional inputs. While Chemprop often generates very good models without requiring any input beyond the SMILES, it has been shown that outside information added as additional features can further improve performance[36,53]. Users can provide their custom additional features by adding keywords and paths to the data files containing the features.

For molecule-level features x_m, a path to the features can be specified using the keyword `--features_path PATH/TO/FEATURES`. The provided molecular features are concatenated to the learned molecular embedding prior to the FFN network. The features can be provided as a numpy `.npy` file or CSV file. For both file formats, the features must be in the same order as the SMILES strings in the data file. The features file should not contain the SMILES strings as features will be associated with the corresponding molecule based on the ordering in the file. The features file should contain numerical values, with columns corresponding to different features and rows corresponding to molecule data points. By default, provided features are normalized unless the flag `--no_features_scaling` is used.

For additional atomic features x_v, the path to the features can be provided using the keyword `--atom_descriptors_path PATH/TO/FEATURES`. The supported file formats include `.npz`, `.pkl`, and `.sdf`. Two options are available to select in

Table III. Selected published studies based on Chemprop.

Ref.	Year	Prediction
10	2020	Growth inhibitory activity against *E. coli*; led to an identification of a potential new drug
50	2021	Chemical synergy against SARS-CoV-2; identified two drug combinations with strong antiviral synergy in vitro
28	2021	IR spectra of molecules
52	2021	Atomic charges, Fukui indices, NMR constants, bond lengths, and bond orders
59	2022	Reaction rates and barrier heights
60	2022	Barrier heights of reactions
29	2022	Molecular optical peaks
53	2022	Solvation free energy, solvation enthalpy, and Abraham solute descriptors
91	2022	Activity coefficients
54	2022	Solid solubility of organic solutes in water and organic solvents
51	2022	Absorption, distribution, metabolism, and excretion (ADME) properties for drug discovery
55	2023	Lipophilicity
56	2023	Solvent effects on reaction rate constants
57	2023	Critical properties, acentric factor, and phase change properties
58	2023	Fuel properties
92	2023	Molecular optical peaks
70	2023	Molecular optical peaks and partition coefficients for closed-loop active learning
93	2023	Growth inhibitory activity against *A. baumannii*; led to an identification of a potential new drug
94	2023	Differences in pharmacokinetic properties of pairs of molecules
95	2023	Toxicity measurements consisting of 12 nuclear receptor signaling and stress response pathways
96	2023	Senolytic activity of compounds to selectively target senescent cells

which way atom descriptors are used. The option `--atom_descriptors descriptor` concatenates the additional features to the embedded atomic features after the D-MPNN. On the other hand, the option `--atom_descriptors feature` concatenates the features to the initial atomic feature vectors prior to the D-MPNN, such that they can be used during message-passing. Additional bond-level features can be provided via `--bond_descriptors_path PATH/TO/FEATURES` in the same format as the atom-level features. Similarly, users must choose in which way bond descriptors are used. The option `--bond_descriptors descriptor` concatenates the new bond-level features to the embedded bond features after the D-MPNN, which can only be used for bond-level property prediction, while the option `--bond_descriptors feature` concatenates the new features with the default bond feature vectors before the D-MPNN. Further details on how to input the atom and bond features for training and inference tasks, with examples, are given in the *SI*.

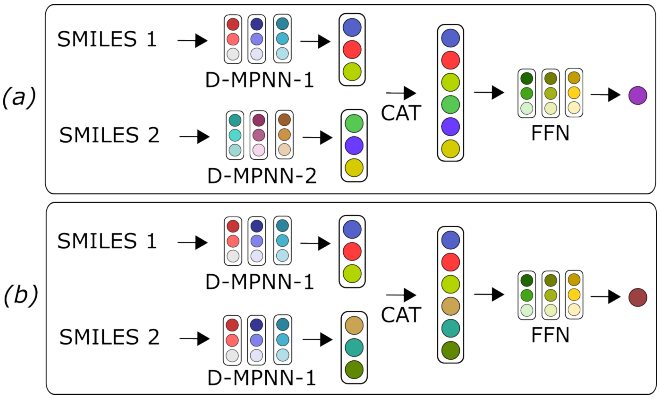

Figure 2. Example of how two molecules ($N = 2$) can be embedded in Chemprop. a) A separate D-MPNN is used for each molecule or b) the same D-MPNN is used. After embedding, the different molecular vectors are concatenated (CAT) and used as input to the feed forward network (FFN) for property prediction.

5. Multi-molecule models

Some properties depend on the structure of multiple molecules. For example, when properties related to solvation need to be predicted both a solute and a solvent are required as input to the model. Users can provide multiple molecules as input to Chemprop. The number of molecules N is specified with the keyword `--number_of_molecules`. For the example of a solute-solvent pair $N = 2$. To train a new model using multiple molecules as an input, the SMILES string of each molecule must be provided as a separate column in the input CSV file. If N molecules are used, Chemprop assumes that the SMILES strings are located in the first N columns by default. Alternatively, the names of the specific columns containing the SMILES of the different molecules can be specified using the `--smiles_columns <column_1> ...` option.

The embedding of multiple molecules in Chemprop can be done in two different ways as schematically represented in Figure 2 for $N = 2$. When multiple molecules are used, by default Chemprop trains a separate D-MPNN for each molecule (Figure 2a). If the option

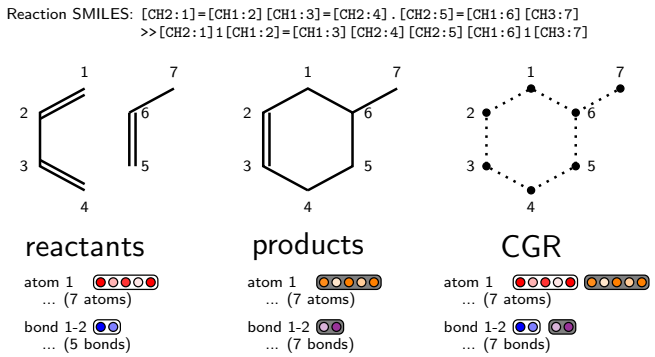

Figure 3. Construction of the condensed graph of reaction (CGR) of an example reaction. The vertices and edges are obtained as the union of the respective reactant and product vertices and edges. The features are obtained as a combination of the reactant (white background) and product (gray background) features for atoms and bonds.

`--mpn_shared` is specified, the same D-MPNN is used for all molecules (Figure 2b). In both cases, the D-MPNN of each molecule uses the same hyperparameters. The embeddings of the different molecules are then concatenated prior to the FFN. Note that the current implementation of multiple molecules in Chemprop does not ensure permutational invariance towards the input molecules. This is suited to situations where the input molecules have different roles, e.g. molecule 1 = solute, molecule 2 = solvent.

6. Reaction support

Chemprop supports the input of atom-mapped reactions, i.e. pairs of reactants and products SMILES connected via the "`>>`" symbol by using the keyword `--reaction`. The pair of reactants and products is transformed into a single pseudomolecule, namely the condensed graph of reaction (CGR), and then passed to a regular D-MPNN block. The construction of a CGR within Chemprop is described in detail in Ref. 59, and summarized in the following. In general, the input of a reaction vs. a molecule only affects the setup of the graph object and its initial features, but not any other part of the architecture like the D-MPNN, aggregation or FFN. The graph of a reaction has a different set of edges E (but the same set of vertices V, at least for balanced reactions) as the graph of a molecule, as shown in Fig. 3 for an arbitrary Diels-Alder reaction. To build the graph of the CGR pseudomolecule, the set of atoms is obtained as the union of the sets of atoms in the reactants and products. Similarly, the set of bonds is obtained as the union of the sets of bonds in the reactants and products. The CGR thus has at least as many atoms and bonds as the reactant and as the product, usually more. The initial atom and bond feature vectors contain information on both the reactant (white background) and product (gray background) features. Whenever information is not available, e.g. because a bond did not exist in either the reactants or products, the features are set to zero. A simple concatenation of reactant and product features can be used to obtain the pseudomolecule features (keyword `--reaction_mode reac_prod`). Since the atomic number does not change upon reaction, its one-hot encoding is not repeated in the second part of the feature vector. For many reaction properties the change in the local structure upon reaction, i.e. the difference between reactants and products, is very informative. Since neural networks are known to not perform well for adding and subtraction operations, we also provide options to include the difference in properties directly. Namely, one can concatenate the difference in atom and bond features with the reactant properties (keyword `--reaction_mode reac_diff`, default) or with the product properties (keyword `--reaction_mode prod_diff`). Further, we provide options to automatically balance imbalanced reactions (via setting `--reaction_mode` to `reac_prod_balance`, `reac_diff_balance` or `prod_diff_balance`), which is useful if a dataset contains both balanced and imbalanced reactions. For datasets containing only balanced reactions, or only imbalanced reactions, e.g. where leaving groups are never specified, this is usually not necessary.

Optionally, Chemprop accepts an additional molecule object as input, such as a solvent, or a reagent, etc. which is passed to its own D-MPNN similar to the multi-molecule model. The output of the reaction D-MPNN and molecule D-MPNN is concatenated after atomic aggregation, before the FFN. This option is available via the `reaction_solvent` keyword. The size of the molecule D-MPNN can be varied with the keywords `hidden_size_solvent` and `depth_solvent`. We note that this additional solvent or reagent needs to be passed as a separate SMILES column in the input CSV file. Reagents passed within the reaction SMILES string (between the `>` symbols) are disregarded. An example of the reaction-solvent model can be found in Ref. 56, in which a Chemprop model is trained to predict kinetic solvent effects for a diverse range of reactions and solvents.

7. Spectra data support

Chemprop supports prediction of whole-spectrum properties for molecules. An initial version of this capability was discussed in Ref. 28 for use with IR absorbance spectra. Targets for the spectra dataset type are composed of an array of intensity values, set at fixed bin locations typically specified in terms of wavelengths, frequencies, or wavenumbers. Target intensity values and resulting predictions are required to be positive, as in the case of ideal absorbance spectra. Spectral Information Divergence was originally developed as a method of comparing spectra to reference databases[99] and is adapted in Chemprop to be used as a loss function, considering the

deviation of the spectrum as a whole rather than independently at each bin location. Spectra predictions are normalized to sum all bin intensities to 1.

The treatment of spectra targets can handle spectra with gaps or missing values within a dataset. With the expectation that spectra will often be collected in contexts where a portion of the range will be obscured or invalid, Chemprop can be instructed to create exclusion regions in the spectra where no predictions are provided and targets are ignored for training purposes. An example of this would be heavy solvent absorbance in IR spectra, where samples collected in CCl_4 are not reliable in the range 1500–1600 cm^{-1}, while samples collected in other solvents may be valid in this range. The phase of data points used in training the model can be provided using `--phase_features_path PATH/TO/FEATURES` and an associated mask array used to exclude spectra regions for particular phases can be provided using `--spectra_phase_mask_path PATH/TO/MASK`.

8. Latent representations

Graph neural networks enable learning both molecular representation and property end-to-end directly from the molecular graph. As detailed above in Section II A, the learned node representations are aggregated into a molecule-level representation after the message-passing phase. In analogy to canonical methods of molecular representation, we refer to this molecule-level representation as the "learned fingerprint." This embedding is then further fed into an FFN readout network that connects this representation to the target properties in a differentiable manner. While each hidden layer of the FFN calculates its own hidden representation as well, we focus on the *final* hidden representation (i.e., the input prior to the output layer of the FFN), which we refer to as the "embedding" of an input. Both of these vectors may be referred to as "latent representations" of a molecule, as it relates to a particular trained model. Chemprop supports the calculation of either from a trained model for a given set of molecules using the `chemprop_fingerprint` command line entry point and the corresponding value of `--fingerprint_type`: `mpn` for learned fingerprints and `last_ffn` for molecular embeddings. If an ensemble of models is used to calculate a latent representation, the vector returned by Chemprop will be a concatenation of the latent representations of each model in the ensemble.

9. Loss function options

Chemprop can train models according to many common loss functions. The loss functions available for a given task are determined by the dataset type (regression, classification, multi-class, or spectra). Regression models can be trained with mean squared error (MSE), bounded MSE (for datasets that are a mix of equalities and inequalities), negative log-likelihood (NLL) based on an assumption of a Gaussian underlying distribution, or NLL based on an assumption of a normal-inverse gamma (NIG) underlying distribution. MSE is the default loss for regression tasks. If an NLL loss function is used during training, one can do uncertainty quantification at inference time via mean-variance estimation (MVE)[100] or evidential uncertainty[101]. Classification tasks default to the binary cross entropy loss and have additional options of Matthews correlation coefficient (MCC) and Dirichlet (evidential classification)[102]. The binary cross entropy and MCC have analogues available for multi-class problems. There are two options available for training on spectra: spectral information divergence (SID)[99] and first-order Wasserstein distance (a.k.a. earthmover's distance)[103]. Loss functions must be differentiable since they are used to calculate gradients that update the model parameters, but Chemprop also provides the option to use several non-differentiable metrics for early stopping with the validation set while training with a differentiable loss function.

10. Transfer learning

Transfer learning is a general strategy of using information gained through the training of one model to inform and improve the training of a a related model. Often, this strategy is used to transfer information from a previously trained model of a large dataset to a model of a small dataset in order to improve the performance of the model of the small dataset. The simplest method of transfer learning would be taking predictions or latent representations from one model and supplying them as additional features to another model (see Sections III 4 and III 8).

In Chemprop, different strategies are available to transfer learned model parameters from a previously trained model to a new model as a form of transfer learning. The model parameters from the pretrained model can be used to initialize the new model with the `--checkpoint_dir`, `--checkpoint_path`, or `--checkpoint_paths` options that refer to the location of previously trained model(s). If used only for model initialization, the transferred weights will be updated normally during training of the new model. Alternatively, the `--checkpoint_frzn` option can be used to set the transferred model parameters and hold them unchanged during backpropagation, *i.e.* freezing these parameters. By default, all the D-MPNN layers are frozen while leaving the FFN layers unfrozen. The first n layers of FFN can also be frozen by using the option `--frzn_ffn_layers n`. The options to initialize (a) or freeze the D-MPNN (b) and FFN (c) model parameters are visualized in Figure 4. The last example (c) shows how the layers of the D-MPNN are frozen, as well as the first two layers of the FFN. Two additional layers are added to the FFN which can be optimized during train-

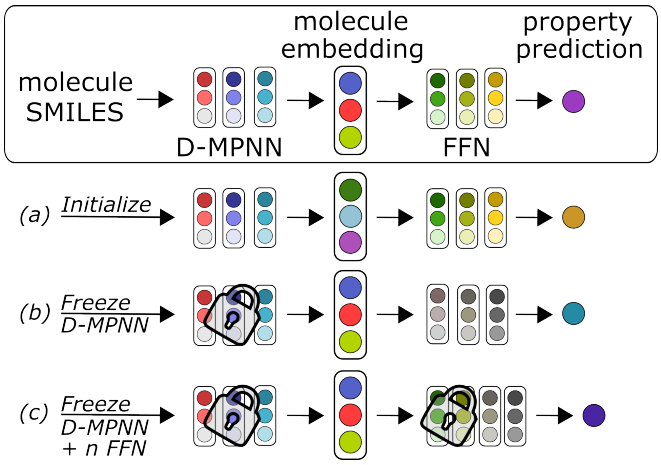

Figure 4. Options to transfer model parameters from a pretrained model (squared) to a new model, by (a) initializing the new model parameters or by freezing the (b) D-MPNN layers and (c) n FFN layers

ing of the new model.

11. Hyperparameter optimization

Hyperparameter optimization is a key step when building ML models that can lead to significant gains in model performance. Given the sheer size of the search space with deep learning models, this step can often be operationally challenging. Chemprop provides a command line utility, `chemprop_hyperopt`, that automates this process by removing the need to manually define the search space of hyperparameters. Users can simply supply a list of keywords from which to build a hyperparameter search space (Table IV). The number of trials of hyperparameter combinations to be tested can be set using the `--num_iters` argument. By default, the search space will first be randomly sampled for `num_iters`/2 trials before switching to targeted sampling via the tree-structured Parzen estimator algorithm[104,105] for the remaining trials. The number of random trials to be used can be changed by setting `--startup_random_iters` to a value less than `num_iters`.

Hyperparameter optimization can be the most resource-intensive step in model training. In order to search a large parameter space adequately, a large number of trials would be needed. Chemprop allows for parallel operation of multiple hyperparameter optimization instances, so that the entire set of trials does not need to be run in series. Parallel operation can be achieved by setting the location of trial checkpoint files with `--hyperopt_checkpoint_dir` to be a single shared location for multiple hyperparameter optimization instances. This allows for multiple instances of the program to share and contribute to the same trial history, reducing the wall time needed to perform hyperparameter optimization significantly.

12. Uncertainty tools

Chemprop includes a variety of popular uncertainty estimation, calibration, and evaluation tools. The estimation methods include ensembles[106], dropout[107], mean-variance estimation (MVE)[100], and evidential[101], as well as a special version of ensemble variance for spectral predictions[28], and the inherently probabilistic outputs of classification models. After estimating the uncertainty in a model's predictions, it is often helpful to calibrate these uncertainties to improve their performance on new predictions. We provide four such methods for regression tasks (z-scaling[108], t-scaling, Zelikman's CRUDE[109], and MVE weighting[110]) and two for classification (single-parameter Platt scaling[111] and isotonic regression[112]). In addition to standard metrics such as RMSE, MAE, etc. for evaluating predictions, Chemprop also includes several metrics specifically for evaluating the quality of uncertainty estimates. These include negative log likelihood, Spearman rank correlation, expected normalized calibration error (ENCE)[113], and miscalibration area[113]. Any valid classification or multiclass metric used to assess predictions can also be used to assess uncertainties.

13. Atom/bond-level targets

Chemprop supports a multitask constrained D-MPNN architecture for predicting atom- and bond-level properties by using the keyword `--is_atom_bond_targets`. This model enables a D-MPNN to be trained on multiple atomic and bond properties simultaneously. Specifically, the atomic and bond embeddings are linked with corresponding atomic and bond properties by FFNs. Additionally, an attention-based constraining method is used to correct the discrepancy between the sum of a predicted atomic property and its molecular net value, such as the partial charge of a molecule. An initial version of this capability was developed in Ref. 52. The initial code implementation only supported neutral species, and constraints could only be applied to atomic properties, not bond properties. Moreover, the total loss across different targets was not automatically scaled; instead, the weights of each target needed to be specified. To overcome these limitations, we modified the algorithm and implemented a more flexible version in Chemprop. For details on the input formats to be used for atom/bond targets, both for training and for inference, see the *SI*.

IV. RESULTS

A. Model Performance: General benchmarking

In the following, we present benchmarking results on predicting molecular targets on single-molecule datasets.

Table IV. Searchable hyperparameters using `chemprop_hyperopt`.

Keyword	Description
`activation`	the activation function used after each linear layer, when necessary
`aggregation`	the aggregation function used when constructing a molecule-level representation from node-level representations
`aggregation_norm`	the normalization factor if using `norm` aggregation
`batch_size`	the minibatch size
`depth`	the number of message-passing iterations
`dropout`	the dropout probability after each layer in both the D-MPNN encoder and FFN
`ffn_hidden_size`	the size of each hidden layer in the FFN
`ffn_num_layers`	the number of layers in the FFN
`hidden_size`	the message size in the D-MPNN encoder
`linked_hidden_size`	the size of *both* the messages in the D-MPNN encoder and the hidden layers in the FFN. This argument is overriden by either `hidden_size` or `ffn_hidden_size`
`max_lr`	the maximum learning rate used in the learning rate scheduler
`init_lr`	the initial learning rate expressed as the ratio of `init_lr` to `max_lr`
`final_lr`	the final learning rate expressed as the ratio of `final_lr` to `max_lr`
`warmup_epochs`	the number of epochs over which to ramp up the learning rate up from `init_lr` to `max_lr` expressed as a fraction of the total training epochs
`basic`	search over `depth`, `ffn_num_layers`, `dropout`, and `linked_hidden_layers`
`learning_rate`	search over `init_lr`, `max_lr`, `final_lr`, and `warmup_epochs`
`all`	all of the above hyperparameters

Table V. Test set metrics for the different targets of QM9. The top grouping of tasks was trained together in a single multitask model. The bottom grouping of results for U0 and gap shows the results for single-task models. The atomized basis of the thermochemical properties U0, U298, H298, and G298 were used for training.

Model	Target	MAE	RMSE
multitask	mu	0.326	0.582
	alpha	0.232	0.516
	HOMO	0.002 39	0.004 19
	LUMO	0.002 41	0.004 12
	gap	0.003 27	0.005 86
	r2	17.4	33.2
	ZPVE	0.000 258	0.000 362
	Cv	0.112	0.222
	U0	1.90	3.18
	U298	1.92	3.19
	H298	1.92	3.19
	G298	1.88	3.16
individual	U0	1.08	2.44
	gap	0.003 15	0.005 88

1. MoleculeNet & OGB

In the original publication of the algorithm behind Chemprop,[36] the MoleculeNet datasets were used as benchmark to compare against other non-deep-learning algorithms, such as Morgan Fingerprints used with random forest regression.[36] In this work, we do not fully repeat the original coverage of the MoleculeNet datasets. We revisit three of the datasets that continue to be of interest: QM9, HIV, and PCBA. The most significant differences in the benchmark models presented here and those presented in Ref. 36 are the improved hyperparameter search and the adoption of the "norm" aggregation setting. For PCBA, we additionally test the performance of the model on the scaffold split that is obtained from OGB.[67]

First, we trained a multitask model on all 12 targets in the QM9 dataset, which produced an average MAE of 2.14 and RMSE of 3.94 across all targets. Though reporting averaged metrics is common, the differing orders of magnitude among the target properties biases the averaged result heavily toward targets of larger magnitudes. In Table V, we report the test set metrics individually by task. We also trained benchmark single-task models on the U0 and HOMO-LUMO gap targets, reported in Table V. In this benchmark, the performance observed on the single-task treatment of U0 is significantly better than the multitask version with RMSE of 2.44 and 3.18 Ha, respectively. The single-task model did not have a clear improvement on HOMO-LUMO gap performance.

The results of the benchmark Chemprop models trained on the HIV and PCBA datasets are presented in VI. For the PCBA dataset which has 128 classification tasks, the test scores are averaged over all tasks. The Chemprop model achieves a ROC-AUC of 0.8084 on the scaffold split for the HIV prediction. While our model underperforms compared to the best models from MoleculeNet and OGB leaderboards, our model provides better predictions than the average models from both leaderboards on the HIV scaffold split. For the PCBA random split, the Chemprop model has a PRC-AUC of 0.2178, outperforming the best model from MoleculeNet, the DeepChem graph convolutional model[114] with a PRC-AUC of 0.136. Compared to the best model from OGB,

Table VI. Test set results for HIV and PCBA classification tasks compared with MoleculeNet (MolNet) and OGB leaderboards (higher = better). For the PCBA scaffold split, the test set only had a single class for 'PCBA-493208' task, and therefore 'PCBA-493208' was omitted from the test scores.

Dataset	Metric	Split type	MolNet best	MolNet average	OGB best	OGB average	This work
HIV	ROC-AUC	Scaffold	0.841	0.788	0.8420	0.7936	0.8084
PCBA	PRC-AUC	Random	0.136	0.119	-	-	0.2178
PCBA	AP	Random	-	-	-	-	0.2236
PCBA	PRC-AUC	Scaffold	-	-	-	-	0.2988
PCBA	AP	Scaffold	-	-	0.3167	0.2716	0.3028

Table VII. RMSEs for predicting logP (dimensionless) for the SAMPL6, SAMPL7, and SAMPL9 challenges.

	SAMPL6	SAMPL7	SAMPL9
Number of submissions	105	36	18
Average RMSE of submissions	1.51	1.44	2.11
Best RMSE	0.38	0.49	1.12
RMSE of this work	0.33	0.46	1.05

which uses the heterogeneous interpolation on graph[115], our model has a lower AP of 0.3028 on the PCBA scaffold split, but we are able to achieve better performance than the average models.

2. PCQM4Mv2

A benchmark model was trained on the PCQM4Mv2 dataset curated by the Open Graph Benchmark.[67] This dataset contains the molecular HOMO-LUMO gap calculated by DFT in units of eV. The benchmark Chemprop model achieved a test set MAE of 0.0951 eV and RMSE of 0.154 eV. The OGB hosts a leaderboard for performance for this dataset, based on a blinded test set. The test set used for this model was part of the open data and is expected to be similar but not the same as the test set reported on the leaderboard.

3. SAMPL

When training Chemprop to predict water-octanol partition coefficients (logP), we obtain an RMSE of 0.51 on a random test set with our conventional data splits of 80% test, 10% validation and 10% test. This corresponds to an RMSE of 0.70 kcal mol^{-1} for the transfer free energy ΔG from water to octanol at 298 K, which is related to logP by

$$\Delta G = -RT \ln 10 \cdot \log P \tag{8}$$

where R is 8.314 J mol^{-1} K^{-1} and T is the temperature. We then retrained a production model on the full logP dataset without any validation or test splits using the same hyperparameters to predict logP of the molecules in the SAMPL6, SAMPL7, and SAMPL9 blind prediction challenges. The performance of Chemprop is shown in Table VII, where Chemprop outperforms all other submissions from the SAMPL6, SAMPL7 and SAMPL9 challenges. We note that submissions range from quantum mechanics (QM) models and molecular mechanics (MM) models to empirical models relying on heuristic rules or machine learning, as well as mixtures thereof. Our work therefore does not only outperform other empirical models, but also a large variety of QM and MM models. In general, logP is often used in drug development, where it serves as an indicator of lipophilicity, which is known to impact the absorption, distribution, metabolism, excretion, and toxicity of drug candidates.[116] We thus demonstrate the ability of Chemprop to aid in important tasks such as drug discovery. Moreover, we note that the best performing submission in SAMPL7 was made by a biotechnology company independent of our group, using a Chemprop model trained on a different database, further highlighting the usefulness and impact of our software.

B. Model Performance: Specific feature demonstrations

In the following, we present benchmarking results for speciality features of Chemprop, namely the training on reactions or multiple molecules, the prediction of atom/bond-level targets or spectra, and the use of uncertainty quantification methods.

1. Atom/bond-level targets

As shown in Fig. 5, the performance of a multitask constrained D-MPNN was evaluated on a dataset containing six atomic and bond quantum mechanical (QM) descriptors, with testing errors agreeing well with previous findings.[52]

Bond dissociation energy (BDE) prediction was also examined using a single-task model for BDE and a multitask model for both BDE and partial charge. The single-task model achieved an MAE of 0.60 kcal mol^{-1}, which is comparable to the testing error of 0.58 reported by the GNN model in ALFABET.[72] However, the GNN model in ALFABET was exclusively engineered for the purpose of BDE prediction, whereas the multitask model

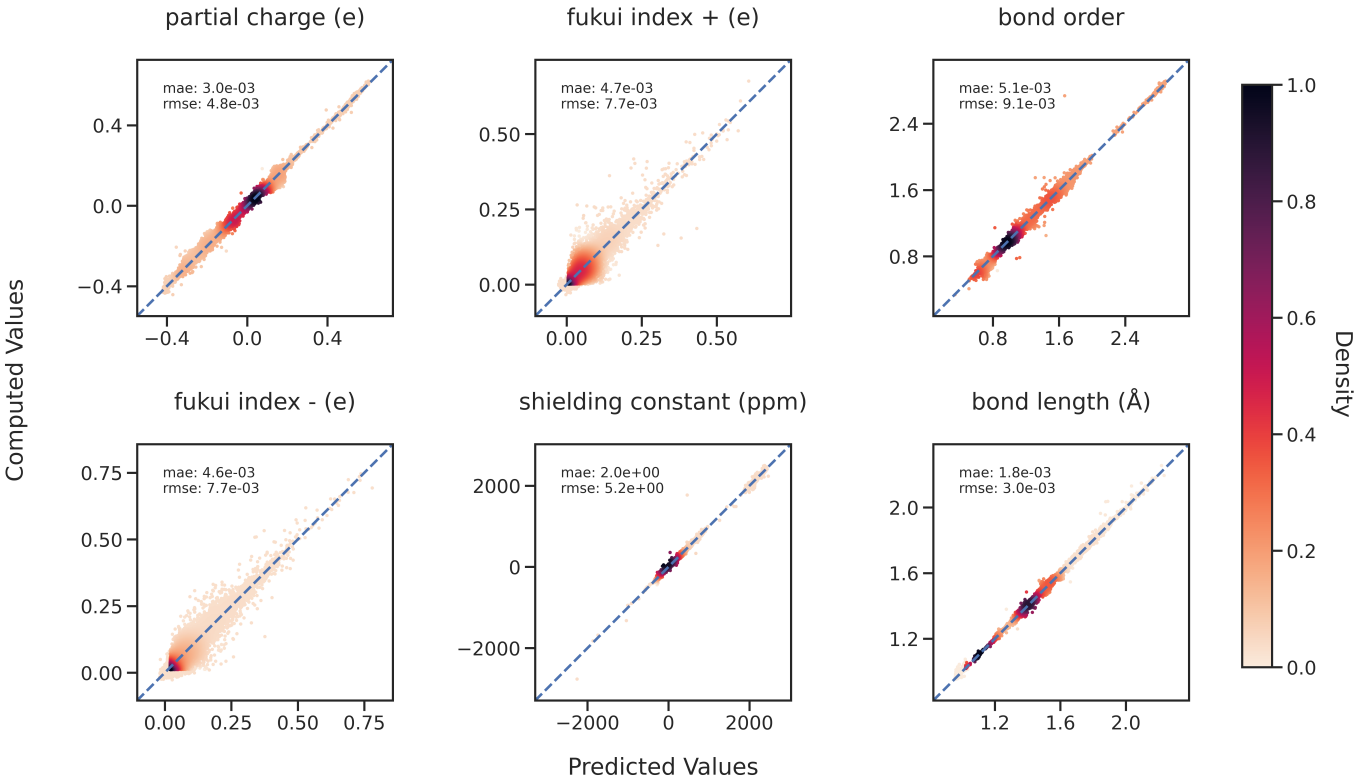

Figure 5. Comparing QM computed descriptors with multitask constrained model predictions on a held-out testing set.

in Chemprop is capable of training on diverse atom- and bond-level properties concurrently. The multitask model, trained on both atom-level (i.e., partial charges) and bond-level (i.e., BDE) descriptors, achieved better performance with a testing MAE of 0.56 kcal mol^{-1} on BDE predictions, outperforming both the GNN model and the single-task model. This was due to the synergistic effect of co-training.

The Chemprop prediction of DDEC partial charge on small organic molecules gave a low MAE (RMSE) of 0.0053 (0.0080) e and 0.0047 (0.0097) e for dielectric constants of $\epsilon = 4$ (in protein) and $\epsilon = 78$ (in solvent), respectively. Furthermore, the models were tested on external test sets to assess their transferability to real-world applications. The MAE (RMSE) for external test set 1 (organic liquids) and external test set 2 (FDA-approved drugs) was 0.008 (0.013) e and 0.006 (0.012) e, respectively. The results indicated that Chemprop significantly outperformed random forest regression model, with an RMSE of 0.023 e and 0.018 e for the two external test sets, as reported by Bleiziffer et al.[73]

These findings suggest that Chemprop is promising for predicting various atom- and bond-level properties of molecules, with potential applications in drug discovery and materials science.

Table VIII. MAEs for predicting the barrier heights of organic reactions in kcal mol^{-1} for this work (top), other graph-convolutional approaches (middle, taken from Ref. 75 and 90), and simple machine learning approaches (bottom, taken from Ref. 75, 76 and 90).

	S_N2	E2	Cycloadd	RDB7	RGD1
This work	2.57	2.53	2.57	4.04	5.85
WL GNN[75]	8.49	8.04	2.76	-	-
WL ml-QM-GNN[75,90]	2.76	2.65	2.64	-	-
Chemprop reactant[75]	2.85	2.75	-	-	-
Regression QM desc.[75,90]	6.43	6.07	5.94	-	-
KRR + BoB[76]	3.49	3.27	-	-	-
KRR + SLATM[76]	2.92	2.92	-	-	-
KRR + FCHL19[76]	2.87	2.75	-	-	-
KRR + one-hot[76]	2.14	2.40	-	-	-
KNN + Morgan[90]	-	-	4.33	-	-
RF + QM desc.[90]	-	-	3.73	-	-
RF + Morgan[90]	-	-	3.50	-	-
XGBoost + QM desc.[90]	-	-	3.78	-	-
XGBoost + Morgan[90]	-	-	3.84	-	-

2. Reaction barrier heights

Table VIII summarizes the MAEs obtained for different reaction barrier height datasets.

For E2 and S_N2 reactions we can directly compare or work against the models by Stuyver et al.[75] and Heinen et al.[76] We find that Chemprop significantly outperforms the Weisfeiler-Lehman (WL) architecture from Stuyver et al.[75,90] even when adding quantum-mechanical (QM) descriptors (termed "ml-QM-GNN" in Table VIII) to the WL network. We furthermore note that Ref. 75 also reports the performance of a Chemprop model, but trained it only on the reactants, not the full reactions. Here, we can directly observe the advantage offered by using the full reaction to construct the input graph representations. Chemprop furthermore outperforms the multivariate regression of quantum-mechanical descriptors of Ref. 75. Compared to the kernel ridge regression (KRR) models of Ref. 76, Chemprop outperforms the models based on BoB, SLATM, and FCHL19 representations by a large margin. The KRR model on a simple one-hot encoding of the nucleophile, electrophile and substituents close to the reactive center offers a slight performance benefit at the disadvantage of not being able to generalize at all to new reactants or reactions.

For [3 + 2] dipolar cycloadditions, we compare Chemprop in reaction mode to WL type models with and without QM features, regressions on QM descriptors, as well as different machine learning model (k-nearest neighbor (KNN), random forests (RF) and XGBoost) on either QM descriptors or Morgan fingerprints, and find that Chemprop outperforms all other models.

A large benefit of Chemprop in reaction mode over all other architectures in Table VIII is furthermore its generality and versatility. It is straightforward to train any machine learning model on a single type of reactions (like S_N2, E2, or cycloadditions), but finding a representation and architecture that can predict reaction properties of a large variety of reactions is much more difficult. Here, we showcase the ability of Chemprop to learn from diverse reaction datasets using the RDB7[78] and the RGD1-CNHO[79] datasets. We find larger MAEs compared to the simpler single-type datasets. Albeit not reaching chemical accuracy, our models still produce state-of-the-art performances given the diversity of reactions and range of barrier heights in both datasets. In Ref. 60, a refined Chemprop model with customized atom features and pretrained on DFT data of lower level of theory yields an MAE of 2.6 kcal mol^{-1}. Importantly, Chemprop does not make use of the three-dimensional structure of the reactants, products and transition states but estimates barrier heights solely from the change in bonds, thus requiring minimal information to predict a new reaction. We furthermore note that simpler approaches such as models trained only on reactant structures or descriptors are not applicable to diverse reaction datasets.

3. UV/Vis absorption

Chemprop achieved an MAE of 15.6 nm, RMSE of 30.0 nm, and R^2 of 0.919 on our dataset of experimental absorption peak wavelengths across a diverse set of dye molecules in a variety of solvents. This was previously demonstrated to outperform state-of-the-art fingerprint-based methods[29]. Our train-validation-test splitting for this task was constrained such that all measurements of the same molecule in different solvents would be assigned to the same split to avoid data leakage. Previous work has shown that data leakage can lead to overly-optimistic estimates of generalization ability on similar datasets[29]. Any work on multi-molecule tasks should carefully consider the implications of the choice of splitting technique to avoid data leakage from highly correlated samples (in cases such as measurements of the same property in different solvents) or from duplicated samples with flipped molecule columns (in cases of symmetric multi-molecule properties).

Table IX. Uncertainty evaluation metrics (dimensionless) for QM9 gap predictions. *NLL*: negative log likelihood; ρ: Spearman rank correlation; *ENCE*: expected normalized calibration error; *MA*: miscalibration area. The arrows indicate if smaller or larger values indicate better performance.

Method	NLL (↓)	ρ (↑)	ENCE (↓)	MA (↓)
Ensemble	-3.88	0.34	0.27	0.15
Evidential	-4.09	0.46	0.21	0.14
MVE	-4.13	0.48	0.15	0.09

4. IR spectra

Chemprop spectra prediction was benchmarked using gas-phase IR absorbance data provided publicly by NIST.[84] Similarity between the predicted spectra and the target spectra are assessed using Spectral Information Divergence, SID. The benchmark average SID for predictions on the test set was 0.32. Qualitatively, a SID value of 0.32 is a good prediction which generally tends to match the location and magnitude of all major peaks and the location of most minor peaks, while smoothing some of the details of peak shape. To give some context to this value, we also provide some simple baselines for comparison. The average SID of a uniform distribution against the dataset was 2.52. The average SID of a roundrobin pairing of every spectrum in the dataset with every other spectrum in the dataset was 2.89. The average SID of a normalized sum of all the spectra against each individual member of the dataset was 1.45.

5. Uncertainty estimation for QM9 gap

Table IX summarizes the performance of three uncertainty quantification (UQ) methods (ensemble, evidential, and mean-variance estimation (MVE)) selected from Chemprop's available UQ options. For the evidential uncertainty, we used the total uncertainty (the sum

of aleatoric and epistemic components). We trained all models on the gap values from the QM9 dataset and calibrated the predictions using the z-scaling method[108] with the standard deviation as the regression calibrator metric. We then evaluated the methods based on four metrics: negative log likelihood (NLL), Spearman rank correlation (ρ), expected normalized calibration error (ENCE), and miscalibration area (MA). On this task, we observe that MVE performs the best across all four metrics, while ensemble performs the worst across all metrics, with evidential in between. However, we emphasize that UQ performance can vary depending on task, dataset size, representation, and other factors. The results presented here simply serve as an illustration of Chemprop UQ capabilities; we refer readers to substantial UQ benchmarking work done previously for further discussion of the merits and pitfalls of various methods and metrics on chemical and materials data[113,117–121].

C. Timing

Training and inference timing benchmarks for Chemprop can be found in Tables X, XI, and XII. These benchmarks were measured on three systems: a compute cluster node with CPU only, a compute cluster node with a GPU resource, and a laptop. We used an Intel Xeon Platinum 8260 processor (2.4 GHz, 48 CPU cores) for cluster CPU benchmarks and an Intel Xeon Gold 6248 (2.5 GHz, 40 CPU cores) processor with an Nvidia Volta V100 GPU for the cluster GPU benchmark timing. Both devices are part of the MIT Supercloud[122]. For both systems, we restricted the maximum numbers of CPU cores accessible to Chemprop to 8. For laptop timing, we used a Thinkpad X1 Carbon with an Intel Core i7-1280P (1.8 GHz, 14 CPU cores) processor and no enabled GPU. Our benchmark datasets were randomly sampled subsets of the QM9 HOMO-LUMO gap targets with sizes of 100,000, 10,000, and 1,000.

The training times for Chemprop models found in Table X includes all training processes, including time for data preprocessing and model evaluation. This training was carried out with a 80/10/10 training-validation-test split of the data. The hyperparameters were chosen to be in the typical range used for datasets of this size: hidden size of 1000, feed forward hidden size of 1000, 4 message passing layers, 2 feed forward layers, and 50 epochs. The training times found in Table XI are average training times on a per epoch basis. This average time includes the feed forward, loss function calculation, model update, and model evaluation steps. The average time excludes the time taken in the first training epoch, data preprocessing, and final model evaluation. Inference timing benchmarks can be found in Table XII. These times include all inference processes, including postprocessing of the predictions.

Training time shows significant speed improvement when moving from the laptop platform to the cluster CPU system and further improvement moving from the cluster CPU system to the cluster GPU system. This trend is followed across the tested dataset sizes. In each case, the speedup is greater than a factor of 2. Training for single models on moderately sized datasets can be carried out reasonably even on a laptop. For large datasets, hyperparameter optimization, and model structures involving many submodels, training on cluster resources or using a GPU is recommended. Inference times in the 10,000 and 100,000 dataset sizes are also improved when moving from laptop to cluster CPU to cluster GPU, but the progressive improvement is smaller than for training. Inference using any of the system levels tested is relatively fast for these dataset sizes.

Table X. Train times in hours:minutes:seconds for subsets of the QM9 dataset

Device	1k	10k	100k
Laptop	0:01:53	0:19:15	3:24:57
Cluster CPU	0:00:46	0:07:06	1:15:18
Cluster GPU	0:00:19	0:02:48	0:31:06

Table XI. Average training times for an epoch in seconds for subsets of the QM9 dataset, excluding the first epoch.

Device	1k	10k	100k
Laptop	2.2	23.1	245
Cluster CPU	0.9	8.4	90
Cluster GPU	0.3	3.2	36

V. CONCLUSION

We have presented the software package Chemprop, a powerful toolbox for machine learning of chemical properties of molecules and reactions. Significant improvements have been made to the software since its initial release and study,[36] including the support of multi-molecule properties, reactions, atom/bond-level properties, and spectra. Additionally, several state-of-the-art approaches to estimate the uncertainty of predictions have been incorporated, as well as pretraining and transfer learning procedures. Furthermore, the code now offers a variety of customization options, such as custom atom and bond features, a large variety of loss functions, and saving the learned feature embeddings for subsequent use with different algorithms. We have showcased and benchmarked Chemprop on a variety of tasks and datasets, and have found competitive performances for molecular property prediction compared to other approaches available on public leaderboards. Specifically, the performance for the prediction of water-octanol partition coefficients, reaction barrier heights, atomic partial charges, and absorption spectra set the current state-of-the-art. For properties depending on the three-dimensional conformation of a molecule such as in QM9 or PCQM4Mv2 Chemprop led

Table XII. Inference times in hours:minutes:seconds for subsets of the QM9 dataset.

Device	1k	10k	100k
Laptop	0:00:01	0:00:11	0:01:34
Cluster CPU	0:00:02	0:00:08	0:01:12
Cluster GPU	0:00:03	0:00:07	0:00:46

to performances comparable to the average reported in public leaderbords, but cannot outperform 3D models. In summary, Chemprop is a powerful, fast and convenient tool to learn conformation-independent properties of molecules, sets of molecules, or reactions.

DATA AND SOFTWARE AVAILABILITY

Chemprop, including all features described in this manuscript, is available open-source on GitHub, `github.com/chemprop/chemprop`. An extensive documentation including tutorials is available online,[88] including a workshop on YouTube.[97] Scripts and data splits to fully reproduce this study are available on GitHub, `github.com/chemprop/chemprop_benchmark` and on Zenodo, `doi.org/10.5281/zenodo.8174267` respectively.

ACKNOWLEDGEMENT

E.H. acknowledges support from the Austrian Science Fund (FWF), project J-4415. K.P.G. was supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. 1745302. Parts of the data reported within this paper were generated with resources from the MIT SuperCloud Lincoln Laboratory Supercomputing Center.[122]

REFERENCES

[1] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, in *International conference on machine learning* (PMLR, 2017) pp. 1263–1272.
[2] J. Klicpera, J. Groß, and S. Günnemann, arXiv preprint arXiv:2003.03123 (2020).
[3] S. Zhang, Y. Liu, and L. Xie, arXiv preprint arXiv:2011.07457 (2020).
[4] F. H. Vermeire, Y. Chung, and W. H. Green, J. Am. Chem. Soc. **144**, 10785–10797 (2022).
[5] M. R. Dobbelaere, Y. Ureel, F. H. Vermeire, L. Tomme, C. V. Stevens, and K. M. Van Geem, Ind. Eng. Chem. Res. **61**, 8581 (2022).
[6] M. R. Dobbelaere, P. P. Plehiers, R. Van de Vijver, C. V. Stevens, and K. M. Van Geem, J. Phys. Chem. A **125**, 5166 (2021).
[7] J. G. Rittig, K. Ben Hicham, A. M. Schweidtmann, M. Dahmen, and A. Mitsos, Comput. Chem. Eng. **171**, 108153 (2023).
[8] J. G. Rittig, M. Ritzert, A. M. Schweidtmann, S. Winkler, J. M. Weber, P. Morsch, K. A. Heufer, M. Grohe, A. Mitsos, and M. Dahmen, AIChE J. **69**, e17971 (2023).
[9] L. Fleitmann, P. Ackermann, J. Schilling, J. Kleinekorte, J. G. Rittig, F. vom Lehn, A. M. Schweidtmann, H. Pitsch, K. Leonhard, A. Mitsos, A. Bardow, and M. Dahmen, Energ. Fuel. **37**, 2213 (2023).
[10] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, *et al.*, Cell **180**, 688 (2020).
[11] N. De Cao and T. Kipf, arXiv preprint arXiv:1805.11973 (2018).
[12] Y. Dan, Y. Zhao, X. Li, S. Li, M. Hu, and J. Hu, Npj Comput. Mater. **6**, 1 (2020).
[13] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik, ACS Cent. Sci. **4**, 268 (2018).
[14] J. N. Wei, D. Duvenaud, and A. Aspuru-Guzik, ACS Cent. Sci. **2**, 725 (2016).
[15] M. H. Segler and M. P. Waller, Chem. Eur. J. **23**, 5966 (2017).
[16] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen, ACS Cent. Sci. **3**, 434 (2017).
[17] C. W. Coley, W. H. Green, and K. F. Jensen, Acc. Chem. Res. **51**, 1281 (2018).
[18] S. Szymkuć, E. P. Gajewska, T. Klucznik, K. Molga, P. Dittwald, M. Startek, M. Bajczyk, and B. A. Grzybowski, Angew. Chem. Int. Ed. **55**, 5904 (2016).
[19] M. H. Segler, M. Preuss, and M. P. Waller, Nature **555**, 604 (2018).
[20] M. A. Kayala, C.-A. Azencott, J. H. Chen, and P. Baldi, J. Chem. Inf. Model. **51**, 2209 (2011).
[21] M. A. Kayala and P. Baldi, J. Chem. Inf. Model. **52**, 2526 (2012).
[22] D. Fooshee, A. Mood, E. Gutman, M. Tavakoli, G. Urban, F. Liu, N. Huynh, D. Van Vranken, and P. Baldi, Mol. Syst. Des. Eng. **3**, 442 (2018).
[23] J. Bradshaw, M. J. Kusner, B. Paige, M. H. Segler, and J. M. Hernández-Lobato, arXiv preprint arXiv:1805.10970 (2018).
[24] H. Bi, H. Wang, C. Shi, C. Coley, J. Tang, and H. Guo, in *International Conference on Machine Learning* (PMLR, 2021) pp. 904–913.
[25] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen, Chem. Sci. **10**, 370 (2019).
[26] M. Sacha, M. Błaz, P. Byrski, P. Dabrowski-Tumanski, M. Chrominski, R. Loska, P. Włodarczyk-Pruszynski, and S. Jastrzebski, J. Chem. Inf. Model. **61**, 3273 (2021).
[27] P. Schwaller, T. Gaudin, D. Lanyi, C. Bekas, and T. Laino, Chem. Sci. **9**, 6091 (2018).
[28] C. McGill, M. Forsuelo, Y. Guan, and W. H. Green, J. Chem. Inf. Model. **61**, 2594 (2021).
[29] K. P. Greenman, W. H. Green, and R. Gómez-Bombarelli, Chem. Sci. **13**, 1152 (2022).
[30] K. Dührkop, Bioinf. **38**, i342 (2022).
[31] D. H. Nguyen, C. H. Nguyen, and H. Mamitsuka, Bioinf. **35**, i164 (2019).
[32] D. H. Nguyen, C. H. Nguyen, and H. Mamitsuka, Brief. Bioinf. **20**, 2028 (2019).
[33] M. A. Stravs, K. Dührkop, S. Böcker, and N. Zamboni, Nat. Methods **19**, 865 (2022).
[34] E. N. Muratov, J. Bajorath, R. P. Sheridan, I. V. Tetko, D. Filimonov, V. Poroikov, T. I. Oprea, I. I. Baskin, A. Varnek, A. Roitberg, *et al.*, Chem. Soc. Rev. **49**, 3525 (2020).
[35] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, J. Comput. Aided Mol. Des. **30**, 595 (2016).
[36] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, *et al.*, J. Chem. Inf. Model. **59**, 3370 (2019).
[37] Ł. Maziarka, T. Danel, S. Mucha, K. Rataj, J. Tabor, and S. Jastrzebski, arXiv preprint arXiv:2002.08264 (2020).

[38]D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, Adv. Neural Inf. Process. Syst. **34**, 21618 (2021).
[39]K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, J. Chem. Phys. **148**, 241722 (2018).
[40]O. T. Unke and M. Meuwly, J. Chem. Theory Comput. **15**, 3678 (2019).
[41]J. Gasteiger, J. Groß, and S. Günnemann, arXiv preprint arXiv:2003.03123 (2020).
[42]F. Bigi, S. N. Pozdnyakov, and M. Ceriotti, arXiv preprint arXiv:2303.04124 (2023).
[43]B. Winter, C. Winter, J. Schilling, and A. Bardow, Digital Discovery **1**, 859 (2022).
[44]V. Bagal, R. Aggarwal, P. Vinod, and U. D. Priyakumar, J. Chem. Inf. Model. **62**, 2064 (2021).
[45]S. Honda, S. Shi, and H. R. Ueda, arXiv preprint arXiv:1911.04738 (2019).
[46]S. Chithrananda, G. Grand, and B. Ramsundar, arXiv preprint arXiv:2010.09885 (2020).
[47]M. Haghighatlari, J. Li, F. Heidar-Zadeh, Y. Liu, X. Guan, and T. Head-Gordon, Chem **6**, 1527 (2020).
[48]T. Frank, O. Unke, and K.-R. Müller, Adv. Neural Inf. Process. Syst. **35**, 29400 (2022).
[49]S. M. Kandathil, J. G. Greener, and D. T. Jones, Proteins: Struct. Funct. Bioinf. **87**, 1179 (2019).
[50]W. Jin, J. M. Stokes, R. T. Eastman, Z. Itkin, A. V. Zakharov, J. J. Collins, T. S. Jaakkola, and R. Barzilay, Proc. Natl. Acad. Sci. U.S.A. **118** (2021), 10.1073/pnas.2105070118.
[51]M. A. Lim, S. Yang, H. Mai, and A. C. Cheng, J. Chem. Inf. Model. **62**, 6336 (2022).
[52]Y. Guan, C. W. Coley, H. Wu, D. Ranasinghe, E. Heid, T. J. Struble, L. Pattanaik, W. H. Green, and K. F. Jensen, Chem. Sci. **12**, 2198 (2021).
[53]Y. Chung, F. H. Vermeire, H. Wu, P. J. Walker, M. H. Abraham, and W. H. Green, J. Chem. Inf. Model. **62**, 433 (2022).
[54]F. H. Vermeire, Y. Chung, and W. H. Green, J. Am. Chem. Soc. **144**, 10785 (2022).
[55]C. Isert, J. C. Kromann, N. Stiefl, G. Schneider, and R. A. Lewis, ACS Omega **8**, 2046 (2023).
[56]Y. Chung and W. H. Green, ChemRxiv (2023), 10.26434/chemrxiv-2023-f20bg.
[57]S. Biswas, Y. Chung, J. Ramirez, H. Wu, and W. H. Green, J. Chem. Inf. Model. (2023), accepted.
[58]T. Larsson, F. Vermeire, and S. Verhelst, SAE Tech. Pap. , 2023 (2023).
[59]E. Heid and W. H. Green, J. Chem. Inf. Model. **62**, 2101 (2022).
[60]K. A. Spiekermann, L. Pattanaik, and W. H. Green, J. Phys. Chem. A **126**, 3976 (2022).
[61]G. Landrum, "Rdkit: Open-source cheminformatics," (2006), https://www.rdkit.org/.
[62]A. M. Schweidtmann, J. G. Rittig, J. M. Weber, M. Grohe, M. Dahmen, K. Leonhard, and A. Mitsos, Comput. Chem. Eng. **172**, 108202 (2023).
[63]H. L. Morgan, J. Chem. Doc. **5**, 107 (1965).
[64]"Descriptor computation(chemistry) and (optional) storage for machine learning," `https://github.com/bp-kelley/descriptastorus`, (accessed April 6 2023).
[65]D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2017).
[66]Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, Chem. Sci. **9**, 513 (2018).
[67]W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," (2021).
[68]R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. Von Lilienfeld, Sci. Dat. **1**, 140022 (2014).
[69]"The SAMPL Challenges," `https://github.com/samplchallenges`, (accessed March 15, 2023).
[70]B. Koscher, R. B. Canty, M. A. McDonald, K. P. Greenman, C. J. McGill, C. L. Bilodeau, W. Jin, H. Wu, F. H. Vermeire, B. Jin, *et al.*, ChemRxiv (2023), 10.26434/chemrxiv-2023-r7b01.
[71]P. C. St. John, Y. Guan, Y. Kim, S. Kim, and R. S. Paton, "BDE-db: a collection of 290,664 homolytic bond dissociation enthalpies for small organic molecules," `https://doi.org/10.6084/m9.figshare.10248932.v1` (2019), (accessed March 22, 2023).
[72]P. C. St. John, Y. Guan, Y. Kim, S. Kim, and R. S. Paton, Nat. Commun. **11**, 2328 (2020).
[73]P. Bleiziffer, K. Schaller, and S. Riniker, J. Chem. Inf. Model. **58**, 579 (2018).
[74]G. F. von Rudorff, S. N. Heinen, M. Bragato, and O. A. von Lilienfeld, Mach. Learn.: Sci. Technol. **1**, 045026 (2020).
[75]T. Stuyver and C. W. Coley, J. Chem. Phys. **156**, 084104 (2022).
[76]S. Heinen, G. F. von Rudorff, and O. A. von Lilienfeld, J. Chem. Phys. **155**, 064105 (2021).
[77]T. Stuyver, K. Jorner, and C. W. Coley, Sci. Dat. **10**, 66 (2023).
[78]K. Spiekermann, L. Pattanaik, and W. H. Green, Sci. Dat. **9**, 417 (2022).
[79]Q. Zhao, S. M. Vaddadi, M. Woulfe, L. A. Ogunfowora, S. S. Garimella, O. Isayev, and B. M. Savoie, Sci. Dat. **10**, 145 (2023).
[80]J. F. Joung, M. Han, M. Jeong, and S. Park, Sci. Dat. **7**, 1 (2020).
[81]C.-W. Ju, H. Bai, B. Li, and R. Liu, J. Chem. Inf. Model. **61**, 1053 (2021).
[82]V. Venkatraman, R. Raju, S. P. Oikonomopoulos, and B. K. Alsberg, J. Cheminf. **10**, 1 (2018).
[83]V. Venkatraman and L. Kallidanthiyil Chellappan, Data **5**, 45 (2020).
[84]NIST Mass Spectrometry Data Center, *"Infrared Spectra" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69*, edited by P. J. Lindstrom and W. G. Mallard (National Institute of Standards and Testing, Gaithersburg, MD) `http://webbook.nist.gov` (Accessed 2019-10-3).
[85]M. Nakata and T. Shimazaki, J. Chem. Inf. Model. **57**, 1300–1308 (2017).
[86]J. S. Delaney, J. Chem. Inf. Comput. **44**, 1000 (2004).
[87]"Chemprop: Molecular property prediction," `https://www.github.com/chemprop/chemprop` (), (accessed April 6 2023).
[88]"Chemprop," `https://chemprop.readthedocs.io/en/latest/` (), (accessed April 6 2023).
[89]Y. Wang, J. Xiao, T. O. Suzek, J. Zhang, J. Wang, Z. Zhou, L. Han, K. Karapetyan, S. Dracheva, B. A. Shoemaker, E. Bolton, A. Gindulyte, and S. H. Bryant, Nucleic Acids Res. **40**, D400 (2012).
[90]T. Stuyver and C. Coley, Chem. Eur. J. , e202300387 (2023).
[91]K. C. Felton, H. Ben-Safar, and A. Alexei, in *1st Annual AAAI Workshop on AI to Accelerate Science and Engineering (AI2ASE)* (2022).
[92]A. D. McNaughton, R. P. Joshi, C. R. Knutson, A. Fnu, K. J. Luebke, J. P. Malerich, P. B. Madrid, and N. Kumar, J. Chem. Inf. Model. **63**, 1462 (2023).
[93]G. Liu, D. B. Catacutan, K. Rathod, K. Swanson, W. Jin, J. C. Mohammed, A. Chiappino-Pepe, S. A. Syed, M. Fragis, K. Rachwalski, J. Magolan, M. G. Surette, B. K. Coombes, T. Jaakkola, R. Barzilay, J. J. Collins, and J. M. Stokes, Nat. Chem. Biol. (2023), 10.1038/s41589-023-01349-8.
[94]Z. Fralish, A. Chen, P. Skaluba, and D. Reker, (2023), 10.26434/chemrxiv-2023-gbchq.
[95]M. Colomba, S. Benedetti, D. Fraternale, A. Guidarelli, S. Coppari, V. Freschi, R. Crinelli, G. E. N. Kass, A. Gorassini, G. Verardo, C. Roselli, M. A. Meli, B. Di Giacomo, and M. C. Albertini, Nutrients **15**, 2132 (2023).
[96]F. Wong, S. Omori, N. M. Donghia, E. J. Zheng, and J. J. Collins, Nature Aging **3**, 734 (2023).
[97]"Chemprop workshop," `https://www.youtube.com/watch?v=TeOl5E8Wo2M` (), (accessed April 6 2023).
[98]G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, arXiv preprint arXiv:1207.0580 (2012).

[99]C.-I. Chang, IEEE Transactions on Information Theory **46**, 1927–1932 (2000).
[100]D. A. Nix and A. S. Weigend, in *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, Vol. 1 (IEEE, 1994) pp. 55–60.
[101]A. Amini, W. Schwarting, A. Soleimany, and D. Rus, Adv. Neural Inf. Process. Syst. **33**, 14927 (2020).
[102]M. Sensoy, L. Kaplan, and M. Kandemir, Adv. Neural Inf. Process. Syst. **31** (2018).
[103]C. Villani *et al.*, *Optimal transport: old and new*, Vol. 338 (Springer, 2009).
[104]J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, Adv. Neural Inf. Process. Syst. **24** (2011).
[105]J. Bergstra, D. Yamins, and D. Cox, in *Proceedings of the 30th International Conference on Machine Learning* (PMLR, 2013) pp. 115–123, iSSN: 1938-7228.
[106]B. Lakshminarayanan, A. Pritzel, and C. Blundell, Adv. Neural Inf. Process. Syst. **30** (2017).
[107]Y. Gal and Z. Ghahramani, in *Proceedings of The 33rd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 48, edited by M. F. Balcan and K. Q. Weinberger (PMLR, New York, New York, USA, 2016) pp. 1050–1059.
[108]D. Levi, L. Gispan, N. Giladi, and E. Fetaya, arXiv preprint arXiv:1905.11659 (2019).
[109]E. Zelikman, C. Healy, S. Zhou, and A. Avati, arXiv preprint arXiv:2005.12496 (2020).
[110]D. Wang, J. Yu, L. Chen, X. Li, H. Jiang, K. Chen, M. Zheng, and X. Luo, J. Cheminf. **13**, 1 (2021).
[111]C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, in *International conference on machine learning* (PMLR, 2017) pp. 1321–1330.
[112]B. Zadrozny and C. Elkan, in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002) pp. 694–699.
[113]G. Scalia, C. A. Grambow, B. Pernici, Y.-P. Li, and W. H. Green, J. Chem. Inf. Model. **60**, 2697 (2020).
[114]H. Altae-Tran, B. Ramsundar, A. S. Pappu, and V. S. Pande, arXiv preprint arXiv:1611.03199 (2016), 10.48550/arXiv.1611.03199.
[115]"Heterogeneous interpolation on graph," `https://github.com/TencentYoutuResearch/HIG-GraphClassification` (2021), (accessed May 28 2023).
[116]Y. Sun, T. Hou, X. He, V. H. Man, and J. Wang, J. Comp. Chem. **44**, 1300 (2023).
[117]K. Tran, W. Neiswanger, J. Yoon, Q. Zhang, E. Xing, and Z. W. Ulissi, Machine Learning: Science and Technology **1**, 025006 (2020).
[118]L. Hirschfeld, K. Swanson, K. Yang, R. Barzilay, and C. W. Coley, J. Chem. Inf. Model. **60**, 3770 (2020).
[119]A. Nigam, R. Pollice, M. F. Hurley, R. J. Hickman, M. Aldeghi, N. Yoshikawa, S. Chithrananda, V. A. Voelz, and A. Aspuru-Guzik, Expert Opin. Drug Discov. **16**, 1009 (2021).
[120]A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia, and C. W. Coley, ACS Cent. Sci. **7**, 1356 (2021).
[121]E. Heid, C. J. McGill, F. H. Vermeire, and W. H. Green, (2023), 10.26434/chemrxiv-2023-00vcg-v2.
[122]A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, in *2018 IEEE High Performance extreme Computing Conference (HPEC)* (IEEE, 2018) pp. 1–6.