# Uncertainty quantification in deep networks for material property predictions

Kevin Tran,[†,¶] Willie Neiswanger,[‡,¶] Junwoong Yoon,[†] Eric Xing,[‡] and Zachary W. Ulissi[*,†]

†*Chemical Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15217*

‡*Machine Learning Department, Carnegie Mellon University, Pittsburgh, PA 15217*

¶*These authors contributed equally to this work*

E-mail: zulissi@andrew.cmu.edu

**Abstract**

Abstract here.

## Introduction

The fields of catalysis and materials science are burgeoning with methods to screen, design, and understand materials.[1–4] This research has spurned the creation of Machine Learning (ML) models to predict various material properties. Unfortunately, the design spaces for these models are sometimes too large and intractable to sample completely. These under-sampling issues can limit the training data and therefore the predictive power of the models. It would be helpful to have an uncertainty quantification (UQ) for a model so that we know when to trust the predictions and when not to. More specifically: UQ would enable various online, active frameworks for materials discovery and design (e.g., active learning,[5]

online active learning,[6] Bayesian optimization,[7] active search,[8] or goal oriented design of experiments[9]).

Such active frameworks have already been used successfully in the field of catalysis and materials informatics. For example: Peterson[10] has used a neural network to perform online active learning of nudged elastic band (NEB) calculations, reducing the number of force calls by an order of magnitude. Torres et al.[11] have also used online active learning to accelerate NEB calculations, but they used a Gaussian Process (GP) model instead of a neural network. Jinnouchi et al.[12] have used online active learning to accelerate molecular dynamics simulations. Each of these active methods are underpinned by models with UQ, which has garnered increasing attention itself.[13,14]

The goal of UQ in predictive models is to quantify the relative likelihood of potential true outcomes associated with a predicted quantity, and to accurately assess the probabilities of these outcomes. For example, given an input for which we wish to make a prediction, a predictive UQ method might return a confidence interval that aims to capture the true outcome a specified percentage of the time, or might return a probability distribution over possible outcomes. Performance metrics for predictive UQ methods aim to assess how well a given quantification of the probabilities of potential true outcomes adheres to a set of observations of these outcomes. Some of the performance metrics for predictive UQ are agnostic to prediction performance—they provide an assessment of the uncertainty independent of the predictive accuracy (i.e. a method can predict badly, but could still accurately quantify its own uncertainty).

To our knowledge though, we have not seen many comparisons of different methods for UQ within the field of catalysis and materials informatics. Here we attempt to resolve this issue by benchmarking different methods for UQ (Figure 1). We acknowledge that there will not be one optimal method across all use cases, but we still find value in sharing these results so that others can build intuition from our results. Perhaps more importantly, we have also establish a protocol for comparing the performance of different modeling and UQ methods.


Figure 1: Placeholder for overview of the paper

# Methods

## Data handling

All regressions in this paper were performed on a dataset of Density Functional Theory (DFT) calculated adsorption energies created with the Generalized Adsorption Simulator for Python (GASpy).[15,16] These data included energies from 21,269 different H adsorption sites; 1,594 N sites; 18,437 CO sites; 2,515 O sites; and 3,464 OH sites; totaling in 47,279 data points. GASpy performed all DFT calculations using the Vienna Ab-initio Simulation Package (VASP)[17–20] version 5.4 implemented in the Atomic Simulation Environment (ASE).[21] The revised Perdew-Burke-Ernzerhof (rPBE) functionals[22] were used along with VASP's pseudopotentials, and no spin magnetism or dispersion corrections were used. Bulk relaxations were performed with a $10 \times 10 \times 10$ k-point grid and a 500 electron volts (eV) cutoff, and only isotropic relaxation were allowed during this bulk relaxation. Slab relaxations were performed with k-point grids of $4 \times 4 \times 1$ and a 350 eV cutoff. Slabs were replicated

in the X/Y directions so that each cell was at least 4.5 Å wide, which reduces adsorbate self-interaction. Slabs were also replicated in the Z direction until they were at least 7 Å thick, and at least 20 Å of vacuum was included in between slabs. The bottom layers of each slab were fixed and defined as those atoms more than 3 Å from the top of the surface in the scaled Z direction.

To split the data into train/validate/test sets, we enumerated all adsorption energies on monometallic slabs and added them to the training set manually. We did this because some of the regression methods in this paper use a featurization that contains our monometallic adsorption energy data,[15] and so having the monometallic adsorption energies pre-allocated in the training set prevented any information leakage between the training set and validation/test sets. After this allocation, we performed a 64/14/20 train/validate/test split that was stratified[23] by adsorbate. We then used the validation set's results to tune various hyperparameters manually. After tuning, we calculated the training set results and present them in this paper exclusively. Note that the test results were obtained using models that were trained only using the training set, not the validation set. This is acceptable because we only seek to compare methods here, not to optimize them.

## Regression methods

We explore a number of predictive UQ methods that aim to quantify the uncertainty for regression procedures, where the predicted quantity is a continuous variable. In order to standardize the assessment of performance, we ensure that each UQ method returns predictive uncertainty results in a consistent format: a distribution over possible outcomes of the predicted quantity for any specified input point. This result format allows us to compute all of the predictive uncertainty performance metrics which we will introduce in subsequent sections. Figure 2 illustrates all of the methods we investigate in this study, and we describe each method in detail below.

**NN:** To establish a baseline for predictive accuracy, we re-trained a previously reported

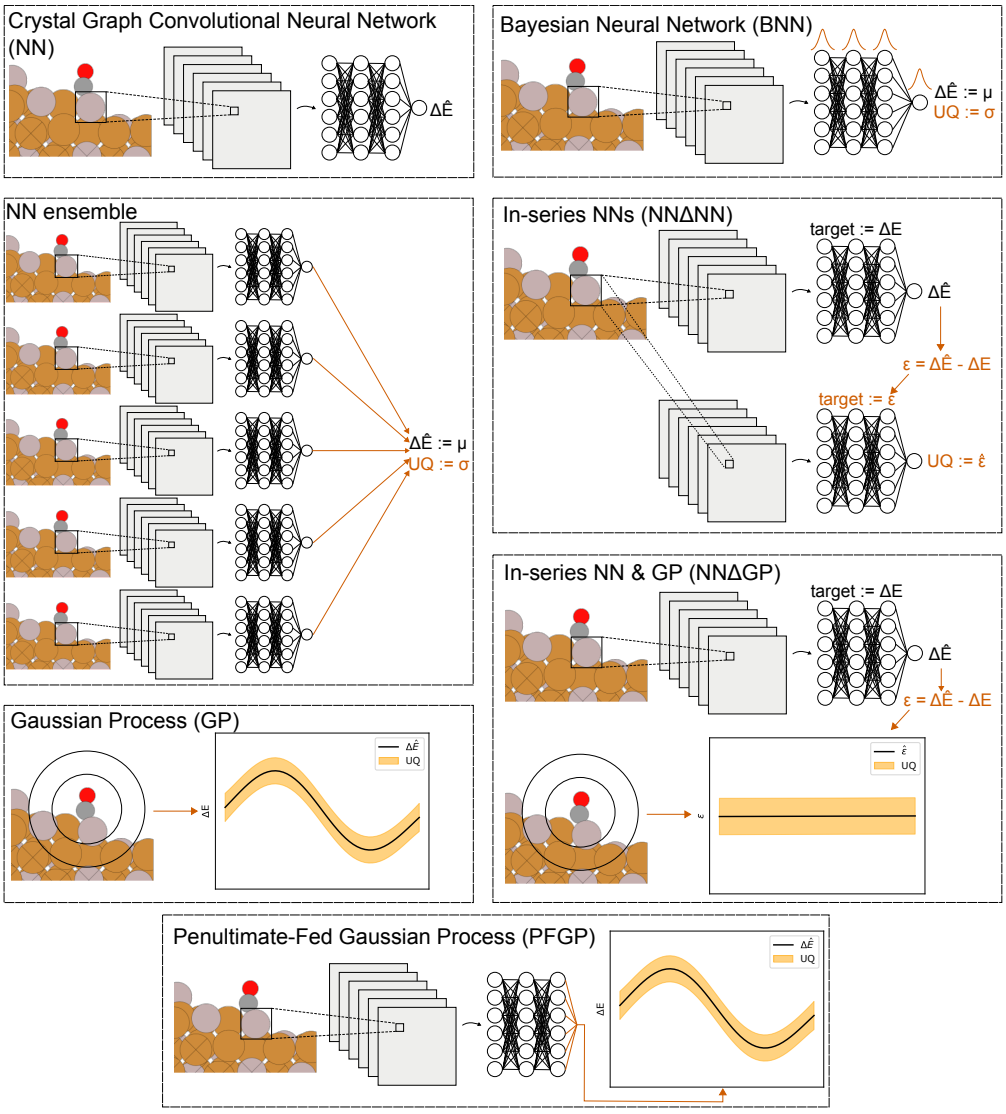



Figure 2: Overview of the various UQ methods we investigated in this study. $\Delta E$ represents DFT-calculated adsorption energies; $\Delta \hat{E}$ represents ML-predicted adsorption energies; $UQ$ represents ML-predicted uncertainty quantifications; $\mu$ represents the mean of a sample of points; $\sigma$ represents the standard deviation of a sample of points; $\epsilon$ represents the residuals between DFT and ML; and $\hat{\epsilon}$ represents the residuals between ML-predicted $\epsilon$ and the actual $\epsilon$.

crystal graph convolutional Neural Network (NN)[24,25] on this study's training set. This NN model projects a three-dimensional atomic structure into a graph, which is then fed into a convolutional neural network to predict various properties. In this case, we predict DFT-calculated adsorption energies, $\Delta$E. Reference Back et al.[25] for additional details.

**NN Ensemble:** We created an ensemble of NNs by K-fold subsampling the training data into five different folds and then training individual NN models on each of these folds. Thus the mean of each of these models' predictions is the final prediction of the ensemble, and the standard deviation of the individual predictions is the ensemble's estimate of uncertainty.

**BNN:** The aim of Bayesian Neural Network (BNN) is to determine the posterior distribution of model parameters rather than a single optimal value of the parameters. In practice, inferring true posterior is very difficult and even infeasible in most cases. Thus, we approximate the model posterior to be as close as possible to the true posterior by doing variational inference. This could be done by training the BNN to minimize the Kullback-Leibler divergence between the variational distribution and the true posterior. We sampled the model parameters K times from the approximated posterior, and used the mean of these predictions as the final prediction and the standard deviation of these predictions as the estimation of uncertainty. We implemented the BNN and performed variational inference using Pyro.[26]

**NN$\Delta$NN:** Suppose we have trained a NN. We may aim to empirically fit an additional mapping that predicts the error of the first NN. Here we show in-series NNs (NN$\Delta$NN), which trains a secondary NN to predict the residuals of the initial NN. When training the first NN, we hold out 10% of the training data. Afterwards, we use the residuals of the initial NN on the held-out portion as training data for the second NN. After the secondary training, this second NN can predict residuals for the first NN on some new set of input data. The predictions of the second NN can then be used as uncertainty estimates.

**GP:** GPs are one of the most common regression methods for producing UQs, and so we use them here as a baseline. We fit a standard GP using the same exact features that

we used in previous work.[15] These features are defined by the elements coordinated with the adsorbate and by the elements of its next-nearest neighbors. Specifically: We use the atomic numbers of these elements, their Pauling electronegativity, a count of the number of atoms of each element near the adsorbate, and the median adsorption energy between the adsorbate and the elements. To ensure that these features interacted well with the GP's kernel, we normalized each of the features to have a mean of zero and standard deviation of one. Reference Tran and Ulissi[15] for additional details. To define the GP, we assumed a constant mean and used a Matern covariance kernel. We trained the length scale of the Matern kernel using the Maximum Likelihood Estimation (MLE) method. All GP training and predictions were done with GPU acceleration as implemented in GPyTorch.[27]

**NN$\Delta$GP:** GPs are Bayesian models in which a prior distribution is first specified and then updated given observations to yield a posterior distribution. The mean of this posterior distribution is used for regression, and the covariance matrix is used for UQ. Typically, in lieu of any additional prior knowledge, practitioners will take the prior distribution to have zero-mean. However, we could instead supply an alternative curve for the prior mean, and then perform the usual Bayesian updates to compute the posterior of this GP given observations. Here, for the GP prior mean, we supply the prediction given by a single pre-trained NN. We call this method in-series NN & GP (NN$\Delta$GP). For the GP, we used a Matern covariance kernel, where we fit the kernel hyperparameters using MLE. All GP training and predictions were done with GPU acceleration as implemented in GPyTorch.[27]

**PFGP:** A limitation of using this formulation of a GP with NN-predicted mean is that it requires the use of hand-crafted features for the GP. This requirement reduces the transferability of the method to other applications where such features may not be readily available. To address this, we formulated a different method whereby we first train a neural network on the learning task (i.e., predict adsorption energies), and then we use the outputs of the penultimate layer of the network as features in a new GP. The GP would then be trained to use these features to produce both mean and uncertainty predictions on the adsorption

energies. We call this a Penultimate-Fed Gaussian Process (PFGP). In this case, we used the baseline NN as the network from which we obtained the penultimate outputs. We also normalized the penultimate outputs of the NN so that each output would have a mean of zero and a standard deviation of one. To define the GP, we assumed a constant mean and used a Matern covariance kernel. We trained the length scale of the Matern kernel using the MLE method. All GP training and predictions were done with GPU acceleration as implemented in GPyTorch.[27]

## Performance metrics

We used six different metrics to quantify the accuracy of the various models: Median Absolute Error (MDAE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Relative Percent Difference (MARPD), and $R^2$ correlation coefficient ($R^2$). We used MDAE because is insensitive to outliers and is therefore a good measure of accuracy for the majority of the data. We used RMSE because it is sensitive to outliers and is therefore a good measure of worst-case accuracy. We used MAE because it lies between MDAE and RMSE in terms of sensitivity to outliers. We used MARPD and $R^2$ because they provide normalized measures of accuracy that may be more interpretable for those unfamiliar with adsorption energy measurements in eV. MARPD values were calculated with Equation 1 where $n$ is the index of a data point, $N$ is the total number of data points, $x_n$ is the true value of the data point, and $\hat{x}_n$ is the model's estimate of $x_n$. In this case, $x_n$ is a DFT-calculated adsorption energy and $\hat{x}_n$ is the surrogate-model-calculated adsorption energy. The ensemble of these metrics provide a more robust view of accuracy than any one metric can provide alone.

$$MARPD = \frac{1}{N}\sum_{n=1}^{N}\left|100 \cdot \frac{\hat{x}_n - x_n}{|\hat{x}_n| + |x_n|}\right| \tag{1}$$

To assess the calibration (or "honesty") of these models' UQs, we created calibration

curves. A calibration curve "displays the true frequency of points in each [prediction] interval relative to the predicted fraction of points in that interval", as outlined by Kuleshov et al. [28]. In other words: We used the standard deviation predictions to create Gaussian-shaped prediction intervals around each test point, and then we compared these intervals to the models' residuals at these points. If the residuals tended to fall outside the prediction intervals too often, then the UQs were considered overconfident. If the residuals tended to fall inside the prediction intervals too often, then the UQs were considered underconfident. Thus "well-calibrated" models had residuals that created a Gaussian distribution whose standard deviation was close to the model's predicted standard deviations. We discuss calibration curves in more detail in the Results section alongside specific examples.

As Kuleshov et al. [28] also pointed out, well-calibrated models are necessary but not sufficient for useful UQs. For example: A well-calibrated model could still have large uncertainty estimates, which are inherently less useful than well-calibrated and small uncertainty estimates. This idea of having small uncertainty estimates is called "sharpness", and Kuleshov et al. [28] define it with Equation 2

$$sha = \frac{1}{N}\sum_{n=1}^{N} var(F_n) \tag{2}$$

where $var(F_n)$ is the variance of the cumulative distribution function $F$ at point $n$. This is akin to the average variance of the uncertainty estimates on the test set. Here we propose and use a new formulation (Equation 3) where we add a square root operation. This operation gives the sharpness the same units as the predictions, which provides us with a more intuitive reference. In other words: Sharpness is akin to the average of the ML-predicted standard deviations.

$$sha = \sqrt{\frac{1}{N}\sum_{n=1}^{N} var(F_n)} \tag{3}$$

Finally, we assess the performance of each predictive uncertainty method via a the nega-

tive log-likelihood (NLL) on a held out set of data. This metric provides an overall assessment that is influenced by both the predictive accuracy of a method as well as the quality of its uncertainty quantification. Previous work has shown the NLL to be a proper scoring rule, which intuitively means that it provides a fair quantitative assessment of the uncertainty quantification and that it can be decomposed into terms that relate to both calibration and sharpness. We include results for NLL since it is a popular performance metric that has been used to quantify uncertainty in a variety of prior work and provides an additional single score for uncertainty quantification methods. Let $\ell_n$ denote the natural logarithm of the predictive probability density function at a given held-out point $n$. We then define NLL using Equation 4. In general, a lower NLL value indicates a better fit.

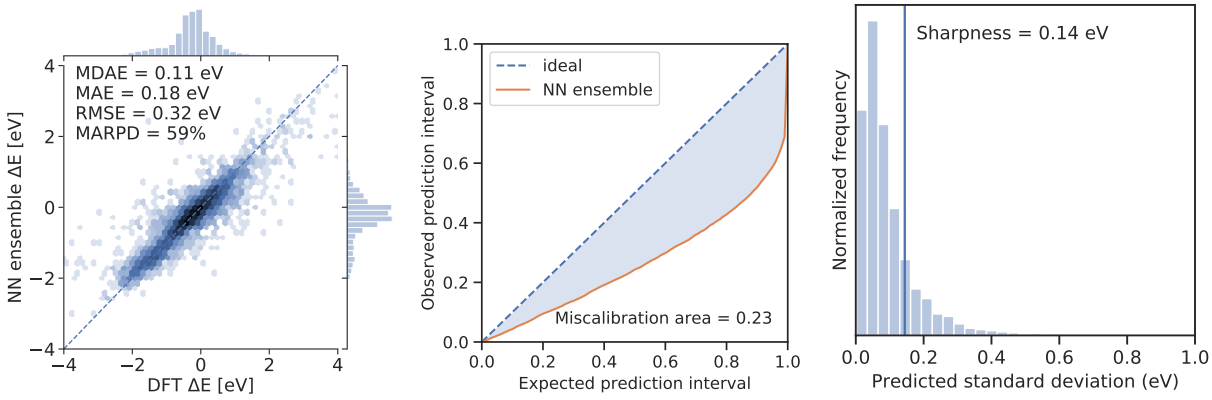
$$NLL = -\sum_{n=1}^{N} \ell_n \tag{4}$$

# Results

## Illustrative examples

Let us first discuss the results of our NN ensemble for illustrative purposes. Figure 3 contains a parity plot, calibration curve, and predicted-uncertainty distribution of our NN ensemble model. The parity plot shows the accuracy of the model; the calibration curve shows the honesty of the model's uncertainty predictions; and the uncertainty distribution shows the sharpness of model's uncertainty predictions. Accurate models have parity plots whose points tend to fall near the diagonal parity line. Calibrated models have calibration curves that approach the ideal diganoal line. Sharp models have uncertainty distributions that tend towards zero. Note that sharpness should not be won at the cost of calibration.

The calibration curve was created by first normalizing all test residuals by their respective uncertainty values, i.e. we divided the residuals by their predicted standard deviations. If we assume that the normalized residuals follow a Gaussian distribution, then 68% of the



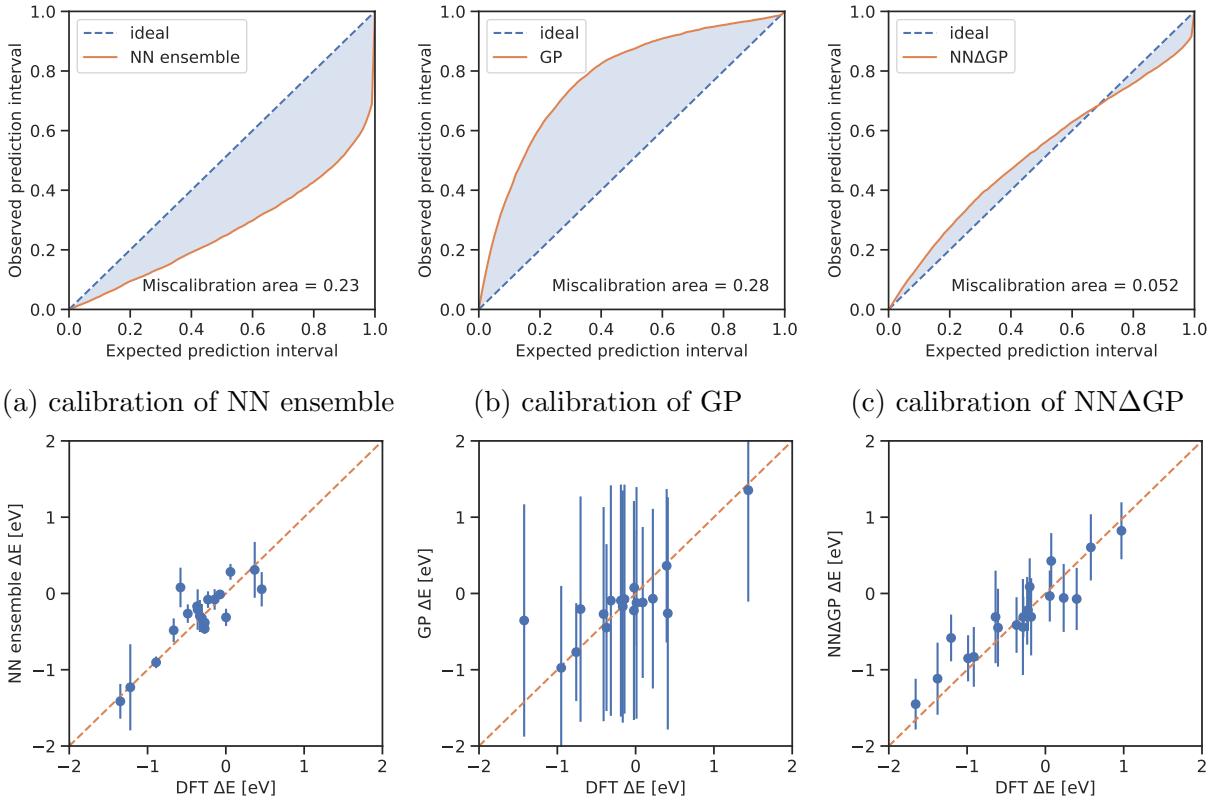

(a) parity plot (b) calibration curve (c) uncertainty distribution

Figure 3: Results of the NN ensemble. Each figure here was created with the test set of 8,289 points.

normalized residuals would fall between [-1, 1], 95% of them would fall between [-2, 2], 99% of them would fall between [-3, 3], etc. To challenge this assumption, we plotted the actual fraction of points within each prediction interval against the expected fraction of points. This plot is a calibration curve. Thus a perfectly calibrated model would have normalized residuals that are perfectly Gaussian, which would yield a diagonal calibration line. Therefore, models' calibration could be qualified by the closeness of their calibration curves to this ideal, diagonal curve. We quantified this closeness by calculating the area between the calibration curve and the ideal diagonal. We call this the miscalibration area, and smaller values indicate better calibration.

The shape of a calibration curve could also yield other insights. If a model's UQs were too low/confident, then the normalized residuals would be too large and they would fall outside their expected prediction intervals too frequently. This would result in a lower fraction of points observed within the expected prediction intervals, which would correspond to a calibration curve that falls below the ideal diagonal. Therefore, overconfident models yield calibration curves that fall under the ideal diagonal, and underconfident models yield calibration curves that fall over the ideal diagonal. Figure 4 illustrates this point by plotting calibration curves of various models alongside their parity plots that contain error bars

corresponding to ±2 standard deviations.



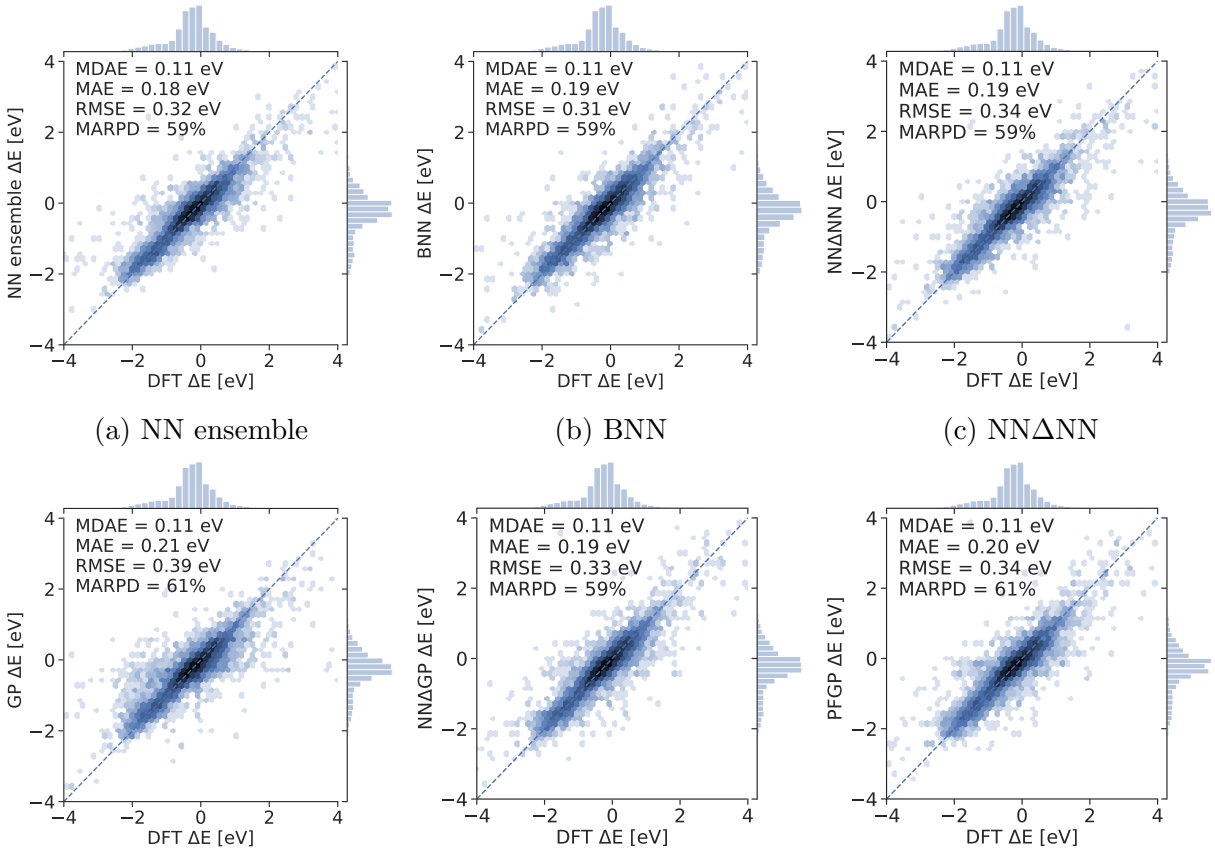

(a) calibration of NN ensemble (b) calibration of GP (c) calibration of NNΔGP

(d) parity of NN ensemble (e) parity of GP (f) parity of NNΔGP

Figure 4: Calibration curves and parity plots of an overconfident NN ensemble, an underconfident GP, and better-calibrated NNΔGP. The vertical uncertainty bands in the parity plots indicate ±2 standard deviations in the uncertainty predictions of each model. For clarity, we sampled only 20 points of the 8,289 test points to put in the parity plots. It follows that relatively overconfident models would have more points with uncertainty bands that do not cross the diagonal parity line; relatively underconfident models would have more points that cross the diagonal parity line; and a well-calibrated model would have *ca.* 19 out of 20 points cross the parity line.

## Summary results

Figure 5 contains parity plots for all UQ methods studied here; Figure 6 contains all calibration curves; and Figure 7 contains all distribution plots of the ML-predicted UQs. These figures illustrate the accuracy, calibration, and sharpness of the different UQ methods, respectively. Table 1 lists their performance metrics.



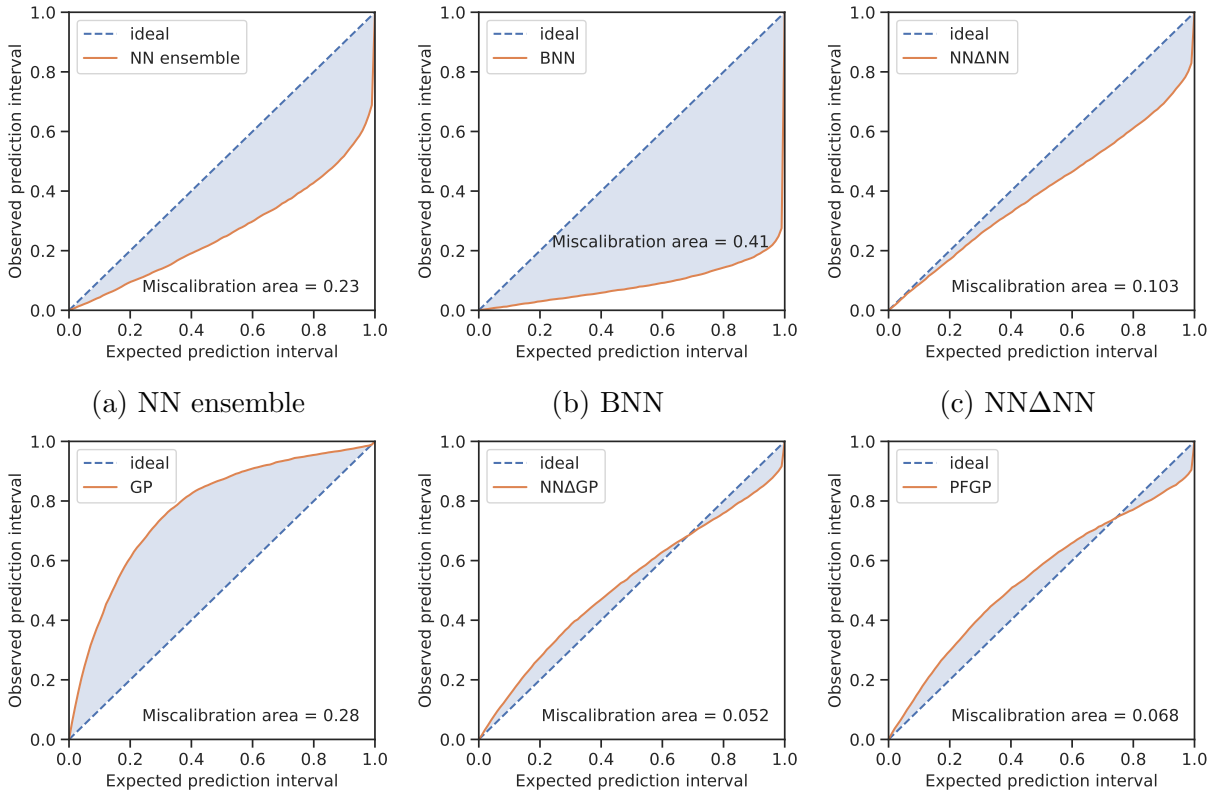

(a) NN ensemble (b) BNN (c) NNΔNN

(d) GP (e) NNΔGP (f) PFGP

Figure 5: Parity plots for all UQ methods used in this study. Shading plots were used in lieu of scatter plots because the large number of test points (8,289) obfuscated patterns. Darker shading indicates a higher density of points. Logarithmically scaled shading was used to accentuate outliers. The dashed, diagonal lines indicate parity.

Table 1: Performance metrics for all methods used in this study, which include: Median Absolute Error (MDAE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Relative Percent Difference (MARPD), $R^2$ correlation coefficient ($R^2$), miscalibration area (MisCal), sharpness (Sha), and negative log-likelihood (NLL). The units of MDAE, MAE, RMSE, and sharpness are all in eV. The units of MARPD are in %. The miscalibration area and NLL are unitless.

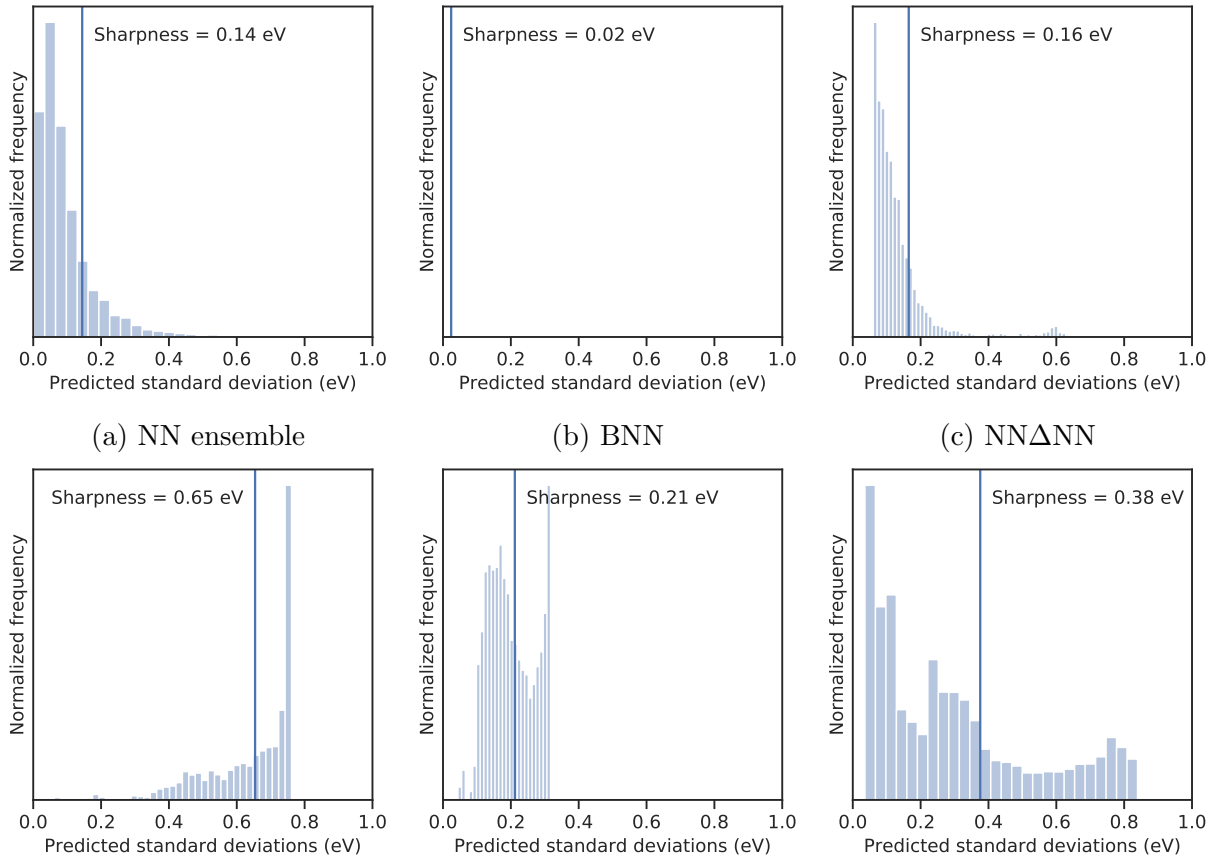
| Method | MDAE | MAE | RMSE | MARPD | $R^2$ | MisCal | Sha | NLL [$\cdot 10^3$] |
|---|---|---|---|---|---|---|---|---|
| NN | 0.11 | 0.19 | 0.34 | 61 | 0.80 | N/A | N/A | N/A |
| NN ensemble | 0.11 | 0.18 | 0.32 | 59 | 0.82 | 0.23 | 0.14 | 192.08 |
| BNN | 0.11 | 0.19 | 0.31 | 59 | 0.83 | 0.41 | 0.03 | 669.61 |
| NNΔNN | 0.11 | 0.19 | 0.34 | 59 | 0.80 | 0.10 | 0.16 | 18.61 |
| GP | 0.11 | 0.21 | 0.39 | 61 | 0.73 | 0.28 | 0.65 | 6.41 |
| NNΔGP | 0.11 | 0.19 | 0.33 | 59 | 0.81 | 0.05 | 0.21 | 6.09 |
| PFGP | 0.11 | 0.20 | 0.34 | 61 | 0.80 | 0.07 | 0.38 | 14.65 |




(a) NN ensemble (b) BNN (c) NNΔNN

(d) GP (e) NNΔGP (f) PFGP

Figure 6: Calibration curves for all UQ methods used in this study. Dashed, blue lines indicate perfect calibration while solid orange lines indicate the experimental calibration of the test set. The blue, shaded area between these lines is defined as the miscalibration area.




(a) NN ensemble (b) BNN (c) NN$\Delta$NN

(d) GP (e) NN$\Delta$GP (f) PFGP

Figure 7: Distribution plots of the ML-predicted standard deviations for each method. Sharpness values are indicated by vertical lines.

Regarding accuracy: All methods' MDAE results are virtually identical, and their MAE results are within 10% of each other. This suggests that all methods have comparable predictive accuracies for inliers. The plain GP has a higher RMSE value than the rest of the methods though. Thus our results suggest that our NN-based methods are more accurate at outlier prediction than our sole non-NN-based method. Each of our NN-based methods may not have practically different accuracies between each other though.

Regarding calibration: The NN ensemble and BNN are overconfident; the GP is underconfident; and the NN$\Delta$NN, NN$\Delta$GP, and PFGP models are relatively calibrated. The three more calibrated methods all share a characteristic that the other methods do not: They all start with a NN that is dedicated for prediction alone, and then they end with some other in-series method to estimate uncertainty. Interestingly, this in-series method of learning predictions and then learning uncertainties is similar in spirit to how deep networks "learn in stages" at each subseqent hidden layer.

Regarding sharpness: The NN ensemble and BNN yield the sharpest uncertainties, although both do so at the cost of calibration. Among the three more calibrated models, the PFGP yields the least sharp uncertainties (0.38 eV); NN$\Delta$GP yields a moderate sharpness (0.21 eV); and NN$\Delta$NN yields the lowest sharpness (0.16 eV). Note how GP-based UQ methods tend to yield less sharp uncertainties than methods based purely on NNs. This suggests that GPs may yield more conservative UQs, for better or for worse.

Regarding NLL: The NN$\Delta$GP and GP methods yield the best (i.e., lowest) NLL values. Note how the under-confident GP model has a worse miscalibration area and sharpness than the NN$\Delta$NN and PFGP models, but a better NLL value. Simultaneously, the two most over-confident models (NN ensemble and BNN) yield the worst NLL results. This shows that better NLL values correlate with relatively conservative estimates of UQ, but not with relatively liberal estimates. In other words: If we use NLL as our main performance metric, then we will favor under-confident UQ estimates in lieu of over-confident estimates.

Given the performance metrics for accuracy, calibration, sharpness, and NLL, we expect

the NN$\Delta$GP method to yield the best performing UQ model for our dataset. It yields the lowest miscalibration area and NLL while maintaining a relatively moderate accuracy and sharpness.

When choosing UQ methods for different applications though, other factors should be considered. For example: Although the NN$\Delta$GP method performed the best here, it required the use of a hand-crafted set of features. If future researchers wish to use the NN$\Delta$GP method to predict other properties from atomic structures, they may have to define their own set of features. This process of feature engineering is non-trivial and varies from application to application. In some cases, it may be easier to use a UQ method that does not require any additional features beyond the NN input, such as NN$\Delta$NN or PFGP.

Another factor to consider is the overhead cost of implementation. For example: The NN ensemble method is arguably the simplest NN-based UQ method used here and may be the easiest or fastest method to implement. Conversely, NN ensembles also have a higher computational training cost than some of the other methods used here, such as NN$\Delta$NN or PFGP. This high training cost is exacerbated if the ensemble is meant to be used in an active framework where the model needs to be trained continuously. As another example: The BNN method yielded perhaps the worst results of all the methods studied here. It could be argued that further optimization of the BNN could have resulted in higher performance. But creation and training of BNNs is still an active area of research with less literature and support than non-Bayesian NNs or GPs. This lack of support led to us spending nearly twice as long creating a BNN compared to the other methods. It follows that further optimization of the BNN would be non-trivial and may not be worth the overhead investment.

# Conclusions

1. Relative accuracies

2. Relative calibrations

3. Relative sharpnesses

4. Overhead

5. Calibration

## Code availability

Visit `https://github.com/ulissigroup/uncertainty_benchmarking` for the code used to create the results discussed in this paper. The code dependencies are listed inside the repository.

## Author information

Corresponding author email: zulissi@andrew.cmu.edu. The authors declare no competing financial interest.

## Acknowledgements

This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We also acknowledge the exceptional documentation and clear examples in the GPyTorch[27] repository, which formed the basis on much of the GP code used for this work.

# References


(1) Medford, A. J.; Kunz, M. R.; Ewing, S. M.; Borders, T.; Fushimi, R. *ACS Catalysis* **2018**, *8*, 7403–7429.

(2) Gu, G. H.; Noh, J.; Kim, I.; Jung, Y. *Journal of Materials Chemistry A* **2019**, *7*, 17096–17117.

(3) Schleder, G. R.; Padilha, A. C. M.; Acosta, C. M.; Costa, M.; Fazzio, A. *Journal of Physics: Materials* **2019**, *2*, 1–46.

(4) Alberi, K. et al. *J. Phys. D: Appl. Phys* **2019**, *52*, 1–48.

(5) Settles, B. In *Synthesis Lectures on Artificial Intelligence and Machine Learning*; Brachman, R. J., Cohen, W. W., Dietterich, T. G., Eds.; Morgan & Claypool, 2012; p 100.

(6) Chu, W.; Zinkevich, M.; Li, L.; Thomas, A.; Tseng, B. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* **2011**, 195–203.

(7) Frazier, P. I. **2018**,

(8) Garnett, R.; Krishnamurthy, Y.; Xiong, X.; Schneider, J.; Mann, R. *Proceedings of the 29th International Conference on Machine Learning, ICML 2012* **2012**, *2*, 1239–1246.

(9) Kandasamy, K.; Neiswanger, W.; Zhang, R.; Krishnamurthy, A.; Schneider, J.; Poczos, B. **2018**,

(10) Peterson, A. A. *Journal of Chemical Physics* **2016**, *145*.

(11) Torres, J. A. G.; Jennings, P. C.; Hansen, M. H.; Boes, J. R.; Bligaard, T. **2018**, *156001*, 1–6.

(12) Jinnouchi, R.; Lahnsteiner, J.; Karsai, F.; Kresse, G.; Bokdam, M. *Physical Review Letters* **2019**, *122*, 225701.

(13) Peterson, A. A.; Christensen, R.; Khorshidi, A. *Phys. Chem. Chem. Phys. Phys. Chem. Chem. Phys* **2017**, *19*, 10978–10985.

(14) Musil, F.; Willatt, M. J.; Langovoy, M. A.; Ceriotti, M. *Journal of Chemical Theory and Computation* **2019**, *15*, 906–915.

(15) Tran, K.; Ulissi, Z. W. *Nature Catalysis* **2018**, *1*, 696–703.

(16) Tran, K.; Aini, P.; Back, S.; Ulissi, Z. W. *Journal of Chemical Information and Modeling* **2018**,

(17) Kresse, G.; Hafner, J. *Physical Review B* **1993**, *47*, 558–561.

(18) Kresse, G.; Hafner, J. *Physical Review B* **1994**, *49*, 14251–14269.

(19) Kresse, G.; Furthmüller, J. *Computational Materials Science* **1996**, *6*, 15–50.

(20) Kresse, G.; Furthmüller, J. *Physical Review B* **1996**, *54*, 11169–11186.

(21) Hjorth Larsen, A. et al. *Journal of Physics: Condensed Matter* **2017**, *29*, 273002.

(22) Hammer, B.; Hansen, L. B.; Nørskov, J. *Physical Review B* **1999**, *59*, 7413–7421.

(23) Thompson, S. K. In *Sampling*, 3rd ed.; Shewhart, W. A., Wilks, S. S., Eds.; John Wiley and Sons Inc., 2012; Chapter 11, pp 139–156.

(24) Xie, T.; Grossman, J. C. *Physical Review Letters* **2018**, *120*, 145301.

(25) Back, S.; Yoon, J.; Tian, N.; Zhong, W.; Tran, K.; Ulissi, Z. W. *The Journal of Physical Chemistry Letters* **2019**, *10*, 4401–4408.

(26) Bingham, E.; Chen, J. P.; Jankowiak, M.; Obermeyer, F.; Pradhan, N.; Karaletsos, T.; Singh, R.; Szerlip, P.; Horsfall, P.; Goodman, N. D. **2018**, 0–5.

(27) Gardner, J. R.; Pleiss, G.; Bindel, D.; Weinberger, K. Q.; Wilson, A. G. **2018**,

(28) Kuleshov, V.; Fenner, N.; Ermon, S. Accurate Uncertainties for Deep Learning Using Calibrated Regression. 35th International Conference on Machine Learning. Stockholm, Sweden, 2018.