A Novel Message Passing Neural Network For Molecular Property Prediction

E-mail:

Introduction

One of the oldest cheminformatics tasks, molecular property prediction has received new attention in light of recent advancements in deep neural networks. These architectures either operate over fixed molecular fingerprints common in traditional QSAR models, or they learn their own task-specifc representations using graph convolutions.[1–11] Both approaches are reported to yield substantial performance gains, continuously raising state-of-the-art accuracy in property prediction.

Despite these successes, many unanswered questions remain. The first question concerns the comparison between learned molecular representations and fingerprints/descriptors. Unfortunately, current published results on this topic do not provide a clear answer. Wu et al. demonstrate that convolution-based models typically outperform fingerprint-based models, while experiments reported in Mayr et al. report the opposite results. Part of these discrepancies can be attributed to differences in evaluation setup including the way datasets are constructed. This leads us to a broader question concerning current evaluation protocols and their capacity to measure the generalization power of a method when applied to a new chemical space, as is common in drug discovery. Unless special care is taken to replicate this distributional shift in evaluation, neural models may overfit the training data but still score highly on the test data. This is particularly true for convolutional models that can learn a poor molecular representation by memorizing the molecular scaffolds in the training data and thereby failing to generalize to new ones. Therefore, a meaningful evaluation of property prediction models needs to explicitly account for scaffold overlap between train and test data in light of generalization requirements.

In this paper, we aim to answer both of these questions by designing a comprehensive evaluation setup for assessing neural architectures. We also introduce a novel algorithm for property prediction that achieves new state-of-the-art performance across a range of datasets. The model has two distinctive features: (1) It operates over a hybrid representation that combines convolutions and fingerprints. This design gives it flexibility in learning a task specific encoding, while providing strong regularization with fixed fingerprints. (2) It learns to construct molecular encodings by using convolutions centered on bonds instead of atoms, thereby avoiding unnecessary loops during the message passing phase of the algorithm and resulting in superior empirical performance.

We evaluate our model and other recently published neural architectures on both publicly available benchmarks, such as those from Wu et al. and Mayr et al., as well as proprietary datasets from Amgen, Novartis, and BASF. Our goal is to assess whether the model's performance on the public datasets and their relative ranking are representative of their ranking on the proprietary datasets. We demonstrate that under a scaffold split of training and testing data, the relative ranking of the models is consistent across the two classes of datasets. We also show that a scaffold-based split of the training and testing data is a good approximation of the temporal split commonly used in industry, in absolute terms of the relevant metrics. By contrast, a purely random split is a poor approximation to a temporal split. To put the state-of-the-art performance in perspective, we report bounds on experimental error and show that there is still room for improving deep learning models to match the accuracy of screening results.

Building on the diversity of our benchmark datasets, we explore the impact of molecular representation with respect to the dataset characteristics. We found that a hybrid representation yields higher performance and generalizes better than either convolution-based or fingerprint-based models. We also note that on small datasets (up to 1000 training molecules) fingerprint models outperform learned representations, which are negatively impacted by data sparsity. Beyond molecular representation issues, we observe that hyperparameter selection plays a crucial role in model performance, consistent with prior work.[13] We show that Bayesian optimization yields a robust, automatic solution to this issue. The addition of ensembling further improves accuracy, again consistent with the literature.[14]

In aggregate, we conclude that our model achieves consistently strong out-of-the-box performance across a wide range of public as well as proprietary datasets, indicating its applicability as a useful tool for chemists actively working on drug discovery.

Our code is publicly available at `https://github.com/swansonk14/chemprop` and includes a web interface that supports non-programmatic access to the model.

Background

Since the core element of our model is a novel graph encoder architecture, our work is closely related to previous work on graph encoders, such as for social networks[6,15] and for chemistry applications.[1,7–9,16–23]

For molecular property prediction specifically, development of QSAR methods began with the application of well-known models like support vector machines[24] or random forests[25] to expert-engineered descriptors or molecular fingerprints, such as the Dragon descriptors[26] or ECFP/Morgan fingerprints.[27] One direction of advancement is the use of domain expert-driven improvement of the base feature representation, in the form of various molecular descriptors,[26,28–31] to drive better performance.[12] Similarly, for datasets with explicit 3D atomic coordinates (which can be difficult for a model to learn implicitly from just a SMILES[32] string), many studies have leveraged this additional information to improve performance significantly.[2,33–36]

The main other line of research is the optimization of the model architecture, whether applying a feed-forward neural network or other model on top of descriptors and fingerprints,[12,37] or even forgoing expert-engineered descriptors entirely to run a neural network directly on SMILES strings[12] or the molecular graph.[1–11] Our model belongs to the last category of models, known as graph convolutional neural networks. In essence, such models learn their own expert feature representations from the data, and have been shown to be very flexible and capable of capturing complex relationships given sufficient data.[2,4] Like us, Liu et al. also evaluate their model against private industry datasets, but we cannot compare against their method directly owing to dataset differences.[38] Finally, Ishiguro et al.[39] make a strong improvement to graph neural networks in a direction orthogonal to our own improvements.

The property prediction models most similar to our own are encapsulated in the Message Passing Neural Network (MPNN) framework presented in Gilmer et al..[4] We build upon this basic framework by contributing a novel message-passing paradigm based on updating representations of directed bonds rather than atoms. Additionally, we further improve the model by combining computed molecule-level features with the molecular representation learned by the MPNN.

Methods

We will first summarize MPNNs in general using the terminology of Gilmer et al., and then expand on the particular novel characteristics of our own model, which we refer to as the Directed MPNN (D-MPNN).

Message Passing Neural Networks

An MPNN is a model which operates on an undirected graph G with node (atom) features x_v and edge (bond) features e_{vw}. MPNNs operate in two phases: a *message passing phase*, which transmits information across the molecule to build a neural representation of the molecule, and a *readout phase*, which uses the final representation of the molecule to make predictions about the properties of interest.

More specifically, the message passing phase consists of T steps. On each step t, hidden states h_v^t and messages m_v^t, associated with each vertex v, are updated using message function M_t and vertex update function U_t according to:

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

where $N(v)$ is the set of neighbors of v in graph G, and h_v^0 is some function of the initial atom features. The readout phase then uses a readout function R to make a property prediction based on the final hidden states according to

$$\hat{y} = R(\{h_v^T | v \in G\}).$$

The output $\hat{y}$ may be either a scalar or a vector, depending on whether the MPNN is designed to predict a single property or multiple properties (in a multitask setting).

Directed MPNN

The main difference between the Directed MPNN (D-MPNN) and the generic MPNN described above is the nature of the messages sent during the message passing phase. Rather than using messages associated with vertices (atoms), our model uses messages associated with directed edges (bonds). The motivation of this design is to prevent totters[40], that is, to avoid messages being passed along any path of the form $v_1 v_2 \cdots v_n$ where $v_i = v_{i+2}$ for some i. Such excursions are likely to introduce noise to the graph representation. Using Figure 1 as illustration, in our model, the message $1 \rightarrow 2$ will be propagated to node 3 and 4 only in

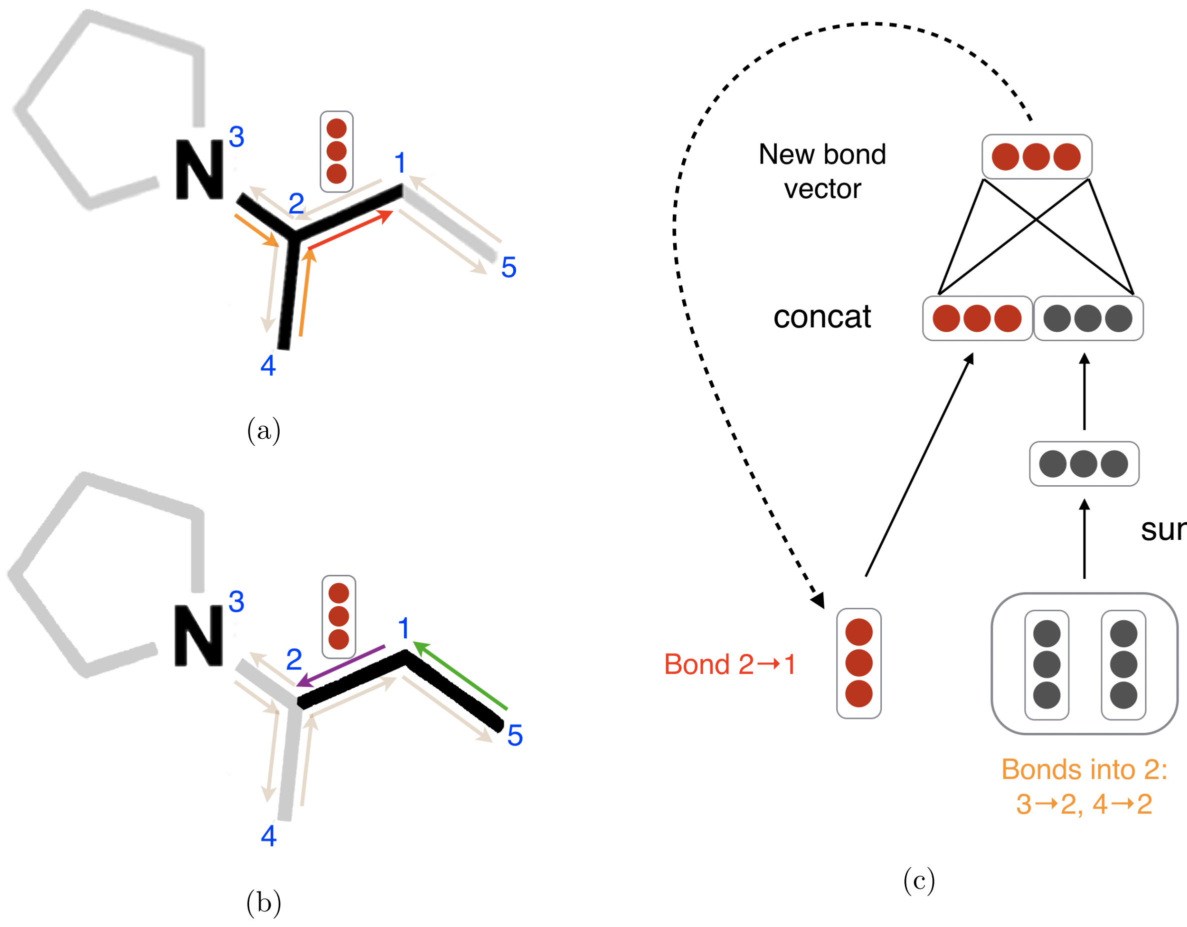


Figure 1: Illustration of bond-level message passing. (a): Messages from the orange directed bonds are used to inform the update to the hidden state of the red directed bond. (b): Similarly, a message from the green bond informs the update to the hidden state of the purple directed bond. (c): Illustration of the update function to the hidden representation of the red directed bond from the top left diagram.

the next iteration, whereas it will be sent to node 1 again in the original MPNN, creating unnecessary loops in the message passing trajectory. This message passing procedure is akin to belief propagation in probabilistic graphical models.[41] We refer to Dai et al. for more discussions about the connection between D-MPNN and belief propagation.

Specifically, the D-MPNN operates on hidden states h_{vw}^t and messages m_{vw}^t instead of node based hidden states h_v^t where the direction of messages matters (i.e., h_{vw}^t is distinct from h_{wv}^t). The corresponding message passing update equations are thus

$$m_{vw}^{t+1} = \sum_{k \in \{N(v) \setminus w\}} M_t(x_v, x_k, h_{kv}^t)$$

$$h_{vw}^{t+1} = U_t(h_{vw}^t, m_{vw}^{t+1}).$$

where message m_{vw}^{t+1} does not depend on its reverse message m_{wv}^t from previous iteration. Prior to the first step of message passing, we initialize edge hidden states with

$$h_{vw}^0 = \sigma(W_i \, \texttt{cat}(x_v, e_{vw}))$$

where $W_i \in \mathbb{R}^{h \times h_i}$ is a learned matrix and $\texttt{cat}(x_v, e_{vw}) \in \mathbb{R}^{h_i}$ is the concatenation of the atom features x_v for atom v and the bond features e_{vw} for bond vw.

We choose to use relatively simple message passing functions M_t and edge update functions U_t. Specifically, we define $M_t(x_v, x_w, h_{vw}^t) = h_{vw}^t$ and we implement U_t with the same neural network on every step,

$$U_t(h_{vw}^t, m_{vw}^{t+1}) = U(h_{vw}^t, m_{vw}^{t+1}) = \sigma(h_{vw}^0 + W_m m_{vw}^{t+1})$$

where $W_m \in \mathbb{R}^{h \times h}$ is a learned matrix with hidden size h and σ is the ReLU activation function.[42] Note that the addition of h_{vw}^0 on every step provides a skip connection to the original feature vector for that edge.

Finally, we return to an atom representation of the molecule by summing the incoming bond features according to

$$m_v = \sum_{w \in N(v)} h_{vw}^T$$
$$h_v = \sigma(W_a \texttt{cat}(x_v, m_v))$$

where $W_a \in \mathbb{R}^{h \times h}$ is a learned matrix.

Altogether, the D-MPNN message passing phase operates according to

$$h_{vw}^0 = \sigma(W_i \ \texttt{cat}(x_v, e_{vw}))$$

followed by

$$m_{vw}^{t+1} = \sum_{k \in \{N(v) \setminus w\}} h_{kv}^t$$
$$h_{vw}^{t+1} = \sigma(h_{vw}^0 + W_m m_{vw}^{t+1})$$

for $t \in \{0, 1, \ldots, T\}$, followed by

$$m_v = \sum_{w \in N(v)} h_{vw}^T$$
$$h_v = \sigma(W_a \texttt{cat}(x_v, m_v)).$$

The readout phase of the D-MPNN is identical to the readout phase of a generic MPNN. In our implementation of the readout function R, we first sum the atom hidden states to obtain a feature vector for the molecule

$$h = \sum_{v \in G} h_v.$$

Finally, we generate property predictions $\hat{y} = f(h)$ where $f(\cdot)$ is a feed-forward neural network.

Features

We now describe in detail our model's featurization, which we construct using the open-source RDKit.[43]

Table 1: Atom Features. All features are one-hot encodings except for atomic mass, which is a real number scaled to be on the same order of magnitude.

Feature	**Description**	**Size**
Atom type	Type of atom (ex. C, N, O).	100
# Bonds	Number of bonds the atom is involved in.	6
Formal charge	Integer electronic charge assigned to atom.	5
Chirality	Unspecified, tetrahedral CW/CCW, or other.	4
# Hs	Number of bonded Hydrogen atom.	5
Hybridization	sp, sp^2, sp^3, sp^3d, or sp^3d^2.	5
Aromaticity	Whether this atom is part of an aromatic system.	1
Atomic mass	Mass of the atom, divided by 100.	1

Table 2: Bond Features. All features are one-hot encodings.

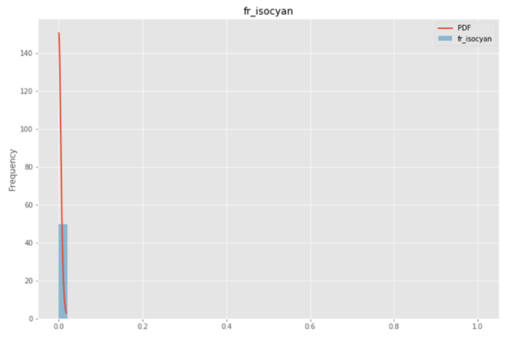
Feature	**Description**	**Size**
Bond type	Single, double, triple, or aromatic.	4
Conjugated	Whether the bond is conjugated.	1
In ring	Whether the bond is part of a ring.	1
Stereo	None, any, E/Z, cis/trans.	6

In the featurization process, our D-MPNN model begins by calculating the features for each atom, listed in Table 1. Then each directed bond takes on the concatenation of the features of the atom that it originates from and the bond-specific features in Table 2.

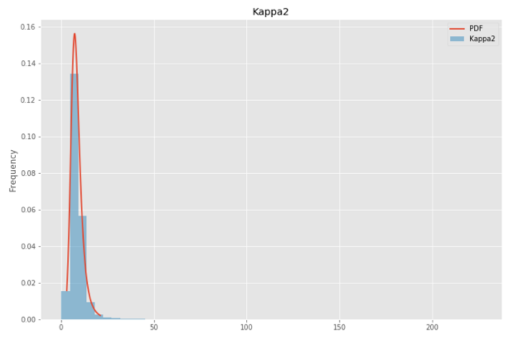
We implement our model using PyTorch.[44]

D-MPNN with Features

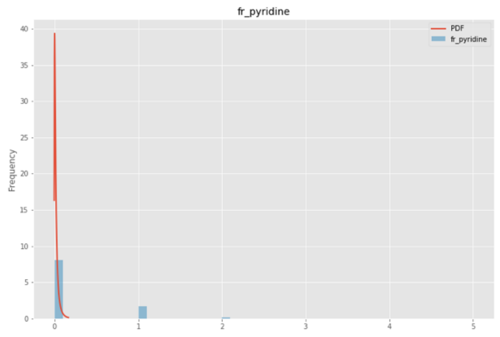
Finally, we discuss further extensions and optimizations to performance. Although an MPNN should ideally be able to extract *any* information about a molecule that might be relevant



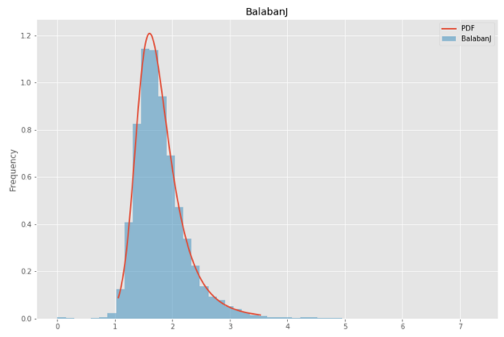
(a) fr_isocyan



(b) Kappa2



(c) fr_pyridine



(d) BalabanJ

Figure 2: Four example distributions fit to a random sample of 100,000 of compounds used for biological screening in Novartis. Note that some distributions for discrete calculations, such as fr_pyridine are not fit especially well. This is an active area for improvement.

to predicting a given property, two limitations may prevent this in practice. First, many property prediction datasets are very small, i.e., on the order of only hundreds or thousands of molecules. With so little data, MPNNs are unable to learn to identify and extract all features of a molecule which might be relevant to property prediction, and are susceptible to overfitting to artifacts in the data. Second, most MPNNs use fewer message passing steps than the diameter of the molecular graph, i.e. $T < \texttt{diam}(G)$, meaning atoms that are a distance of greater than T bonds apart will never receive messages about each other. This results in a molecular representation that is fundamentally local rather than global in nature, meaning the MPNN will most likely struggle to predict properties that depend on global features like atom count.

In order to counter these limitations, we introduce a variant of the D-MPNN that incorporates global molecular features that can be computed rapidly *in silico*, using RDKit. The neural network architecture requires that the features are appropriately scaled to prevent features with large ranges dominating smaller ranged features as well as preventing issues where features in the training set are not drawn from the same sample distribution as the testing sets. To prevent these issues, a large sample of molecules was used to fit cumulative density function (CDF) to all features. CDFs were used as opposed to simpler scaling algorithms mainly because CDFs have the useful property that each value has the same meaning: the percentage of the population observed below the raw feature value. Min-max scaling can be easily biased with outliers and Z-score scaling assumes a normal distribution which is most often not the case for chemical features, especially if they are based on counts.

The CDFs were fit to a sampling of 100k compounds from the Novartis internal catalog using the distributions available in the scikit-learn package,[45] a sampling of which can be seen in Figure 2. One could do a similar normalization using publicly available databases such as ZINC[46] and PubChem.[47] scikit-learn was used primarily due to the simplicity of fitting and the final application. However, more complicated techniques can be used in the future to fit to empirical CDFs such as finding the best fit general logistic function, which has been shown to be successful for other biological datasets.[48] No review was taken to remove odd distributions. For example, azides are hazardous and rarely used outside of a few specific reactions, as reflected in the fr_azide distribution in Figure 2. As such, since the sample data was primarily used for chemical screening against biological targets, the distribution used here may not accurately reflect the distribution of reagents used for chemical synthesis. For the full list of calculated features, refer to the supplementary material. Code for computing and using the CDFs is available at `https://github.com/bp-kelley/descriptastorus`.

To incorporate these features, we modify the readout phase of the D-MPNN to apply the feed-forward neural network f to the concatenation of the learned molecule feature vector h and the computed global features h_f,

$$\hat{y} = f(\texttt{cat}(h, h_f)).$$

This is a very general method of incorporating external information and can be used with any MPNN and any computed features or descriptors.

Hyperparameter Optimization

The performance of MPNNs, like most neural networks, can depend largely on the settings of the various model hyperparameters, such as the hidden size of the neural network layers. Thus to maximize performance, we perform hyperparameter optimization via Bayesian Optimization[13] using the `Hyperopt`[49] python package. We specifically optimize our model's depth (number of message-passing steps), hidden size (size of bond message vectors), number of feed-forward network layers, and dropout probability.

Ensembling

A common technique in machine learning for improving model performance is ensembling, where the predictions of multiple independently trained models are combined to produce a more accurate prediction.[14] We apply this technique by training several copies of our model, each initialized with a different random seed but trained on the same set of training data, and then averaging the predictions of the models (each with equal weight) to generate an ensemble prediction.

Since ensembling can be applied to any type of model and was not used in the prior work that we directly compare to, all direct comparisons are using a single model. However, we also report the results using an ensemble to illustrate the maximum possible performance using our model architecture.

Experiments

Data

Table 3: Summary statistics of the public datasets used in this paper. Note: PDBbind-F, PDBbind-C, and PDBbind-R refer to the full, core, and refined PDBbind datasets from ref. 2.

Category	**Dataset**	**# Tasks**	**Task Type**	**# Compounds**	**Metric**
Quantum Mechanics	QM7	1	Regression	6,834	MAE
Quantum Mechanics	QM8	12	Regression	21,786	MAE
Quantum Mechanics	QM9	12	Regression	133,884	MAE
Physical Chemistry	ESOL	1	Regression	1,128	RMSE
Physical Chemistry	FreeSolv	1	Regression	642	RMSE
Physical Chemistry	Lipophilicity	1	Regression	4,200	RMSE
Biophysics	PDBbind-F	1	Regression	9,880	RMSE
Biophysics	PDBbind-C	1	Regression	168	RMSE
Biophysics	PDBbind-R	1	Regression	3,040	RMSE
Biophysics	PCBA	128	Classification	437,928	PRC-AUC
Biophysics	MUV	17	Classification	93,087	PRC-AUC
Biophysics	HIV	1	Classification	41,127	ROC-AUC
Biophysics	BACE	1	Classification	1,513	ROC-AUC
Physiology	BBBP	1	Classification	2,039	ROC-AUC
Physiology	Tox21	12	Classification	7,831	ROC-AUC
Physiology	ToxCast	617	Classification	8,576	ROC-AUC
Physiology	SIDER	27	Classification	1,427	ROC-AUC
Physiology	ClinTox	2	Classification	1,478	ROC-AUC
Physiology	ChEMBL	1310	Classification	456,331	ROC-AUC

We test our model on 19 publicly available datasets from Wu et al. and Mayr et al.. These datasets include a wide range of regression and classification targets spanning quantum mechanics, physical chemistry, biophysics, and physiology. The datasets range in size from less than 200 molecules to over 450,000 molecules. Summary statistics for all the datasets are provided in Table 3[1]. Additional information on the class balance of the classification datasets is provided in the Supplementary Materials[2].

Experimental Procedure

Cross-Validation and Hyperparameter Optimization. Since many of the datasets are very small (two thousand molecules or fewer), we use a cross-validation approach to decrease noise in the results both while optimizing the hyperparameters and while determining final performance numbers. For consistency, we maintain the same approach for all of our datasets. Specifically, for each dataset, we use 50 iterations of Bayesian optimization on 3 randomly-seeded 80:10:10 data splits to determine the best hyperparameters. Then we use 10 additional randomly-seeded 80:10:10 data splits to determine the performance of the best set of hyperparameters. Due to computational cost, we used only 1 randomly-seeded data split (instead of 3) during hyperparameter optimization on QM9, MUV, PCBA, and ChEMBL. On PCBA and ChEMBL, we performed 20 iterations of Bayesian optimization and we used 3 data splits (instead of 10) when evaluating the best hyperparameters.

When optimizing hyperparameters, we consider it desirable to share a single set of hyperparameters across all of the 80:10:10 test splits, so that it is possible to report a single set of optimized hyperparameters for future use. Therefore, to determine the best hyperparameters, we take the set of hyperparameters with the best average performance on the 3

[1]For some datasets, the number of compounds in Table 3 does not precisely match the numbers from Wu et al. because we removed a small number of molecules which could not be processed by RDKit.[43] Furthermore, we have fewer molecules in QM7 because we used SMILES generated by Wu et al. from the original 3D coordinates in the dataset, but the SMILES conversion process failed for $\sim$ 300 molecules. (Note that QM7 and some of the other MoleculeNet datasets are more commonly used in benchmarking models that leverage 3D information, rather than purely SMILES-based methods like our own)

[2]The MUV dataset is particularly unbalanced, with only 0.2% of molecules classified as positive. This makes our model unstable, leading to the wide variety in performance in the subsequent sections.

validation splits. Although the hyperparameter selection may be informed by the full dataset as a result, we minimize this effect by testing on a new set of 10 randomly-seeded splits. When we run the best model from Mayr et al. for comparative purposes, we optimize their model's hyperparameters in the same fashion.

Split Type. While we only optimize the models in a random split of the data[3], we evaluate the optimized models on both random and scaffold-based splits, as well as on the original splits from Wu et al. and Mayr et al.. Our scaffold split is similar to that of Wu et al.. Molecules are partitioned into bins based on their Murcko scaffold calculated by RDKit.[43] Any bins larger than half of the desired test set size are placed into the training set, in order to guarantee the scaffold diversity of the validation and test sets. All remaining bins are placed randomly into the training, validation, and test sets until each set has reached its desired size. We use a 80%/10%/10% split for all of our experiments unless otherwise stated.

Compared to a random split, a scaffold split is a more challenging and realistic evaluation setting as shown in Figure 9 below. As real-world property prediction naturally involves a chronological data split, where one trains a model on past data to predict on future data, we seek to approximate the difficulty of this chronological split in our evaluations on datasets where chronological information is not available. Thus, with the exception of our comparison to the MoleculeNet models from Wu et al., where we use their original data splits, all of our evaluations are conducted on scaffold-based splits.

Baselines. We compare our model to the following baselines:

- The best model for each dataset from Wu et al.
- The best model on ChEMBL from Mayr et al., a feed-forward neural network on a concatenation of assorted expert-designed molecular fingerprints

[3]We performed hyperparameter optimization on several of the datasets using a scaffold-split, but we found that performance was comparable to using the hyperparameters selected from performing optimization on a random split. For time reasons, we therefore only performed optimization on a random split and used those hyperparameter settings for both random and scaffold splits.

- Random forest on binary Morgan fingerprints
- Feed-forward network on binary Morgan fingerprints
- Feed-forward network on count-based Morgan fingerprints
- Feed-forward network on RDKit-calculated fingerprints

The models in Wu et al. include MPNN,[4] Weave,[3] GraphConv, kernel ridge regression, gradient boosting,[50] random forest,[25] logistic regression, directed acyclic graph models,[51] support vector machines, Deep Tensor Neural Networks,[10] multitask networks,[52] bypass networks,[53] influence relevance voting,[54] and/or ANI-1,[55] depending on the dataset. Full details can be found in Wu et al.. For the feed-forward network model from Mayr et al., we modified the authors' original code with their guidance in order to run their code on all of the datasets, not just on ChEMBL. We tuned learning rates and hidden dimensions in addition to the extensive hyperparameter search already present in their code.

Results and Discussion

In the following sections, we analyze the performance of our model on both public and proprietary datasets. Specifically, we aim to answer the following questions:

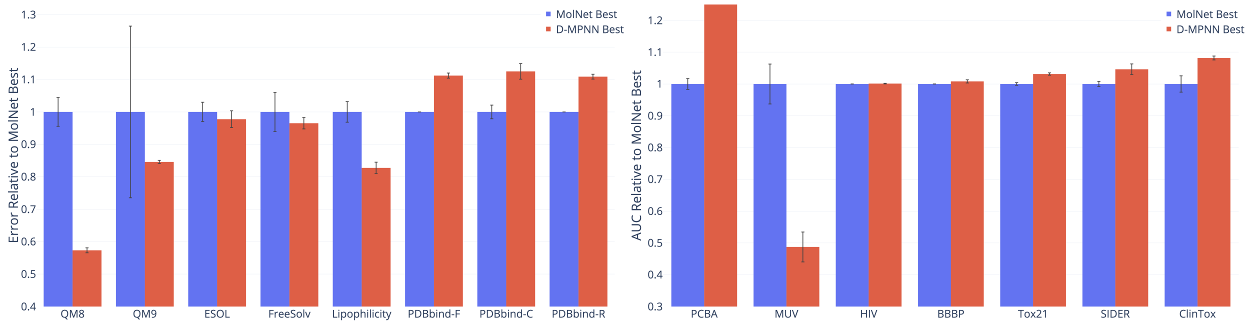
1. How does our model perform on both public and proprietary datasets compared to public benchmarks, and how close are we to experimental reproducibility?
2. How should we be splitting our data, and how does the method of splitting affect our evaluation of the model's generalization performance?
3. What are the key elements of our model, and how can we maximize performance?

In the following sections, all results are displayed as plots showing change relative to a baseline model rather than showing absolute performance numbers. This is because different datasets use different metrics and the scale of the performance numbers can differ drastically between datasets. For the regression datasets, the plots show error (either RMSE or MAE) relative to the baseline model, meaning lower is better, while for the classification datasets, the plots show AUC (either ROC-AUC or PRC-AUC) relative to the baseline model, meaning higher is better. Table 3 indicates the metric for each dataset. Tables showing the exact performance numbers for all experiments can be found in the Supplemental Information. In addition, error bars show the standard deviations of the means across runs, calculated by dividing the standard deviation of individual runs by the square root of the number of runs.

Comparison to Baselines

After optimizing our model, we compare our best single model on each dataset against models from prior work.

Comparison to MoleculeNet



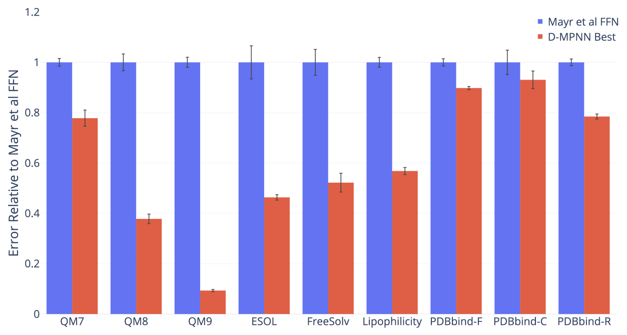
(a) Regression Datasets. (b) Classification Datasets.

Figure 3: Comparison of our best single model (i.e. optimized hyperparameters and optionally RDKit features but without ensembling) to the best models from Wu et al..

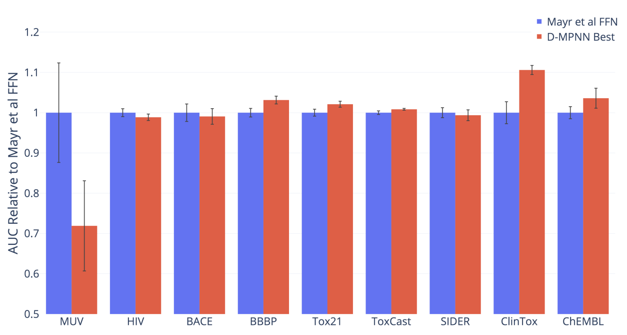
We first compare our D-MPNN to the best model from MoleculeNet[2,56] on the same datasets on which Wu et al. evaluate their models. We were unable to reproduce their original data splits on BACE, Toxcast, and QM7, but we have evaluated our model against their original splits on all of the other datasets. The splits are a mix of random, scaffold, and time splits, as indicated in Figure 3.

We observe that D-MPNN performs significantly better than the best model for each dataset in MoleculeNet, with two exceptions. The first is the MUV dataset, which is large but extremely imbalanced; only 0.2 percent of samples are labeled as positives. Wu et al. also encountered great difficulty with this extreme class imbalance when experimenting with the MUV dataset; all other datasets we experiment on contain at least 1% positives (see the Supplemental Information for full class balance information). The second exception is the three variants of the PDBbind dataset, where there is auxiliary 3D information available. The current iteration of our D-MPNN does not use 3D coordinate information, and we leave this extension to future work. Thus it is unsurprising that our D-MPNN model underperforms models using 3D information on a protein binding affinity prediction task such as PDBbind, where 3D structure is key. Nevertheless, our D-MPNN model outperforms the best graph-based method in MoleculeNet on PDBbind. Moreover, we note that on other datasets that provide 3D coordinate information (QM8 and QM9), our model outperforms the best model in MoleculeNet with or without 3D coordinates.

Comparison to Mayr et al.



(a) Regression Datasets.



(b) Classification Datasets.

Figure 4: Comparison of our best single model (i.e. optimized hyperparameters and optionally RDKit features) to the model from Mayr et al.. Note that PCBA is omitted as we found Mayr et al.'s model to be numerically unstable on this dataset.

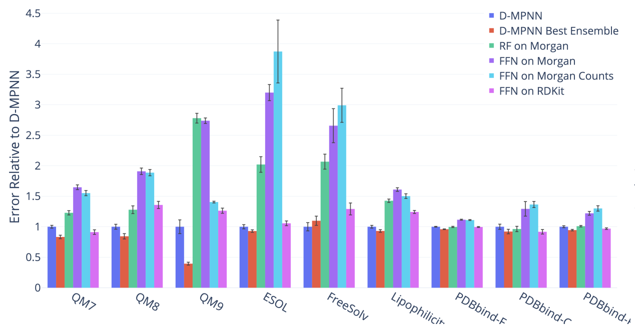
In addition, we compare D-MPNN to the baseline in Mayr et al. in Figure 4. We reproduced the features from their best model on each dataset using their scripts or equivalent packages. We then ran their code and hyperparameter optimization directly on the classification datasets, and modified their code to run on regression datasets with the authors' guidance. On most classification datasets, we match or outperform the baseline in Mayr et al., using either no human-engineered features (D-MPNN) or a very small number of human features (D-MPNN Features). On regression datasets, the baseline from Mayr et al. performs poorly in comparison, despite extensive tuning. We hypothesize that this poor performance on regression in comparison to classification is the result of a large number of binary input features to the output feed-forward network; this hypothesis is supported by the similarly poor performance of our Morgan-fingerprint-FFN baseline. In addition, their method does not employ early stopping based on validation set performance, and therefore may overfit to the training data in some cases.

We note that while our D-MPNN outperforms Mayr et al.'s method on our scaffold split of the ChEMBL dataset, the reverse is true on Mayr et al.'s original splits, which are also scaffold-based but using a different methodology. Interestingly, while our D-MPNN achieves a lower AUC on Mayr et al.'s original splits than on our own splits, Mayr et al.'s model achieves a lower AUC on our scaffold splits than on their original splits. In this paper, we will argue for the use of our scaffold splits, as they are empirically justified by a comparison to chronological splits on real-world industry datasets in Figure 9.

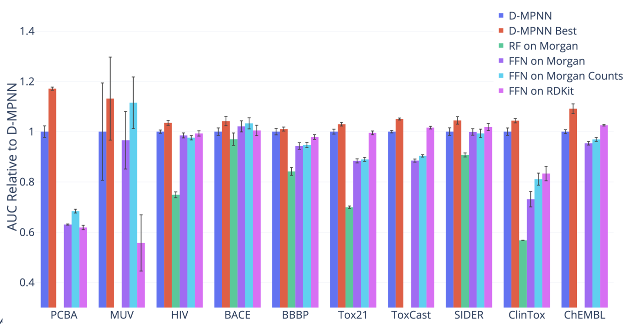
Out-of-the-Box Comparison of D-MPNN to Other Baselines

For our final baseline comparison, we evaluate our model's performance "out-of-the-box," i.e. using all the default settings (hidden size = 300, depth = 3, number of feed-forward layers = 2, dropout = 0) without any hyperparameter optimization and without any additional features. For this comparison, we compare to a number of simple baseline models that use computed fingerprints or descriptors:

1. Random forest (RF) with 500 trees run on Morgan (ECFP) fingerprints using radius 2 and hashing to a bit vector of size 2048.



(a) Regression Datasets.



(b) Classification Datasets.

Figure 5: Comparison of our unoptimized D-MPNN against several baseline models. We omitted the random forest baseline on PCBA, MUV, Toxcast, and ChEMBL due to its large computational cost.

2. Feed-forward neural network (FFN) on Morgan fingerprints.

3. FFN on Morgan fingerprints which use substructure counts instead of bits.

4. FFN on RDKit features.

The parameters of the simple baseline models are also out-of-the-box defaults. We make this comparison in order to demonstrate the strong out-of-the-box performance of our model across a wide variety of datasets. Finally, we include the performance of the automatically optimized version of our model as a reference.

Figure 5 also shows that even without optimization, our D-MPNN is provides an excellent starting point on a wide variety of datasets and targets, though it can be improved further with proper optimization.

Proprietary Datasets

We now run our model on several private industry datasets, verifying that our model's strong performance translates to these real-world datasets.

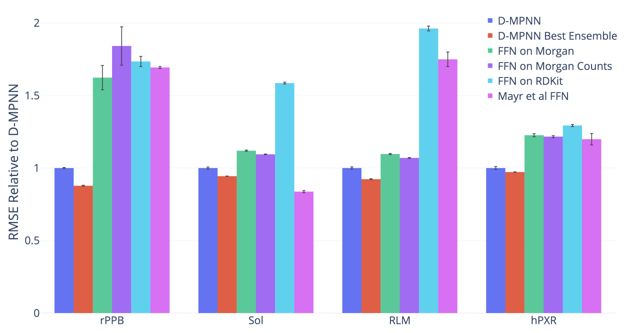
Amgen

We ran our model along with Mayr et al.'s model and our simple baselines on four internal Amgen regression datasets. In addition, we binarized the hPXR dataset according to Amgen's recommendations in order to evaluate on a classification dataset. Details of the datasets are shown in Table 4.

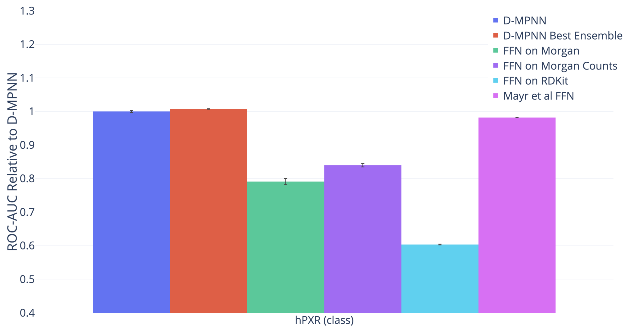
Table 4: Details on internal Amgen datasets.

Category	Dataset	# Tasks	Task Type	# Compounds	Metric
Physical Chemistry	rPPB	1	Regression	1441	RMSE
Physical Chemistry	Sol	3	Regression	18007	RMSE
Physical Chemistry	RLM	1	Regression	64862	RMSE
Physical Chemistry	hPXR	2	Regression	22188	RMSE
Physical Chemistry	hPXR_class	2	Classification	22188	ROC-AUC

For each dataset, we evaluate on a chronological split. Our model significantly outperforms the baselines in almost all cases, as shown in Figure 6. Thus our D-MPNN's strong performance on scaffold splits of public datasets can translate well to chronological splits of private industry datasets.



(a) Regression Datasets.



(b) Classification Datasets.

Figure 6: Comparison of our D-MPNN against baseline models on Amgen internal datasets.

BASF

We ran our model on 10 highly related quantum mechanical datasets from BASF. Each dataset contains 13 properties calculated on the same 30733 molecules, varying the solvent in each dataset. Dataset details are in Table 5.

Table 5: Details on internal BASF datasets.

Category	Dataset	Tasks	Task Type	# Compounds	Metric
Quantum Mechanics	benzene	13	regression	30733	R2
Quantum Mechanics	cyclohexan	13	regression	30733	R2
Quantum Mechanics	dichlormethan	13	regression	30733	R2
Quantum Mechanics	dmso	13	regression	30733	R2
Quantum Mechanics	ethanol	13	regression	30733	R2
Quantum Mechanics	ethyacetat	13	regression	30733	R2
Quantum Mechanics	h2o	13	regression	30733	R2
Quantum Mechanics	octanol	13	regression	30733	R2
Quantum Mechanics	thf	13	regression	30733	R2
Quantum Mechanics	toluol	13	regression	30733	R2

For these datasets, we used a scaffold-based split because a chronological split was unavailable. We found that the model of Mayr et al.[12] is numerically unstable on these datasets, and we therefore omit it from the comparison below. Our D-MPNN models significantly outperform our other baselines, as shown in Figure 7.

TODO This chart isn't the final version of the chart (we need to reformat it). But information-wise, we only plan to show this information + the performance of our optimized models.

Novartis

TODO: waiting on Novartis results

Experimental Error

As a final "oracle" baseline, we compare our model's performance with that of an upper bound: the agreement between multiple runs of the same assay, which we refer to as the *experimental error*. Figure 8 shows the R^2 of our model on the private Amgen regression datasets, together with the performance of Amgen's internal model which uses expert-crafted descriptors; in addition, this graph shows the experimental error. Although our D-MPNN

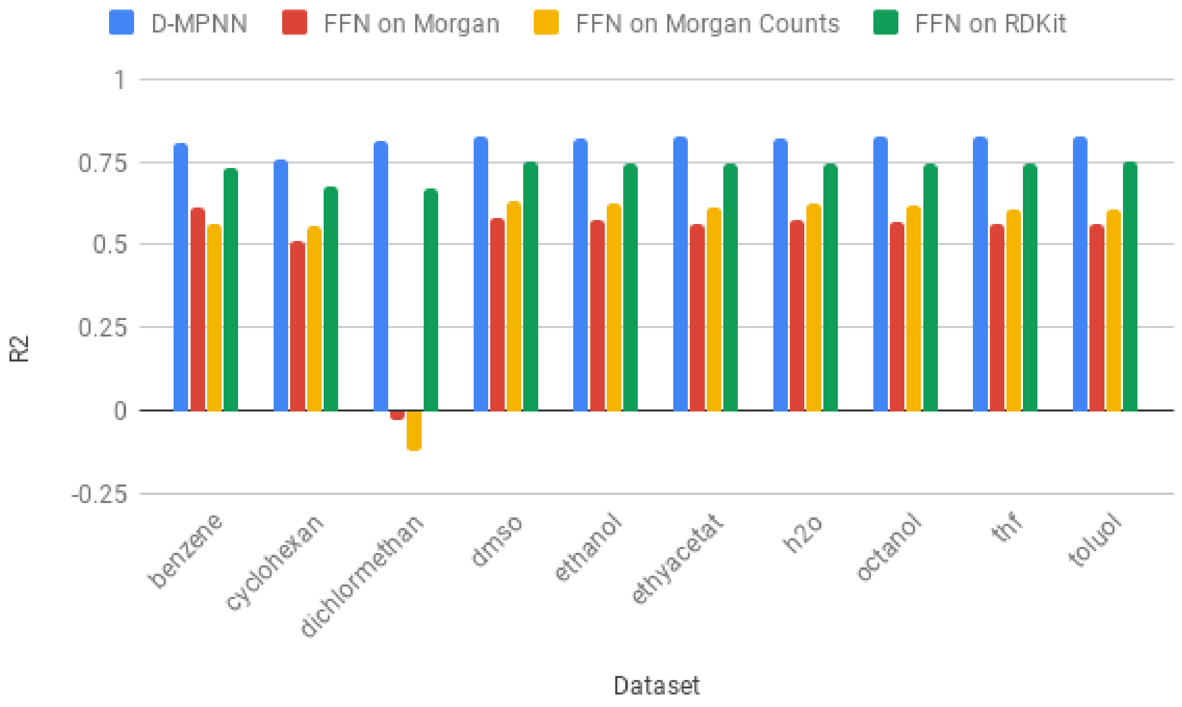


Figure 7: Comparison of our D-MPNN against baseline models on BASF internal datasets.

outperforms the Amgen internal model on all but the smallest rPPB dataset, both models remain far less accurate than the corresponding ground truth assays. Thus there remains a significant space for further performance improvement in the future.

Analysis of Split Type

We now justify our use of scaffold splits for performance evaluation.

The ultimate goal of building a property prediction model is to predict properties on new chemistry in order to aid the search for drugs from new classes of molecules. On proprietary company datasets, performance on new chemistry is evaluated using a chronological split of the data, i.e., everything before a certain date serves as the training set while everything after that date serves as the test set, thereby approximating model performance on molecules that chemists are likely to investigate in the future. Since chronological data typically isn't available for public datasets, we investigate whether we can use our scaffold split as a reasonable

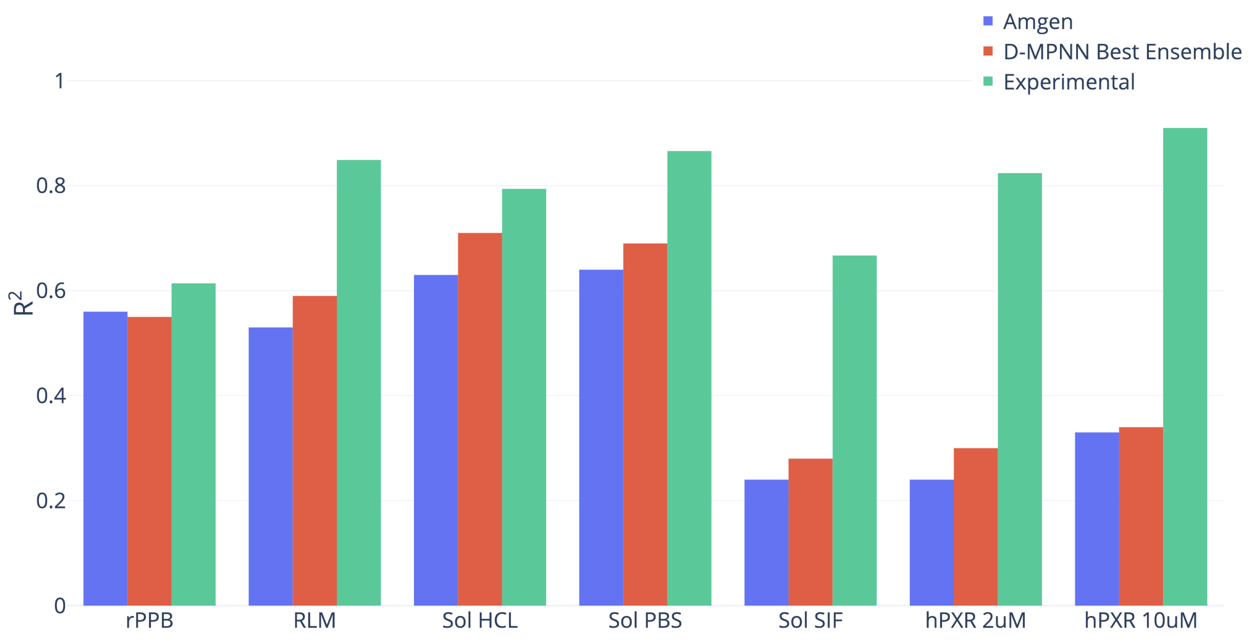


Figure 8: Comparison of Amgen internal model and our D-MPNN, using a chronological split, to experimental error. The experimental error is not evaluated on the exact same split, but the difference in performance is striking.

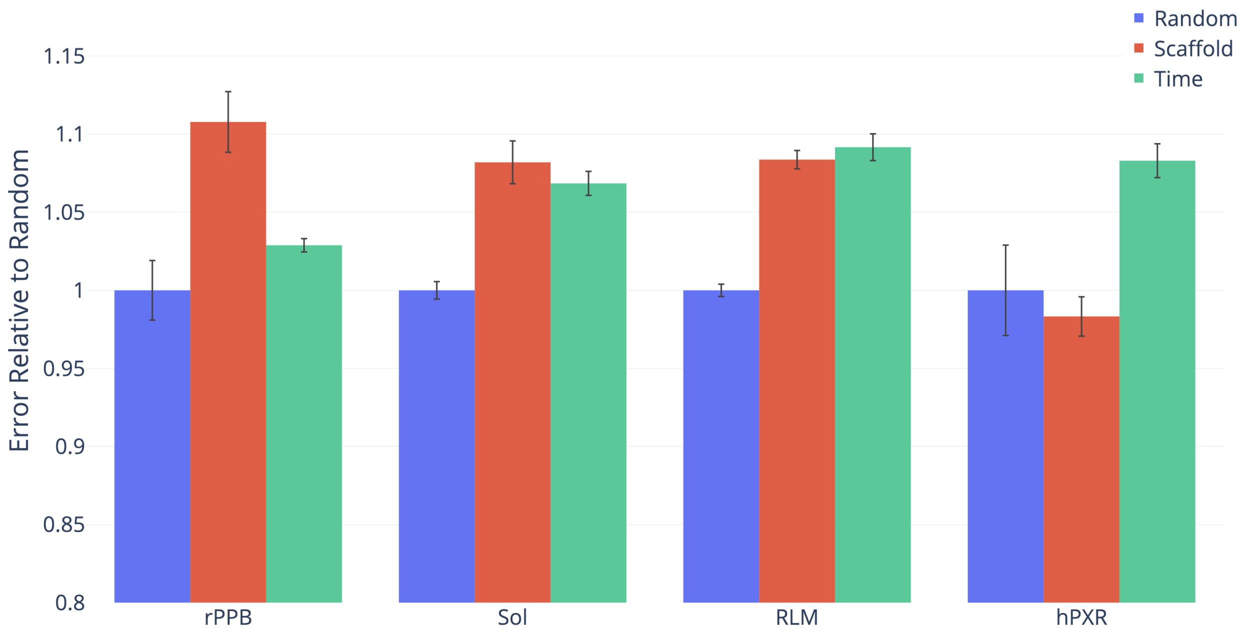
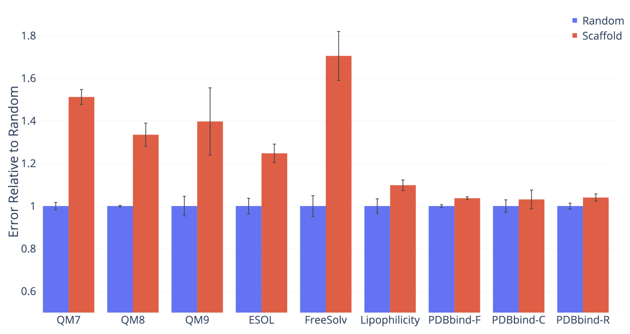
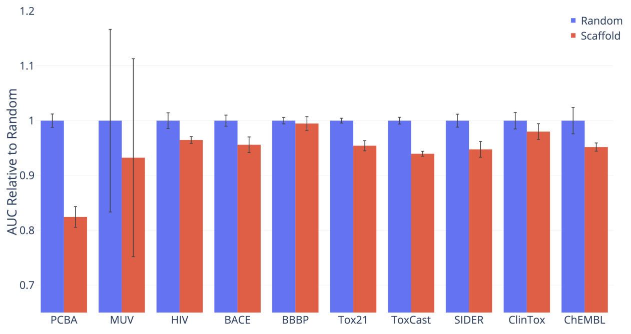


Figure 9: Performance on four Amgen regression datasets according to three methods of splitting the data.



(a) Regression Datasets.



(b) Classification Datasets.

Figure 10: Comparison of two methods of splitting the data.

proxy for a chronological split.

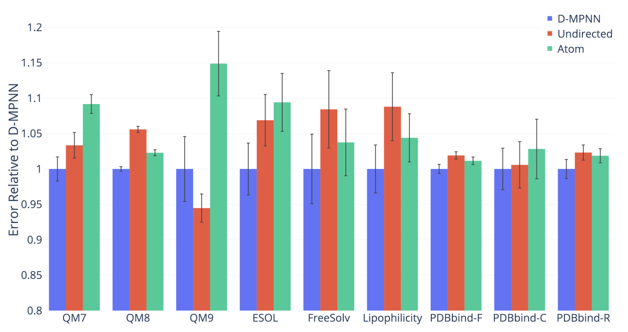
As illustrated in Figure 9, our scaffold split is able to roughly approximate a chronological split on four datasets from Amgen. Figure 10 shows the difference between a random split and a scaffold split on the publicly available datasets, further demonstrating that a scaffold split results in a more difficult, and ideally more useful, measure of performance. Therefore, all subsequent results are reported on a scaffold split in order to better reflect the generalization ability of our model on new chemistry.

Ablations

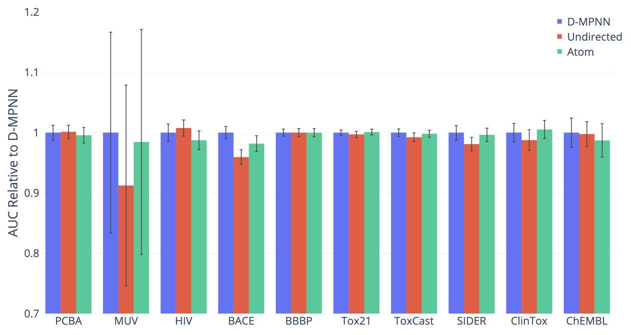
Lastly, we analyze and justify our modeling choices and optimizations.

Message Type

The most important distinction between our D-MPNN and related work is the nature of the messages being passed across the molecule. Most prior work use messages centered on atoms whereas our D-MPNN model uses messages centered on directed bonds. To isolate the effect of the message passing paradigm on property prediction performance, we implemented message passing on undirected bonds and on atoms as well, as detailed in the Supplemental Information and in our code. Figure 11 illustrates the differences in performance between these three types of message passing. While performance is similar on most classification



(a) Regression Datasets.

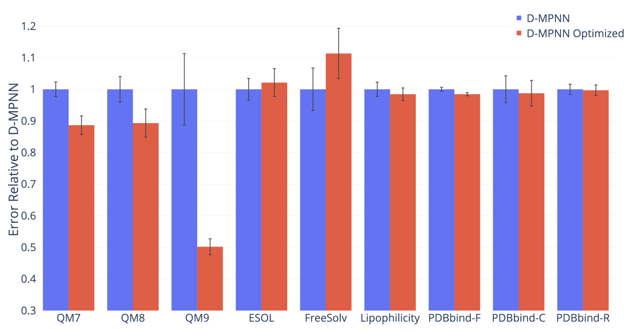


(b) Classification Datasets.

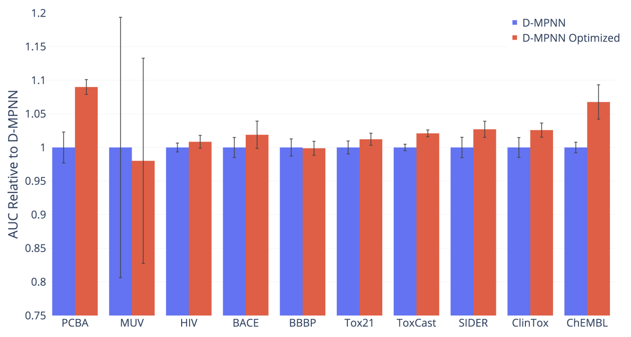
Figure 11: Comparison of different message passing paradigms.

datasets, messages centered on directed bonds outperform messages centered on undirected bonds and on atoms on several of the regression datasets, indicating the benefit of D-MPNN's directed bond approach.

Hyperparameter Optimization



(a) Regression Datasets.

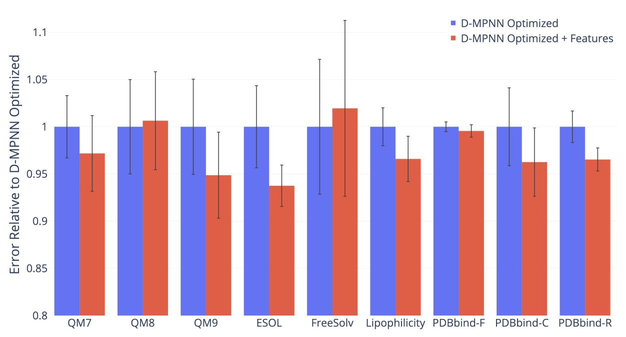


(b) Classification Datasets.

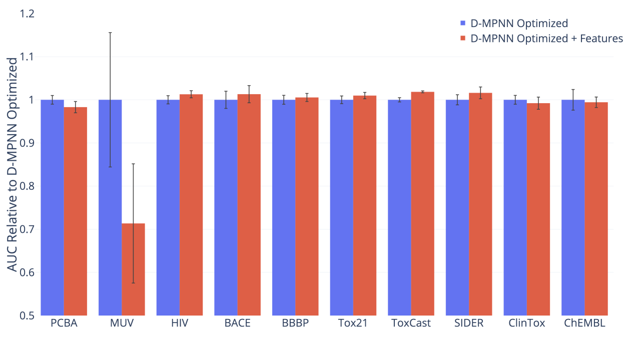
Figure 12: Effect of performing Bayesian hyperparameter optimization on the depth, hidden size, number of fully connect layers, and dropout.

To improve model performance, we performed Bayesian Optimization to select the best model hyperparameters for each dataset. Figure 12 illustrates the benefit of performing this optimization, as model performance improves on virtually every dataset. Interestingly, some datasets are particularly sensitive to hyperparameters. While most datasets experience a moderate 2-5% improvement in performance with hyperparameter optimization, the quantum mechanics datasets (QM7, QM8, and QM9) and PCBA see dramatic improvements in performance, with QM9 performing 37% better after optimization.

RDKit Features



(a) Regreession Datasets.



(b) Classification Datasets.

Figure 13: Effect of adding molecule-level features generating with RDKit to our model.

After selecting the best hyperparameters for each dataset, we examined the impact of adding additional molecule-level features from RDKit to our model. Figure 13 shows the effect on model performance. The results appear to be highly dataset-dependent. Some datasets, such as QM9 and ESOL, show marked improvement with the addition of features, while other datasets, such as PCBA and HIV, actually show worse performance with the features. This is most likely because the features are particularly relevant to certain tasks while possibly confusing and distracting the model on other tasks.

Another interesting trend is the effect of adding features to the three PDBbind datasets. The features appear to help on all three datasets, but the benefit is much more pronounced on the extremely small core dataset than it is on the larger refined and full datasets. This indicates that the features may help compensate for the lack of training data and thus may be particularly relevant in low-data regimes. Thus, it is worthwhile to consider the addition of features both when they are particularly relevant to the task of interest and when the dataset is especially small.

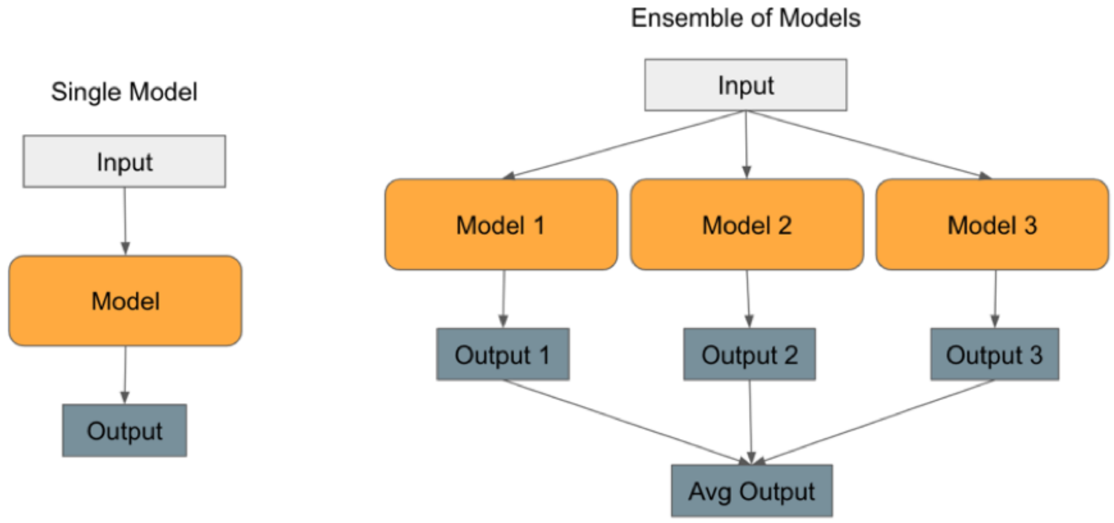
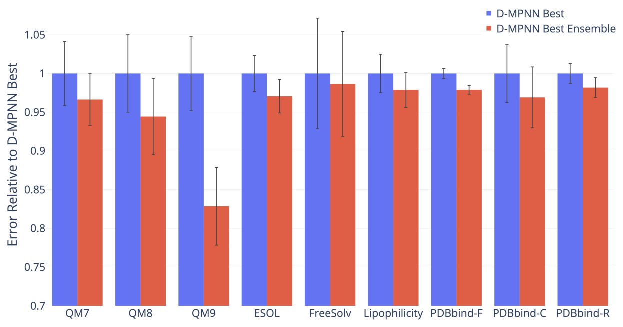
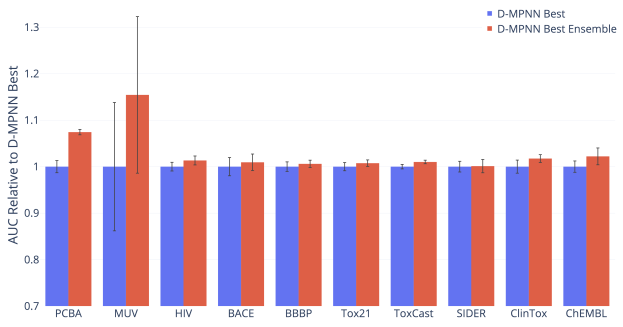


Figure 14: An illustration of ensembling models. On the left is a single model, which takes input and makes a prediction. On the right is an ensemble of models. Each model takes the same input and makes a prediction independently, and then the predictions are averaged to generate the ensemble's prediction.



(a) Regreession Datasets.



(b) Classification Datasets.

Figure 15: Effect of using an ensemble of five models instead of a single model.

Ensembling

To maximize performance, we next trained an ensemble of models. For each dataset, we selected the best single model – i.e. the best hyperparameters along with the RDKit features if the features improved performance – and we trained five models instead of one on each data split. The results appear in Figure 15. On most datasets, ensembling only provides a small benefit, but as with hyperparameter optimization, there are certain datasets, particularly the quantum mechanics datasets, which especially benefit from the effect of ensembling.

Effect of Data Size

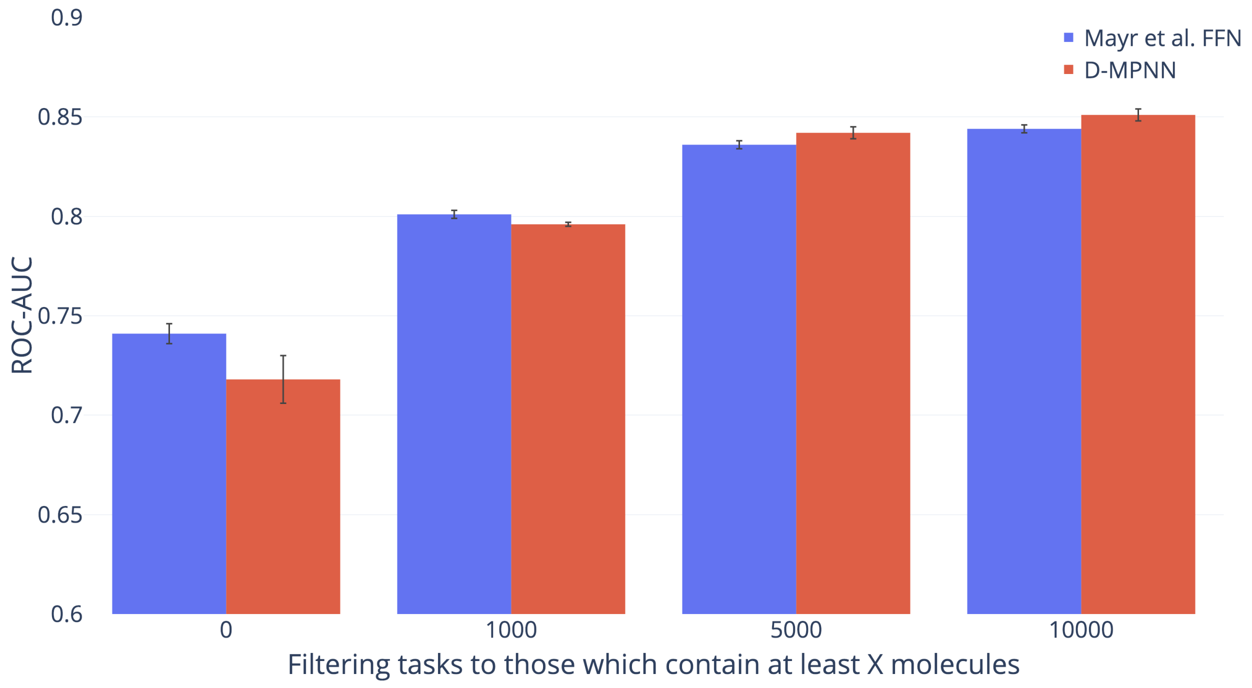


Figure 16: Effect of data size on the performance of the model from Mayr et al. and of our D-MPNN model.

Finally, we analyze the effect of data size on the performance of our model, using the ChEMBL dataset. ChEMBL is a large dataset of 456331 molecules on 1310 targets, but is extremely sparse: only half of the 1310 targets have even 300 labels. For this analysis, we use the original scaffold-based split of Mayr et al., containing 3 cross-validation folds. From Figure 16, it is apparent that our D-MPNN struggles on low-label targets in comparison to this baseline. As our D-MPNN model does not use any human-engineered fingerprints and must therefore learn its features completely from scratch based on the input data, it is unsurprising that the average ROC-AUC score of D-MPNN is worse than that of the feed-forward network running on human-engineered descriptors in Mayr et al.. When we filter the ChEMBL dataset by pruning low-data targets at different thresholds, we find that our D-MPNN indeed outperforms the best model of Mayr et al. at larger data thresholds even on Mayr et al.'s splits; see Figure 16.

Conclusion and Future Work

In this paper, we have argued for the use of scaffold-based data splits as a realistic approximation to the difficulty of the chronological data splits from real-world drug discovery settings. In addition, we have presented a novel bond-based message passing neural network architecture for molecular property prediction, and we have shown that our proposed model significantly outperforms state-of-the-art baselines on scaffold-based data splits.

Several important avenues of future research remain, which hold the potential to significantly improve our model's efficacy and utility. The first is the incorporation of additional 3D information into our model, which currently supports only a very restricted and naive inclusion of such features. On datasets where 3D structural information is available, our model sometimes underperforms previous baselines when those baselines are allowed access to 3D features. The second avenue of improvement is a principled pretraining approach, which some authors have already begun to explore.[57,58] Such an approach could enable a pretrained version of our model to perform more effectively on datasets of just one thousand molecules or even smaller. Finally, this increased understanding of how estimation of model generalizability is affected by split type opens the door to future work in uncertainty quantification and domain of applicability assessment.

Acknowledgement

TODO

The authors thank the Machine Learning for Pharmaceutical Discovery and Synthesis consortium for funding this work.

Supporting Information Available

In our Supplementary Information, we provide further detailed analysis of model comparisons on scaffold as well as random splits, and provide the full tables of performance numbers. In addition, we analyze class balance of classification datasets and provide a list of the RDKit calculated features used by our model.

References

(1) Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems. 2015; pp 2224–2232.

(2) Wu, Z.; Ramsundar, B.; Feinberg, E.; Gomes, J.; Geniesse, C.; Pappu, A. S.; Leswing, K.; Pande, V. MoleculeNet: a benchmark for molecular machine learning. *Chemical Science* **2018**, *9*, 513530.

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

(4) Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; Dahl, G. E. Neural Message Passing for Quantum Chemistry. 2017.

(5) Li, Y.; Tarlow, D.; Brockschmidt, M.; Zemel, R. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493* **2015**,

(6) Kipf, T. N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* **2016**,

(7) Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in Neural Information Processing Systems. 2016; pp 3844–3852.

(8) Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* **2013**,

(9) Coley, C. W.; Barzilay, R.; Green, W. H.; Jaakkola, T. S.; Jensen, K. F. Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction. *57*, 1757–1772.

(10) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. Quantum-chemical insights from deep tensor neural networks. *Nature communications* **2017**, *8*, 13890.

(11) Battaglia, P.; Pascanu, R.; Lai, M.; Rezende, D. J. Interaction networks for learning about objects, relations and physics. Advances in neural information processing systems. 2016; pp 4502–4510.

(12) Mayr, A.; Klambauer, G.; Unterthiner, T.; Steijaert, M.; Wegner, J. K.; Ceulemans, H.; Clevert, D.-A.; Hochreiter, S. Large-scale comparison of machine learning methods for drug target prediction on ChEMBL. *Chemical Science* **2018**, *9*, 5441–5451.

(13) Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; de Freitas, N. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE* **2016**, *104*, 148–175.

(14) Dietterich, T. G. Ensemble methods in machine learning. International workshop on multiple classifier systems. 2000; pp 1–15.

(15) Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. Advances in Neural Information Processing Systems. 2017; pp 1024–1034.

(16) Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks* **2009**, *20*, 61–80.

(17) Henaff, M.; Bruna, J.; LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* **2015**,

(18) Dai, H.; Dai, B.; Song, L. Discriminative embeddings of latent variable models for structured data. International Conference on Machine Learning. 2016; pp 2702–2711.

(19) Lei, T.; Jin, W.; Barzilay, R.; Jaakkola, T. Deriving neural architectures from sequence and graph kernels. *arXiv preprint arXiv:1705.09037* **2017**,

(20) Kusner, M. J.; Paige, B.; Hernández-Lobato, J. M. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925* **2017**,

(21) Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science* **2018**, *4*, 268–276.

(22) Jin, W.; Barzilay, R.; Jaakkola, T. Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv preprint arXiv:1802.04364* **2018**,

(23) Jin, W.; Yang, K.; Barzilay, R.; Jaakkola, T. Learning Multimodal Graph-to-Graph Translation for Molecular Optimization. *arXiv preprint arXiv:1812.01070* **2018**,

(24) Cortes, C.; Vapnik, V. Support vector machine. *Machine learning* **1995**, *20*, 273–297.

(25) Breiman, L. Random forests. *Machine learning* **2001**, *45*, 5–32.

(26) Mauri, A.; Consonni, V.; Pavan, M.; Todeschini, R. Dragon software: An easy approach to molecular descriptor calculations. *Match* **2006**, *56*, 237–248.

(27) Rogers, D.; Hahn, M. Extended-connectivity fingerprints. *Journal of chemical information and modeling* **2010**, *50*, 742–754.

(28) Swamidass, S. J.; Chen, J.; Bruand, J.; Phung, P.; Ralaivola, L.; Baldi, P. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics* **2005**, *21*, i359–i368.

(29) Cao, D.-S.; Xu, Q.-S.; Hu, Q.-N.; Liang, Y.-Z. ChemoPy: freely available python package for computational biology and chemoinformatics. *Bioinformatics* **2013**, *29*, 1092–1094.

(30) Durant, J. L.; Leland, B. A.; Henry, D. R.; Nourse, J. G. Reoptimization of MDL keys for use in drug discovery. *Journal of chemical information and computer sciences* **2002**, *42*, 1273–1280.

(31) Moriwaki, H.; Tian, Y.-S.; Kawashita, N.; Takagi, T. Mordred: a molecular descriptor calculator. *Journal of cheminformatics* **2018**, *10*, 4.

(32) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences* **1988**, *28*, 31–36.

(33) Schütt, K.; Kindermans, P.-J.; Felix, H. E. S.; Chmiela, S.; Tkatchenko, A.; Müller, K.-R. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. Advances in Neural Information Processing Systems. 2017; pp 991–1001.

(34) Kondor, R.; Son, H. T.; Pan, H.; Anderson, B.; Trivedi, S. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144* **2018**,

(35) Faber, F. A.; Hutchison, L.; Huang, B.; Gilmer, J.; Schoenholz, S. S.; Dahl, G. E.; Vinyals, O.; Kearnes, S.; Riley, P. F.; von Lilienfeld, O. A. Machine learning prediction errors better than DFT accuracy. *arXiv preprint arXiv:1702.05532* **2017**,

(36) Feinberg, E. N.; Sur, D.; Wu, Z.; Husic, B. E.; Mai, H.; Li, Y.; Sun, S.; Yang, J.; Ramsundar, B.; Pande, V. S. PotentialNet for Molecular Property Prediction. *ACS central science* **2018**, *4*, 1520–1530.

(37) Lee, A. A.; Yang, Q.; Bassyouni, A.; Butler, C. R.; Hou, X.; Jenkinson, S.; Price, D. A. Ligand biological activity predicted by cleaning positive and negative chemical correlations. *Proceedings of the National Academy of Sciences* **2019**,

(38) Liu, K.; Sun, X.; Jia, L.; Ma, J.; Xing, H.; Wu, J.; Gao, H.; Sun, Y.; Boulnois, F.; Fan, J. Chemi-net: a graph convolutional network for accurate drug property prediction. *arXiv preprint arXiv:1803.06236* **2018**,

(39) Ishiguro, K.; Maeda, S.-i.; Koyama, M. Graph Warp Module: an Auxiliary Module for Boosting the Power of Graph Neural Networks. *arXiv preprint arXiv:1902.01020* **2019**,

(40) Mahé, P.; Ueda, N.; Akutsu, T.; Perret, J.-L.; Vert, J.-P. Extensions of marginalized graph kernels. Proceedings of the twenty-first international conference on Machine learning. 2004; p 70.

(41) Koller, D.; Friedman, N.; Bach, F. *Probabilistic graphical models: principles and techniques*; MIT press, 2009.

(42) Nair, V.; Hinton, G. E. Rectified Linear Units Improve Restricted Boltzmann Machines. *Proceedings of the 27th International Conference on Machine Learning* **2010**,

(43) Landrum, G. RDKit: Open-source cheminformatics. 2006.

(44) Paszke, A.; Gross, S.; Chintala, S.; Chanan, G. PyTorch. 2017.

(45) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-learn: Machine learning in Python. *Journal of machine learning research* **2011**, *12*, 2825–2830.

(46) Irwin, J. J.; Shoichet, B. K. ZINC- A free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling* **2005**, *45*, 177–182.

(47) Kim, S.; Thiessen, P. A.; Bolton, E. E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B. A. PubChem substance and compound databases. *Nucleic acids research* **2015**, *44*, D1202–D1213.

(48) Sartor, M. A.; Leikauf, G. D.; Medvedovic, M. LRpath: a logistic regression approach for identifying enriched biological groups in gene expression data. *Bioinformatics* **2008**, *25*, 211–217.

(49) `https://github.com/hyperopt/hyperopt`.

(50) Friedman, J. H. Greedy function approximation: a gradient boosting machine. *Annals of statistics* **2001**, 1189–1232.

(51) Lusci, A.; Pollastri, G.; Baldi, P. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of chemical information and modeling* **2013**, *53*, 1563–1575.

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

(53) Ramsundar, B.; Liu, B.; Wu, Z.; Verras, A.; Tudor, M.; Sheridan, R. P.; Pande, V. Is multitask deep learning practical for pharma? *Journal of chemical information and modeling* **2017**, *57*, 2068–2076.

(54) Swamidass, S. J.; Azencott, C.-A.; Lin, T.-W.; Gramajo, H.; Tsai, S.-C.; Baldi, P. Influence relevance voting: an accurate and interpretable virtual high throughput screening method. *Journal of chemical information and modeling* **2009**, *49*, 756–766.

(55) Smith, J. S.; Isayev, O.; Roitberg, A. E. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical science* **2017**, *8*, 3192–3203.

(56) Ramsundar, B.; Eastman, P.; Leswing, K.; Walters, P.; Pande, V. *Deep Learning for the Life Sciences*; O'Reilly Media, 2019; `https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837`.

(57) Navarin, N.; Tran, D. V.; Sperduti, A. Pre-training Graph Neural Networks with Kernels. *arXiv preprint arXiv:1811.06930* **2018**,

(58) Goh, G. B.; Siegel, C.; Vishnu, A.; Hodas, N. Using Rule-Based Labels for Weak Supervised Learning: A ChemNet for Transferable Chemical Property Prediction. Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018; pp 302–310.

Graphical TOC Entry

Some journals require a graphical entry for the Table of Contents. This should be laid out "print ready" so that the sizing of the text is correct. Inside the `tocentry` environment, the font used is Helvetica 8 pt, as required by *Journal of the American Chemical Society*.
The surrounding frame is 9 cm by 3.5 cm, which is the maximum permitted for *Journal of the American Chemical Society* graphical table of content entries. The box will not resize if the content is too big: instead it will overflow the edge of the box.
This box and the associated title will always be printed on a separate page at the end of the document.