Characterizing Uncertainty in Machine Learning for Chemistry

Esther Heid,1,2,∗ Charles J. McGill,1,3,∗ Florence H. Vermeire,1,4 and William H. Green1,†

1 Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, United States
2 Institute of Technology, TU Wien, 1060 Vienna, Austria
3 Department of Chemical and Life Science Engineering, Virginia Commonwealth University, Richmond, Virginia 23284, United States
4 Department of Chemical Engineering, KU Leuven, Celestijnenlaan 200F, B-3001 Leuven, Belgium

Characterizing uncertainty in machine learning models has recently gained interest in the context of machine learning reliability, robustness, safety, and active learning. Here, we separate the total uncertainty into contributions from noise in the data (aleatoric) and shortcomings of the model (epistemic), further dividing epistemic uncertainty into model bias and variance contributions. We systematically address the influence of noise, model bias, and model variance in the context of chemical property predictions, where the diverse nature of target properties and the vast chemical space give rise to many different distinct sources of prediction error. We demonstrate that different sources of error can each be significant in different contexts and must be individually addressed during model development. Through controlled experiments on datasets of molecular properties, we show important trends in model performance associated with the level of noise in the dataset, size of the dataset, model architecture, molecule representation, ensemble size, and dataset splitting. In particular, we show that 1) noise in the test set can limit a model’s observed performance when the actual performance is much better, 2) using size-extensive model aggregation structures is crucial for extensive property prediction, 3) ensembling is a reliable tool for uncertainty quantification and improvement specifically for the contribution of model variance, and 4) evaluations of cross-validation models understate their performance. We develop general guidelines on how to improve an underperforming model when falling into different uncertainty contexts.

I. INTRODUCTION

Machine learning models for chemical applications such as predicting molecular and reaction properties are becoming not only increasingly popular, but also increasingly accurate, for example for quantum-mechanical properties,1–3 biological effects,4–6 physico-chemical properties,7–11 reaction yields,12–14 or reaction rates and barriers.15–19 Also, promising developments in the field of retrosynthesis20–24 and forward reaction prediction,25–28 have been made.

However, despite the increase in accuracy, many machine learning models fail in real-world applications.29,30 This can be due to a lack of generalization, lack of ability to filter out erroneous predictions for edge cases, or because the employed training and test sets are simply not reflective of the application of interest, so that the developed model is suboptimal for the proposed task. Poor choice of test set can overestimate, or more commonly, underestimate the actual errors that a user will encounter when the model is applied. Optimizing a mediocre model can be tedious, time-consuming, and often unfruitful. Moreover, the model architectures, input representations, and dataset characteristics for chemical applications differ considerably from other fields of research, so that following general guidelines for optimizing machine learning models often fails to produce accurate models for molecular and reaction properties. To optimize a model in a targeted and efficient manner, it is imperative to understand and identify possible sources of error and uncertainty in a model.

The separation of the total uncertainty into aleatoric (data-dependent, noise-induced, irreducible) and epistemic (model-dependent, reducible) contributions has recently received increasing attention.32–34 The aleatoric uncertainty is often referred to as the irreducible component of uncertainty that cannot be overcome by improvements to the model. Reduction in aleatoric uncertainty can instead come from improvements in the data itself, such as adding repeat measurements or removing erroneous entries. In contrast, epistemic uncertainty characterizes the reducible uncertainty caused by missing knowledge and can be decreased as the model is improved.35 The epistemic uncertainty can further be split into uncertainty arising from the choice of model (architecture, representation, and featurization) and the ambiguity of parameter optimization once a model is chosen.35 In this work, we follow the convention36,37 of calling the former model bias and the latter variance, but different other names are sometimes used in the literature, such as model uncertainty and approximation uncertainty.35 The difference between reducible and irreducible uncertainty can become blurred in these considerations, especially for different model architectures, different representations, and different data and test sets.32,35 Small dataset sizes contribute to both bias and variance components of epistemic uncertainty because they cause some ambiguity in the optimal model parameters due to sparsity in some regions, but also hinder the model convergence to a meaningful minimum generally. The size or nature of the data may additionally influence the choice of model architecture or machine learning method, providing a further avenue by which aspects of the data can
feed into epistemic uncertainty.

Many approaches toward characterizing the uncertainty of a prediction exist, such as mean-variance estimation, Bayesian approaches, evidential learning, and conformal predictions. Most approaches tackle aleatoric uncertainty, as well as those parts of the epistemic uncertainty that are associated with the ambiguity of the model parameters. However, uncertainty from model bias is usually omitted. Even when the aleatoric error is low and plenty of data is available for training, model bias can still prove to be significant. Model bias can have many forms and causes, among them limited flexibility of the model, limited data coverage, incomplete feature representation of the input data, poor training convergence to an appropriate model, and poor generalizability of training to the test set or to actual applications. We discuss how, especially in chemical systems, uncertainty from model bias can be a large contribution toward the error in a model’s prediction.

Despite the many works on characterizing uncertainty, little advice exists on how to optimize a sub-optimal model once the sources of uncertainty are known. Furthermore, the circumstances under which the epistemic uncertainty modeled by ensembling is actually indicative of the true error are not well researched yet, despite its popularity. We therefore studied the performance of selected deep learning models on chemical prediction tasks where we systematically vary noise in the input data, the number of datapoints, the chosen model architecture, molecular representation, and the number of models in an ensemble. To this aim, we rely not only on literature datasets, but also construct a new, noise-free, chemically meaningful dataset. In the discussion, we then put forward general guidelines for how to detect and circumvent model errors caused by noise, bias, and variance.

We pay particular attention to predictions of physico-chemical targets, since we find some of the sources of uncertainty to be specific to chemistry.

II. METHODS

A. Datasets

In this work, a synthetic dataset was constructed for molecular enthalpy at 298 K in units of kcal/mol as calculated from group additivity coefficients, based on the Benson group-increment theory. The dataset was desired to have characteristics well-suited to the analysis of errors of different types: no or extremely low inherent noise, large total size, a property function that would not be trivially solved by the model (unlike molar masses), and a property function that was sufficiently described by the features available to the model. Accurate representation of experimentally observable enthalpies was not a priority during construction of the dataset. Group additivity coefficients were fitted to the enthalpies calculated for the 134 thousand molecules of the QM9 dataset using ridge regression. Group additivity were bounded by a central non-hydrogen atom and the atoms and bonds within a 1-bond radius. Only groups that were represented at least 100 times in QM9 and molecules made up entirely of those groups were included in the regression. The group additivity coefficients were rounded to the nearest thousandth kcal/mol. No non-nearest-neighbour group contributions were included for symmetry, ring strain, or other inter- or intramolecular interactions. The fitted coefficients were then applied to a larger set of molecules, the GDB11 set of over 26 million unique molecules containing up to 11 C/N/O/F atoms. Choosing only molecules made up entirely of structures included in the fitted coefficients, we obtain 7.9 million molecules. The result is a large dataset with a property function that can be exactly calculated and relies on the local graph structure of molecules. The only inherent noise is at the level of numerical precision. This dataset of artificial enthalpy values is available for public download from a Zenodo repository.

Models trained on the group additivity dataset were evaluated using a single held out test set comprising 10% of the dataset (790,681 datapoints), chosen randomly. When the number of datapoints used in training are indicated in figures, that is the combined number of datapoints in the training and validation sets, split randomly at a ratio of 80:20. When multiple submodels are combined in an ensemble for the synthetic dataset, the same data splits are used in each submodel. Ensemble submodels are differentiated by beginning training of the model from different random parameter initializations. When the number of datapoints used for training is unspecified, a consistent set of about 0.7 million datapoints is used, corresponding to 10% of the non-test data remaining in the dataset.

Furthermore, the QM9 dataset was used as a low-noise real-world dataset. We selected the enthalpy at 298 K and internal energy at 0 K and 298 K as size-extensive properties, as well as the HOMO-LUMO gap as a size-intensive property. We trained directly on the quantum chemical energies, without subtraction of the atomic reference values. A single held out test set comprising 10% of the data (13,083 datapoints) was used. The rest of the data was used for the training and validation set, where a specified number of datapoints were selected randomly and split into training and validation sets in ratios of 80:20. To compute learning curves using QM9, i.e. the model performance dependent on dataset size, differently-sized training and validation sets were drawn containing a specified number of datapoints N, while leaving the test set untouched. Submodels in an ensemble using the QM9 data share the same data splits with different initial model parameters.
Three machine learning architectures were employed within this study: i) directed message passing neural networks (d-MPNNs) as described by Yang et al.$^8$ and implemented in the Chemprop software package$^{54}$ as a class of 2-dimensional graph-convolutional neural networks using learned representations, ii) feed-forward neural networks (FFNN) on molecular fingerprints, and iii) the 3-dimensional convolutional neural network SchNet.$^{55}$ Ensembles of five models were trained for each architecture and task if not specified otherwise.

The d-MPNN model takes the molecular graph as input, and performs several steps of message passing to update atom and bond features with information from their neighborhood to yield an atomic representation. A molecular representation is then obtained by aggregating the atomic representations using an aggregation function such as summing or averaging. Subsequently, a feed-forward neural network transforms the learned molecular representation into the respective target property. Unless otherwise noted, d-MPNN models trained on the synthetic group additivity dataset use a hidden size of 1000, four steps of message passing, two feed-forward layers, scaled sum aggregation (called “norm” in Chemprop), and 200 epochs of training unless otherwise indicated. In contrast, d-MPNN models trained on QM9 differ slightly by using a hidden size of 300. In the following, if the hidden size is specified that is the hidden size in both the d-MPNN and the FFNN part. All other hyperparameters were chosen according to their default values in Chemprop.

FFNNs take a molecular fingerprint, here a Morgan fingerprint$^{56}$ as implemented in RDKit,$^{57}$ as input and transform it into the respective target property. We used FFNNs as implemented in Ref. 54, where we omitted the message-passing. Model training used a hidden size of 300, two feed-forward layers, and 200 epochs of training unless indicated otherwise.

SchNet was used as provided in Ref. 55 with default hyperparameters, and trained only on QM9 tasks. It takes as input the nuclear charges and coordinates of each atom in a molecule which are calculated using quantum chemistry and provided along with the QM9 dataset. The atomic representations of each atom are refined using continuous-filter convolutional layers, thus taking into account other atoms in the molecule based on their relative distance. The atomic representations are then utilized to compute atomic contributions to the overall target, which are subsequently averaged or summed up to the total molecular target value. Since SchNet does not directly support ensembling, models with different initialization seeds were trained manually and their predictions averaged for each datapoint in the test set.

For all architectures, the validation set was used to select the best model within 200 epochs, which was further used to evaluate the test performance.

Throughout this work, we refer to the predictions made by and errors resulting from ensembles of submodels. To explain the meaning of different ensemble metrics, we will use $\hat{y}_{i,j}(X_n)$ to denote the model prediction on the input $X_n$ (test data molecule $n$), where $i$ indicates model initialization and $j$ indicates the split configuration of the full data into training, validation and test sets. The target for each datapoint is given by $y(X_n)$. For ensembling, we obtain $N_{\text{ens}}$ models with different $i$ on the exact same data splits $j = 1$. The prediction of the ensemble, $\overline{y}(X_n)$, is given by

$$\overline{y}(X_n) = \frac{1}{N_{\text{ens}}} \sum_{i=1}^{N_{\text{ens}}} \hat{y}_{i,1}(X_n)$$

where the submodel predictions for a particular test datapoint $n$ are averaged together over the number of submodels included in the ensemble, $N_{\text{ens}}$. The reported mean absolute error of an ensemble model is

$$MAE_{\text{ens}} = \frac{\sum_{n=1}^{N_{\text{test}}} |y(X_n) - \frac{1}{N_{\text{ens}}} \sum_{i=1}^{N_{\text{ens}}} \hat{y}_{i,1}(X_n)|}{N_{\text{test}}}$$

(with an analogous expression for the root mean squared error). Here, we find the absolute error between the ensemble prediction and its corresponding target value. The overall model performance is reported as an average over all $N_{\text{test}}$ datapoints in the test set.

The standard deviation of the ensemble prediction of each point $n$ may be used to define confidence intervals and uncertainty bounds. The standard deviation used is the unbiased standard deviation of the submodel predictions for each datapoint.

$$s(X_n) = \sqrt{\frac{\sum_{i}^{N_{\text{ens}}} (\hat{y}_{i,1}(X_n) - \overline{y}(X_n))^2}{N_{\text{ens}} - 1}}$$

The standard deviation of the ensemble prediction is used to define uncertainty intervals in two different ways in this work. In the case where we use the measure directly to evaluate error magnitude (Fig. 8), we will define the confidence interval for predictions of each test datapoint $n$ indicated by the standard deviation as

$$[\overline{y}(X_n) - t \frac{s(X_n)}{\sqrt{N_{\text{ens}}}}; \overline{y}(X_n) + t \frac{s(X_n)}{\sqrt{N_{\text{ens}}}}]$$

where $t$ is the Student-t factor for the specified confidence $p$ and degrees of freedom $N_{\text{ens}} - 1$. In the case where we are using the ensemble standard deviation only as a relative indicator of total error within the dataset (Fig. 3), the uncertainty bounds will be the standard deviation as given in Eq. (3) and scaled to match the average error of the competing uncertainty method.

The ensemble mean and standard deviation $\overline{y}(X_n)$ and $s(X_n)$ can further be used to estimate the contributions
of bias and variance error to the overall observed MAE via Bayesian inference. Here, we follow the method of Ref. 58. In this approach, the different predictions made by individual models $i$ for a single test datapoint, $\hat{y}_{i,j}(X_n)$, are assumed to be normally distributed around a mean distribution value $\mu_j(X_n)$, with a spread related to the ensemble standard deviation $s(X_n)$. In accordance to the central limit theorem, an ensemble prediction $\bar{y}(X_n)$ will converge to $\mu_j(X_n)$ at very large ensemble sizes. The nonvariance contribution to error is considered to be the absolute error occurring in a theoretical very large ensemble

$$AE_{NV}(X_n) = |\mu_j(X_n) - y(X_n)|$$ \hspace{1cm} (5)

The Nonvariance error consists of bias and noise errors, and in noise-free datasets it represents only the bias error. The variance error is considered to be the difference between the total absolute error of the ensemble prediction and the nonvariance error

$$AE_V(X_n) = |\bar{y}(X_n) - y(X_n)| - E_{NV}(X_n)$$ \hspace{1cm} (6)

Bayesian inference is used to calculate the posterior distribution of $\mu_i(X_n) - y(X_n)$ for each datapoint, using the distribution of $\bar{y}(X_n) - y(X_n)$ over the dataset as an initial prior distribution, which is subsequently iteratively refined. The posterior distribution can be used to calculate expected values of the absolute error from variance and nonvariance defined in Eq. (5) and Eq. (6) for each datapoint. The contributions are then averaged across the dataset to arrive at the expected variance and nonvariance contributions to the dataset MAE.

In place or in addition to ensembling, varying data splits can be used to report a cross-validation score, i.e. the performance of models obtained on $N_{\text{splits}}$ different splits, and thus different test sets. Without ensembling ($i = 1$), the reported mean absolute error is

$$MAE_{\text{CV,reported}} = \frac{\sum_{j=1}^{N_{\text{splits}}} \sum_{n=1}^{N_{\text{test}}} |y(X_n) - \hat{y}_{1,j}(X_n)|}{N_{\text{splits}}}$$ \hspace{1cm} (7)

In a conventional approach, the same data set is split between training, validation and test sets several different ways: the first split is $j = 1$, the second split is $j = 2$, etc. The models $j = 1, 2, 3,... N_{\text{splits}}$ trained on these distinct data sets are all different, and each has a different MAE as judged using its own test set. Averaging the MAEs from all the different splits as the “reported” MAE, Eq. (7), gives a reasonable estimate of the split-independent MAE, and this can be helpful particularly for small data sets.

However, one usually wants a single model to make predictions, so it is common to view these $N_{\text{splits}}$ models as an ensemble, and average their predictions. The MAE of the prediction of the ensemble is given by Eq. (8); note that to evaluate Eq. (8) correctly one needs a held-out test set not used to train any of the $N_{\text{splits}}$ models. The MAE computed using Eq. (8) is more representative of the MAE that will be observed in new predictions made with the ensemble than Eq. (7), and as shown below it is usually smaller.

$$MAE_{\text{CV,true}} = \frac{\sum_{n=1}^{N_{\text{test}}} |y(X_n) - \frac{\sum_{j=1}^{N_{\text{splits}}} \hat{y}_{j,n}(X_n)}{N_{\text{splits}}}|}{N_{\text{test}}}$$ \hspace{1cm} (8)

D. Software and Data Availability

The Chemprop software and SchNet software used in model training are both freely available through GitHub. The constructed noise-free dataset of group additivity enthalpies is available through Zenodo.9 The QM9 dataset can be downloaded from the MoleculeNet website.9 The implementation of the Bayesian inference method for calculating nonvariance contribution is available through GitHub.60 Other scripts necessary to train the models analyzed in this work and recreate the results are provided through GitHub.61

III. RESULTS

In the following, we describe the influence of noise, bias and variance on the observed model performance, as well as possible pitfalls associated with each type of error. We often discuss the shape of the learning curve, i.e. the test set error as it depends on the size of a dataset, as different types of limitations caused by noise, bias or variance can lead to unique patterns in the learning curve. The slope of the learning curve characterizes the change in error upon addition of data and can be utilized to predict how much data is needed to achieve a specific accuracy. In general, a steep, negative slope on a log-log plot without plateaus is desirable.

A. Noise

Noise in the target data obstructs a model’s ability to learn meaningful relations between an input and a target. In general, noise can be of random, uniform nature (homoscedastic), affecting all datapoints with the same error probability distribution, or systematic (heteroscedastic), where different domains of data are affected by different error probability distribution. We discuss both options separately in the following, because they require different remedies. In our demonstration of random noise, we also show that noise has distinct effect behaviors when it is present in the training set versus the test set, with the effects in the training set actually leading to reducible errors that can be improved with additional training data whereas the effect of noise in the test data is irreducible.
Random noise

To showcase the influence of noise on a machine learning model, we use the noise-free dataset of artificial, additively enthalpies to train a d-MPNN model. The respective model performance with different sizes of the dataset is depicted in Fig. 1 for different levels of noise. For a clean, noise-free training, validation and test set (labeled “clean/clean”, left panel), a standard d-MPNN can learn the target property to seemingly arbitrary accuracy, because the task is simple and learnable. Adding Gaussian noise with standard deviation, i.e. magnitude, of 1 kcal/mol to the training data but not the test data (labeled “noisy/clean”, left panel) leads to a loss in performance, diverging after the RMSE for the clean model approaches the noise level. The model continues to learn with added data and could still achieve reasonable accuracies, requiring more data for the same performance compared to the model trained on the clean data. Though noise-based, the error from noise introduced while training is not irreducible. However, when noise also affects the test set (labeled “noisy/noisy”, left panel), it leads to an additional perceived loss in observed performance. The trained model is the exact same for the “noisy/clean” and “noisy/noisy” curves, only the test set differs in the addition of noise. The true model performance is thus described by the “noisy/clean” curve, but instead the noise in the test set causes the “noisy/noisy” curve to be observed. The learning curve of the noisy test and training set approaches an asymptote at 1 kcal/mol, which is the standard deviation of the employed noise distribution. Upon addition of more data, no further improvement in observed performance is perceived. The aleatoric limit is reached, where the observed test set error is dominated by noise. The effect of noise in the test set on the perceived model error is irreducible. This aleatoric limit is not a true limit of the model performance, however, but a property of the test set used to evaluate the model. Users who observe this sort of asymptotic behavior with respect to dataset size, should consider test set noise as a possible cause.

The right panel of Fig. 1 depicts the observed test set performance of a noisy test and training set with different levels of noise and different numbers of training points. We can see how the model performance changes as it approaches the aleatoric limit (dashed black line) where the RMSE equals the standard deviation of the noise distribution. With a small number of training points, such as 71 or 711 (indigo and violet curves), the test set error is not governed by noise (but instead dominated by bias and variance errors caused by the tiny number of datapoints), so that the magnitude of added random noise does not influence the observed performance significantly. As the aleatoric limit increases and approaches the performance of the other three dataset sizes, the RMSE of the datasets is deflected upward. As the noise level surpasses the baseline non-noise error for the dataset sizes, model performances converge and becomes indistinguishable as can be seen at the 1 kcal/mol noise level for the two largest dataset sizes. A similar trend with the presence of an aleatoric limit due to controlled addition of noise was also noted by Xie et al.\textsuperscript{62}

The noise we discussed so far was drawn from a Gaussian distribution. We also tested uniform, hyperbolic, and bimodal noise distributions, where the respective parameters were chosen so that each distribution had a standard deviation of 1 kcal/mol and was centered around 0 kcal/mol. Fig. 2 depicts the respective distributions and their observed model performances. Both the training and test set contain noise. We did not observe any difference in overall model performance between different error distributions, as long as the mean and standard deviation of the noise was the same, respectively. Though noise distributions found in real data
may be non-Gaussian, if homoscedastic, they should still follow the same trends of approaching an asymptote due to noise.

2. Systematic noise

If different regions of chemical space lead to larger noise levels, it is possible for a model to learn which regions are unreliable if the loss function is adapted accordingly, as first reported by Nix et al.\textsuperscript{38} When a model is trained using mean-variance estimation, the model outputs two values per target instead of one, namely the mean and variance. The two outputs of a mean-variance estimation model describe the model prediction probabilistically, with the mean being the center of the prediction distribution and the variance indicating the Gaussian spread of uncertainty around the mean. Other variations and extensions of mean-variance estimation also exist, such as evidential deep learning where the values returned by the model express uncertainty distributions for the values of the mean and variance.\textsuperscript{44} Mean-variance estimation and similar techniques can be very successful in training models on noisy datasets if the error is a function of the input features, since it allows the model to learn on which datapoints to focus, and which to regard as unreliable.\textsuperscript{34,46,63–65} However, it is not amenable to noise that is uniformly distributed over all datapoints or systematic noise that is applied based on external factors not represented in the input features of the training data. For example, if one measurement instrument had increased noise in data collection but the identity of the instrument used in collection was not included in the input features and could not be inferred from the input features, then the systematic noise applied according to the external factor of a faulty instrument would not be distinguishable. Concerns around sub-optimal performance of mean-variance estimation techniques have been recently reported in literature.\textsuperscript{65} We therefore recommend that users consider whether there are identifiable sources of systematic noise related to model input features and that they compare performance of a mean-variance estimation model against a simple model.

As with our demonstrations of behavior under random noise (Fig. 1, Fig. 2), we use the dataset of noise-free additive enthalpies to demonstrate behavior under systematic noise. We use two different cases of systematic noise application as demonstrations, using training dataset sizes of 711,613 datapoints. In the first case (Fig. 3, violet), we apply Gaussian noise of standard deviation 20 kcal/mol for Nitrogen-containing molecules and Gaussian noise of standard deviation 2 kcal/mol for non-Nitrogen-containing molecules. For this case we contrast the performance of uncertainty estimation by ensembling (bottom left) with the mean-variance estimation method (bottom right). As we discuss in a later section, ensembling is a measurement of variance error and does not directly incorporate noise error. Ensembling also requires a scaling calibration to match the magnitude of errors unless variance error dominates, so the ensemble uncertainty was scaled so that the ensemble and mean-variance estimation would have the same average value. In this case, the ensembling method of uncertainty estimation does a poor job of distinguishing the noise regimes, with a mean uncertainty 10.6 kcal/mol for molecules with negative original target enthalpy and mean uncertainty 11.5 kcal/mol for molecules with positive original target enthalpy. The mean-variance estimation is able to distinguish and quantify the noise regimes appropriately, with mean uncertainty 2.4 kcal/mol for molecules with negative original target enthalpy and mean uncertainty 19.5 kcal/mol for molecules with positive original target enthalpy. This example shows how mean-variance estimation can distinguish between noise regimes better than a method suited to other error types.

B. Bias

For noise-less datasets, the accuracy of a model in general increases with the size of a dataset, as visible and discussed in Fig. 1. The performance is also influenced by the model size, \textit{i.e.} the number of parameters, as well as the input representation and architecture of the model. These factors contribute to the error caused by model bias and are discussed in the following.

1. Data coverage

We first discuss model bias errors due to the number of datapoints, using models trained with the d-MPNN. Fig. 4, top left, depicts the model performance as a function of training set size and model size (size of hidden layers in the message passing and feed-forward networks). For a model of a given number of parameters, increasing
the number of datapoints increases the accuracy of the model’s predictions, where the slope on a log-log plot is nearly independent of the number of parameters. The error reduction with more data is presumptively data coverage error, but where is this data coverage error coming from? Is it caused by model bias, where a low number of datapoints does not allow the model to find the true global minimum in the high-dimensional parameter space? Or is it caused by variance, where differently initialized models converge to different trained models, and their average does not represent the true target? Our recent work applying Bayesian inference to ensembling uses the observed variation in prediction error within an ensemble to estimate the distribution of errors before variance is applied, i.e. the nonvariance component of the error.\textsuperscript{58} Utilizing this method, we decompose the total error into contributions from variance and bias (here, bias is computed as all error not from variance, which is a valid assumption for noise-less datasets). The mean absolute errors shown in Fig. 4 are for single models, though the inference of the bias and variance contributions was made using the distribution of predictions observed in ensembles of 5 submodels.

The bottom panel of Fig. 4 depicts the total error for hidden sizes of 2000 (left) and 20 (right) decomposed into variance (dashed line) and bias (dotted line). For small dataset sizes, the error in both models is dominated by bias from data coverage. For large dataset sizes the error is governed by variance, i.e. the disagreement between different models in the ensemble trained from differently initialized starting points. The two models, with a hidden size of 2000 and 20, differ in the absolute magnitudes of errors (with the smaller model exhibiting significantly larger errors), but the percentage of error being caused by model bias are similar, as shown in Fig. 4, top right panel. The percentage of error from bias due to data coverage thus decreases with increasing dataset size, irrespective of the size of the hidden layers of the model. Increasing the dataset size decreases errors from both bias and variance, but to a different extent, where bias decreases faster. Adding data is thus an important factor to decrease model bias and arrive at a model that is mainly limited by errors stemming from variance. In general, the vast chemical space makes data size and coverage a large source of error compared to other fields of research, where many chemical structures are unique or under-represented in (experimental) datasets. The implications of this shortcoming on uncertainty estimations are discussed later in this work. In the following, we first investigate other possible sources of model bias.
bias. A too small model (for example
decreases the absolute magnitude of the error from model
$h=2000$, shows that increasing the number of parameters
indefinitely is not advisable. A comparison of
target for a higher number of parameters for a given dataset
Figure 5. Mean absolute errors as a function of dataset size
for mean (dashed line) and sum (continuous line) aggregation
for d-MPNN (2D) and SchNet (3D) models. Left: Enthalpies

2. Model architecture and representation

As visible in Fig. 4, top left, the model performs bet-
ter for a higher number of parameters for a given dataset
size, but the effect levels off, so that adding more pa-
rameters indefinitely is not advisable. A comparison of
the dotted lines in Fig. 4, bottom panel, for $h=20$ and
$h=2000$, shows that increasing the number of parameters
decreases the absolute magnitude of the error from model
bias. A too small model (for example $h=20$) therefore
contributes to model bias and should be avoided.

Besides model size, there are also other factors con-
tributing to model bias, such as molecular representa-
tions and model architectures. We explore the effects of
architecture and representation by comparing the perfor-
ance of a d-MPNN to SchNet. Message passing neural
networks are built on 2-dimensional graph representa-
tions, whereas SchNet takes the 3-dimensional coordi-
nates as input. We therefore expect SchNet to perform
better for targets that depend on the 3-dimensional con-
formation, such as the enthalpy in the QM9 dataset.

The left panel of Fig. 5 depicts the mean absolute er-
rors of a d-MPNN and SchNet for HOMO-LUMO gaps.
The 3-D method (SchNet) needs more training data to
perform well than the simpler 2-D method (d-MPNN)
but provides better performance with very large data set
sizes. To choose the best model for a given data set, it
is therefore advisable to take into account the size and
diversity of the data. For small (or highly diverse and
sparse) data sets, a simpler model is often preferred.

Besides the general model architecture, many smaller
details and hyperparameters largely influence model per-
formance, too. We showcase this effect by examin-
ing the influence of the aggregation function that com-
bines atomic into molecular representations or proper-
ties. Both d-MPNN and SchNet first compute vectors of
properties for each atom in a molecule, and then com-
bine these atom vectors to construct a single fixed-length
learned-fingerprint vector for the molecule. This vector
is the input to a conventional feed-forward neural-net
in the d-MPNN, or directly produces the target within
SchNet. However, what is the best way to combine the
atom vectors, by averaging or summing? In Fig. 5(left),
one can see that either method of combination works
about equally well for predicting HOMO-LUMO gaps.
But in Fig. 5(right), the “sum” method works much bet-
ter than the “avg” method. This is because enthalpy is
an extensive quantity, that increases more or less linearly
with the number of atoms. If one averages (rather than
sums) the atom vectors, one loses information about how
many atoms are in the molecule. In contrast, the HOMO-
LUMO gap has a much weaker dependence on molecular
size, it is more like an intensive quantity, so it can be
modeled using “avg” about as well as it is modeled us-
ing “sum”. An extensive (size-conserving) representation
and architecture is therefore essential for size-extensive
properties like the energy.\(^66\) However, it can be easily
overlooked when training models, especially when train-
ing multi-task models for a mixed set of extensive and
intensive targets such as the QM9 dataset which contains
both. As visible in Fig. 5, choosing an intensive architec-
ture (averaging over all atoms) for an extensive property
such as the enthalpy leads to large performance losses for
both the d-MPNN and SchNet. For an intensive prop-
erty, there is nearly no difference in performance, so we
recommend using extensive representations and architec-
tures when in doubt.

In Fig. 5(right), we furthermore observe that the en-
thalpy which largely depends on the 3-dimensional con-
formation can be modeled by a 3-D approach in much
greater detail. However, a direct comparison is difficult
since SchNet not only differs in the general architecture,
but also in the way the model is initialized. Namely,
SchNet utilizes the mean and standard deviation of aver-
age atomic contributions to the target properties in the
training set to initialize the model with a good guess of
the target property of each molecule. This is especially
helpful for extensive properties since it enforces additiv-
ity of the atomic contributions. As such, d-MPNN and
SchNet are not directly comparable, since the d-MPNN
has to explicitly learn the additivity from the data.

3. Featurization

Once a model architecture and representation for
molecules (2D graph, 3D coordinates, fingerprint, string)
have been chosen, there are still many options for what in-
put features to use for the encoding of molecules within
that representation. The inclusion of features relevant
to the target property can make a significant differ-
ence in the ability of the model to learn the property
function.\(^66,67\) Errors due to the choice of input features
are a form of model bias. Fig. 6 depicts model perform-
ances for the QM9 targets enthalpy and HOMO-LUMO
gap for different model inputs. First, we skipped the mes-
sage passing step and used a Morgan fingerprint\(^56\) of size
10, 100 or 1000 as input to a feed-forward neural net-
work. Second, we modified the default d-MPNN repre-
sentation of the molecular graph not to discern between carbon and nitrogen (labeled 'd-MPNN C,N') or carbon, nitrogen, and oxygen (labeled 'd-MPNN C,N,O') to artificially create bad features. For both targets, d-MPNNs outperform fingerprints, where smaller fingerprints lead to even worse performance. Bad features in the d-MPNN again decrease model performance. With increasing data, models with corrupted features can re-gain performance, since bias from featurization is a reducible error source. Thus, finding optimal features is less important for large datasets, but essential for medium-sized datasets. Despite these insights being rather expected, we find that often not enough attention is paid to featurization when building new models. For example, targets like the enthalpy might require additional features such as ring sizes, which are not default in e.g. the implementation of d-MPNNs we utilized in this work. In fact, adding a one-hot encoded ring size to atom and bond features increases the performance of our d-MPNN from mean absolute errors of 0.30 to 0.19 eV for N=100,000. We also recently trained d-MPNNs to predict solute parameters, solvation free energies, enthalpies or solubilities, where we found atom features specific to solvation such as the presence of H-bond donors or acceptors to be key to good model performance.\textsuperscript{11,68} For the prediction of molecular UV-Vis absorption peaks, we furthermore found that the inclusion of a model prediction trained on low-fidelity data as an additional custom molecular feature within a high-fidelity model can be beneficial.\textsuperscript{69}

C. Variance

As detailed in the previous sections, reducing error from (non-systematic) noise and bias is a tedious and manual process that requires expertise and knowledge of the problem at hand. In contrast, error from variance can be tackled with an easy and automated, though computationally intensive method: ensembling.

1. The importance of ensembling

In this work, we produce ensembles of submodels by starting each training run from differently initialized model parameters, referred to as Deep Ensembling.\textsuperscript{40} Many other techniques for generating randomly differentiated submodels exist, such as bootstrapping,\textsuperscript{41} Monte-Carlo-dropout\textsuperscript{42} or saving snapshots\textsuperscript{43} from different training epochs.\textsuperscript{46,47} Our reported model prediction is the average of the predictions of the submodels in the ensemble (Eq. (1)). Fig. 7 depicts the observed mean absolute error of the test set of several d-MPNN models trained on the artificial enthalpy dataset as a function of the number of models in the ensemble. The different panels in Fig. 7 refer to different situations in which additional model errors have been introduced with data coverage (top left), different model sizes (top right), different magnitudes of noise in the training data (bottom left), and different featurization strategies (bottom right). Regardless of the sources of additional error in the model, ensembling always improves model performance. The performance with increasingly large ensembles will approach an asymptote. This is because ensembling purely addresses variance error. A large ensemble can remove variance error, as the ensemble prediction \( \bar{y}(X_u) \) will converge to the variance distribution mean \( \mu_f(X_u) \) (Eq. (5), (6)). However, bias and noise errors remain even when an ensemble size is made very large. The lower the error from other sources, the larger the performance gains available from ensembling. This
effect could already be anticipated from Fig. 4, where a larger contribution of variance error was observed for lower data coverage error. We further note that the performance gains for adding an additional submodel to an ensemble are diminishing, while the computational cost of training and saving models scales roughly linearly with the number of submodels. Adding a small number of additional submodels to improve the model performance may be justified against the costs while adding a large number of additional submodels may not be.

2. Ensemble variance as a measure for prediction error

Training an ensemble of models and inspecting the variance between predictions of the individual submodels furthermore is a popular method to estimate the uncertainty associated with a prediction.\textsuperscript{40,46,47,70,71} Ensemble uncertainties can be used for risk management or active learning, amongst others, and are thus valuable information when judging the reliability of a prediction. However, uncertainties from ensembles only represent the true error for variance-dominated systems, \textit{i.e.} the model uncertainty caused by model bias is not included. To showcase this, the deviation from the uncertainty from the ensemble variance and the true observed error was computed for all systems of Fig. 7 using an ensemble of five submodels.

There are several available methods to evaluate uncertainty predictions that take into account different aspects of uncertainty. Here we assess the quality of the uncertainty estimation by computing the calibration error curve, which is obtained by counting the fraction of test set datapoints that lie within a \( p \) confidence interval around the predicted value. Confidence intervals were modeled via Eq. (4) on a single split. For a perfectly calibrated model, the observed, empirical coverage (fraction of test set with targets within each interval) should equal \( p \), \textit{i.e.} 95\% of the test set should have a true target value within the 95\% confidence interval spanned by the ensemble mean and variance of each prediction. The area under the calibration error curve, AUCE, measures the deviation of the observed calibration curve from perfect calibration. An AUCE of 0 corresponds to perfect calibration, larger values indicate an imperfect calibration.

Calibration curves and the respective AUCEs for the considered models are shown in Fig. 8, where a very good calibration is observed for systems with low noise and bias (\( N = 711613, h \geq 100, \text{noise} \leq 0.02 \) and mean aggregation). In fact, the artificial dataset employed in this study is an ideal test case for calibration, because it features a controlled amount of noise, and can be approximated with an arbitrary level of accuracy with a sufficient amount of datapoints and model degrees of freedom. We find that Deep Ensembling of d-MPNNs yields a well-calibrated measure of uncertainty for a prediction in this case. However, when adding larger errors from noise or bias, worse values for the AUCE are observed, since the total error of each prediction is now dominated by other contributions, thus impacting the correlation between ensemble uncertainty and true error. Model bias is often ignored as an error source, but can significantly impact the ability of ensemble uncertainties to depict true errors.Datasets with a low amount of datapoints and large noise have been shown to lead to ill-calibrated models in literature (for example the Lipophilicity dataset in Ref. 47), but the contribution of data coverage and other sources of bias is usually overlooked relative to the contribution of noise or variance. Even our artificial, noise-less and easy-to-learn dataset leads to severely ill-calibrated models if the amount of training data is low (for example for \( N = 711 \) or 7116). The bottom panels of Fig. 4 explain this failure: A low amount of training data leads to a bias-dominated model, where the total error is nearly exclusively due to bias, not variance. Because many chemical datasets are made of only hundreds to thousands of datapoints,\textsuperscript{72,73} we expect many deep-learning models to suffer from calibration error caused by data coverage model bias and should not be assessed using ensemble variance alone.

3. Reported performance in cross-validation

Training and reporting several submodels for a task is generally a benefit to model performance, as shown above (Fig. 7). Instead of ensemble splitting with a consistent test set, many use cross-validation approaches where the data splits are varied across the different submodels to arrive at a split-independent estimate of the true model performance. We find that model performance improves with cross-validation as it does with ensembling if all obtained submodels are used for a prediction on a test datapoint. However, we also find that the way performance is
reported for cross-validation does not always include the benefits of using multiple submodels and can therefore lead to misconceptions about the quality of the overall model.

Cross-validation is an important and commonly-used measure to report model performance in a way that reduces the sensitivity to a particular data split. This is done by training models on multiple different splits of the training, validation, and test set and averaging the performance metrics on the different test sets. Though cross-validation involves the training of multiple submodels, it affects the reported performance metric differently than ensembling. The benefits of ensembling come from taking the predictions of each of the submodels and averaging them together to get superior predictions before evaluation. The ensemble model performs very similarly with its true performance of the models on new predictions. The benefits of using multiple submodels and can therefore lead to misconceptions about the quality of the overall model.

Cross-validation is an important and commonly-used measure to report model performance in a way that reduces the sensitivity to a particular data split. This is done by training models on multiple different splits of the training, validation, and test set and averaging the performance metrics on the different test sets. Though cross-validation involves the training of multiple submodels, it affects the reported performance metric differently than ensembling. The benefits of ensembling come from taking the predictions of each of the submodels and averaging them together to get superior predictions before evaluation. The metric from each submodel is roughly the same, so the reported averaged metric does not decrease with the use of more folds, as shown in Eq. (7). However, when we use cross-validation folds to make new predictions on a separate dataset, we treat the cross-validation submodels the same as with an ensemble and average together the predicted values to get superior performance, Eq. (8). This creates a mismatch between the expected model quality based on the reported average metric and the actual realized performance of ensembled predictions of the cross-validation models.

We demonstrate the differences in score reporting using models trained on the artificial enthalpy dataset under cross-validation or ensemble splitting practices, as depicted in Fig. 9. The models report errors on the test splits using the practices described for cross-validation folds (left) and ensembles (right), labeled “reported”. An additional independent test (disjoint from the other data used here) is then used to evaluate the quality of predictions made by the models using all submodels, labeled “independent test set”. The conventional implementation of k-fold cross-validation would be carried out such that a given datapoint will only appear in the test set of one fold; for the purposes of this demonstration we are not following that constraint but instead assigning a new independent random split for each submodel. This deviation from k-fold cross-validation maintains the data split sizes for fair comparison across Fig. 9, but other testing using strict k-fold cross-validation has shown the effect to occur then as well. The two model styles perform roughly the same on the secondary test set, showing the true performance of the models on new predictions. The ensemble model performs very similarly with its reported metrics as with its true performance. However, the cross-validation model reports a performance that is significantly worse than what is revealed with the secondary test set.

D. Splitting and Data Leakage

So far, we have presented how noise, bias and variance can lower the true or perceived performance of a model. An unsatisfactory model performance is detected easily, but the contributions of various error sources are often hardly distinguishable. We hope that the tools and insights presented above can aid scientists to better understand the sources of error in their models. This understanding can guide the next steps to optimize the model. Another possible pitfall comes when the performance of a model on the initial test set is much better than its performance on the actual molecules of interest, which can be hard to detect. In fact, models that report-edly perform well but fail in real-world applications are a major concern and setback within the machine learning community.\(^{29,30}\) In the following, we discuss two important reasons a model may seem to perform deceptively better than it actually does.

1. Generalization performance

Limitations in the model architecture and representation can be easily overlooked if the dataset only spans a small subset of chemical space. This may be the case for databases including only molecules with the same number of atoms, or related chemical structures. As an example, we illustrate how a wrong choice in the aggregation function (which combines atomic into molecular embeddings) for the QM9 target enthalpy can be overlooked if the size of the molecules in the dataset only spans a narrow range. To this aim, we split the QM9 dataset into molecules with 1-6 atoms and 7-9 atoms. The machine learning models are then trained solely using the dataset with 7-9 atoms. Fig. 10 depicts the performance on test sets containing molecules of size 1-6 and 7-9 for a mean (top left) and sum (top right) aggregation function. For the test set containing similarly sized molecules, both aggregation functions lead to acceptable performances,
and one might wrongly conclude that a mean aggregation function is a valid choice for an extensive target like the enthalpy. However, inspecting the test set performance on molecules of size 1-6 reveals that by using a mean aggregation function, the model does not gain any additional performance as more data is added. The bottom panels of Fig. 10 depict the absolute errors for each datapoint in the test set as a function of molecular size. Here, the failure of the model utilizing mean aggregation becomes apparent: molecules with a different size produce absolute errors up to four orders of magnitude larger than molecules with a similar size because the model implicitly learns the average size of the molecules in the training set to circumvent the shortcoming of mean aggregation. We note that for sum aggregation, the extrapolation performance to differently sized molecules is by no means perfect (bottom right panel), but the model at least learns to generalize to some extent. In general, an extrapolation error is visible regardless of the model architecture. This extrapolation error can be assigned to bias by lack of data coverage for molecules with a lower molecular weight.

2. Test set contamination

Another prominent reason for a deceptively good performance is data leakage, where the test set is too similar to the training set. Rigorously splitting a dataset into training, validation and test sets is a crucial task that can be overlooked easily, and may lead to drastically wrong reported performances. In the following, we showcase this pitfall by training a model of the QM9 target internal energy at temperatures $T$ equal 0 K and 298 K. We treat the temperature as an input (in addition to the molecular graph), and train on the single property $U(T)$. The temperature is appended to the aggregated molecular embedding (after the message-passing neural network, before the feed-forward neural network). If all datapoints are treated as independent, a massive amount of data leakage occurs since many molecules in the test set also occur in the training set, albeit at a different temperature. The left panel of Fig. 11, depicts the true (only test set datapoints without leakage) and perceived (datapoints with and without leakage) performance for test sets with a different number of datapoints that are leaked. Depending on the constitution of the test set, different mean absolute errors are observed. For test sets with a large amount of leakage, the model appears to perform deceptively well. This perceived performance does not depict the intended application case, where the model is supposed to predict the internal energy of a new molecule outside the training set. The left panel of Fig. 11, depicts the true (only test set datapoints without leakage) and perceived (datapoints with and without leakage) performance for test sets with a different number of datapoints that are leaked. Depending on the constitution of the test set, different mean absolute errors are observed. For test sets with a large amount of leakage, the model appears to perform deceptively well. This perceived performance does not depict the intended application case, where the model is supposed to predict the internal energy of a new molecule outside the training set. The right panel of Fig. 11 shows the distribution of the absolute errors of test datapoints that are held out during training versus datapoints leaked from the training set, where again the distribution of errors for leaked datapoints does not reflect reality, i.e., the distribution of errors for new, independent molecules.

The data leakage described above is easy to spot, but sometimes test set contamination occurs via more complex mechanisms. One example is our own Ref. 76 where a multitask model was trained on computational activation energies and reaction enthalpies of chemical reactions. Both forward and reverse reactions were included and were treated as independent datapoints, so some of
the reactions in the test set had their reverse counterpart in the training set. Out of several developed model architectures, one performed especially well, with accuracies close to chemical accuracy. However, this chosen architecture mainly excelled over the other potential architectures in exploiting the data leakage efficiently, leading to a seemingly good performance. When tested on independent reactions, however, the model produced errors about half an order of magnitude larger than reported. Only after removing the data leakage were we able to develop different architectures with better performance. In this case, test set contamination not only caused the reported test set error to be too low, but also hindered model development and optimization. Similar cases were reported in literature, where without a rigorous splitting strategy it was impossible to select the best architecture and parameters for reaction models. Splitting in systems involving multiple molecules that each may need rigorous splitting (e.g., solute-solvent pairs) creates an additional complication. In our previous work, we showed the importance of data splitting by excluding several solvents, solutes, and substructures from the training set. For chemical datasets, splitting according to molecular scaffolds or time-stamps can be an appropriate measure to prevent data leakage. In fact, the performance of many models was shown to drop significantly if evaluated on a more rigorous basis than simple random splits. However, scaffold splitting also comes with shortcomings, some of which have no easy remedy, as chemical space is inherently non-uniform and unbalanced. A detailed determination of optimal splitting strategies extends beyond the scope of this study. In general, we recommend rigorously splitting datasets when developing new models, and paying increased attention to possible sources of data leakage.

IV. DISCUSSION

Here, we discuss the main observed trends caused by noise, bias and variance errors. We also attempt to give practical advice on how to boost model performance in each of those cases.

Noise in a dataset leads to a true loss in performance, as well as an additional and significant perceived loss in performance, which may cause a model seemingly to stop learning as soon as the true model error falls below the aleatoric limit. Whenever an asymptotic behavior of the model performance is observed in the learning curve, test noise should be considered as a possible cause. One example of this is shown in Fig. 7 of Ref. 90. Further optimizing a model that has reached the apparent aleatoric limit is difficult, since a change in hyperparameters like model size or architecture will lead to the same perceived test set error even though the true performance (measured by a clean test set) may have improved significantly. It is therefore important to construct test sets with a low amount of noise to develop and optimize high-precision models. We have recently shown the importance of a low noise test set for training neural networks to predict solvation free energies and aqueous solid solubilities, where cleaning the test set from large errors was necessary to develop a meaningful model. When there is reason to believe that a dataset is affected by systematic noise, we recommend testing a model trained using mean-variance estimation or similar and comparing it against a simple model architecture.

For noise-less datasets, the reducible source of error is divided in a bias and variance term. Our recent application of Bayesian inference to ensembling allows users to quantify the error in both reducible contributions. By separating the contributions, it becomes possible to prioritize efforts between reducing model bias and model variance. Reducing the model bias error is tedious and requires user experience. Bias can be reduced by adding more data, and by choosing the best possible molecular representation, model architecture and set of features to relate the molecular structure to the target property. These challenges are particularly common in chemistry; the vast chemical space makes data size and coverage a large source of error compared to other fields of research, where many chemical structures are unique or under-represented in (experimental) datasets. The representation of molecules inside machine learning is without question one of the main challenges in chemical property prediction today. In other fields of machine learning such as computer vision or natural language processing, the size of an image or a sentence does usually not scale with the output target. For example, the number of words in a sentence or letters in a word do not tell us about its meaning, conveyed information or sentience. In contrast, for extensive properties the size of a molecule changes its properties significantly, so that representations and architectures developed in other fields of research must be properly adapted to chemical applications. Careful consideration of several representations and selecting the most appropriate for the target property is crucial in reducing the bias error. Properties may not always be easily delineated between intensive and extensive, so we recommend choosing extensive aggregations in chemical systems when in doubt.

Finding optimal features is important for medium-sized datasets (bias error by featurization reduces when the relation between structure and property can be learned from more data). The customization of atomic and molecular features for a task at hand is an important aspect of model optimization even for deep learning models because the optimal features are not selected automatically. Despite these insights being rather expected, we find that often not enough attention is paid to featurization when building new models.

The variance error can be reduced by, for example, Deep Ensembling, regardless of the other sources of error in the model. There is a trade-off between the gain in model performance and the computational load of training more models. For a quick assessment, we recommend
training an ensemble of five models and using the slope of the performance improvement from subsets of the five models to estimate whether additional models should be added. For a more extensive estimation of possible gains from ensembling, we recommend our method for projecting the expected error of different ensemble sizes from Ref. 58. Depending on the task, dataset, and architecture, but also the availability of computation power and intended use of the model, a different number of ensemble models will lead to the best trade-off between performance and computational workload.

We advise the reader to combine cross-validation with model ensembling. Cross-validation can be critically important for small datasets where the reported performance is very split-dependent. Whenever cross-validation is necessary, we recommend training an ensemble of submodels for each cross-validation fold instead of single submodels. Predictions can then be made using all submodels together across folds in a larger overall ensemble on independent data. The reported metrics from the ensembles within the folds will be somewhat pessimistic relative to the performance of the larger overall ensemble, but the effect will be significantly lessened compared to single-model folds.

Last but not least, we want to highlight the importance of avoiding data leakage, which unfortunately is rather common in chemical datasets. Leaked data and the associated overly optimistic reported model performance hamper the development of new models severely, reduce the confidence in machine learning models and delay their application to real-world scenarios. We therefore urge the reader to pay increased attention to data splitting when developing models on new datasets.

Noise inherent to data is commonly found as experimental uncertainty in chemical datasets. The presence of noise has a different effect on the perceived model performance depending on whether it is found in the training and/or test set. Noise in the test set leads to an observed aleatoric limit and can cause an underestimation of the true model performance. We furthermore highlighted challenges in predicting properties of molecules, such as the choice of size-conserving representations and architectures for the prediction of size-extensive targets. Limitations in the dataset size, model architecture, or representations can cause the overall model error to be dominated by the contributions of model bias. We discuss ensembling as a reliable method to reduce model variance error and the value of using statistical tools to evaluate the portion of the error due to variance. However, in situations where noise or bias error dominate, ensembling cannot be used to correct for those errors, and ensemble variance becomes ineffective at estimating whole-model uncertainty. Lastly, we showcased the effects of splitting and data leakage when assessing the real-world performance of a model and strongly advise researchers to pay close attention to meaningful data splits avoiding leakage.

In summary, machine learning is a valuable and important tool to predict physicochemical properties, but can suffer from error sources uncommon to other fields of research. Increased attention should be paid to noise and bias from data coverage, model architecture and representation to identify and remedy shortcomings of chemistry-related deep learning models concerning their performance and uncertainty calibration.

V. CONCLUSION

We have demonstrated the role of noise, bias and variance for the perceived and true performance of machine learning models, focusing on chemical applications. Understanding the possible sources of errors in an underperforming model is an important prerequisite to identifying potential improvements.

ACKNOWLEDGEMENT

E.H. acknowledges support from the Austrian Science Fund (FWF), project J-4415. The authors acknowledge the Machine Learning for Pharmaceutical Discovery and Synthesis Consortium (MLPDS) for funding. Parts of the data reported within this paper were generated on the MIT SuperCloud Lincoln Laboratory Supercomputing Center. We also thank our colleagues Lagnajit Patanaik and Kevin Greenman for helpful review and discussion of this study.

* These two authors contributed equally
† whgreen@mit.edu
8 K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, et al.,


87 X. Li and D. Fourches, J. Cheminformatics 12, 1 (2020).


89 Y. Zhang et al., Chem. Sci. 10, 8154 (2019).
