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 used 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]

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 six 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 established a protocol for comparing the performance of different modeling and UQ methods.

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

Figure 1: Placeholder for overview of the paper

(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 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 valida-

tion/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.

Regression methods

Figure 2 illustrates all the methods we investigated in this study.

Crystal Graph Convolutional Neural Network (CGCNN): To establish a baseline for predictive accuracy, we re-trained a previously reported CGCNN[24,25] on this study's training set. This CGCNN 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, ΔE. Reference Back et al.[25] for additional details.

CGCNN Ensemble: We created an ensemble of CGCNNs by K-fold subsampling the training data into five different folds and then training individual CGCNN 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.

Bayesian Neural Network (BNN):

Supervised error prediction (delta CGCNN):

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

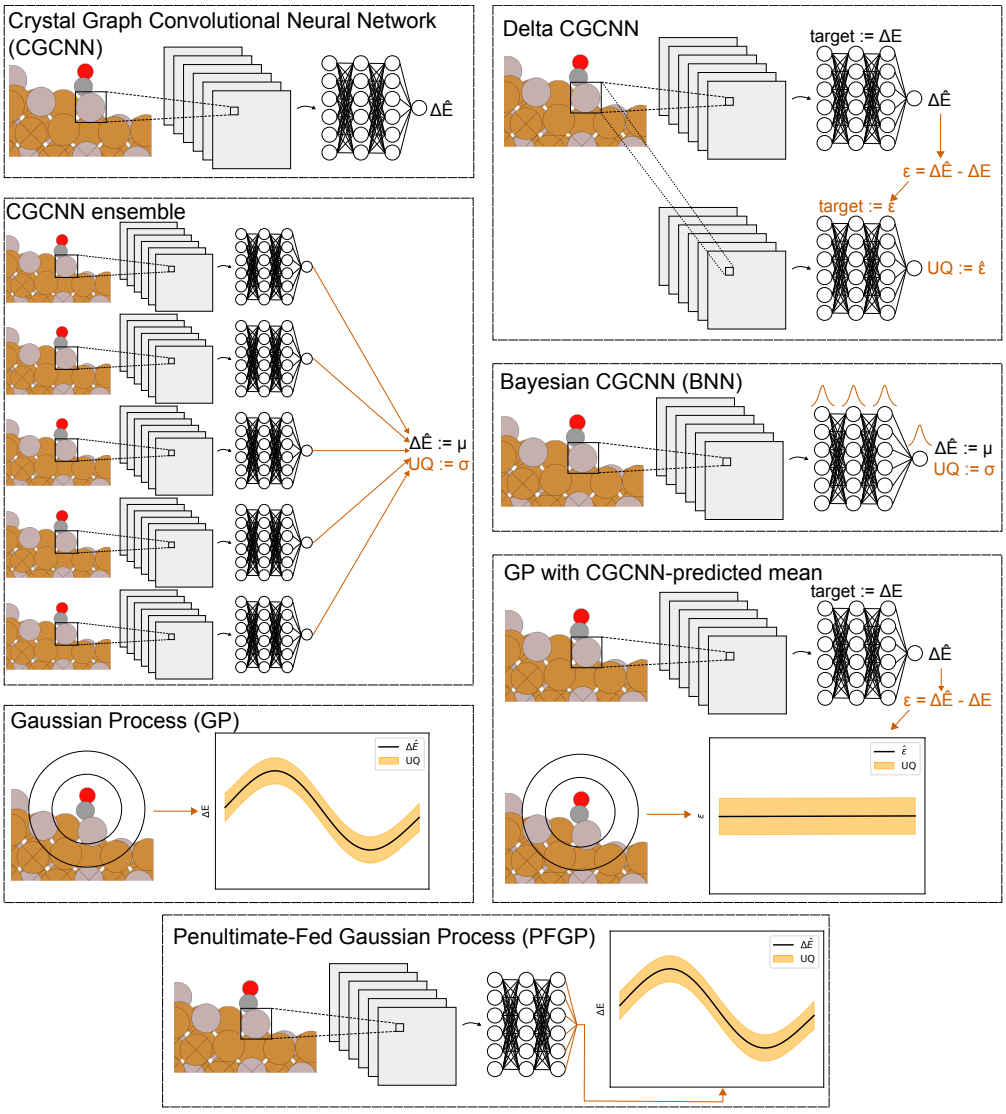

Figure 2: Overview of the various UQ methods we investigated in this study. ΔE represents DFT-calculated adsorption energies; $\Delta \hat{E}$ represents ML-predicted adsorption energies; UQ represents ML-predicted uncertainty quantifications; μ represents the mean of a sample of points; σ represents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

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 with a Gaussian likelihood. All GP training and predictions were done with GPU acceleration as implemented in GPyTorch.[26]

GP with CGCNN-predicted mean:

Penultimate-Fed Gaussian Process (PFGP): A limitation of using this formulation of a GP with CGCNN-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 PFGP. In this case, we used the baseline CGCNN as the network from which we obtained the penultimate outputs. We also normalized the penultimate outputs of the CGCNN 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 with a Gaussian likelihood. All GP training and predictions were done with GPU acceleration as implemented in GPyTorch.[26]

Performance metrics

We used six different metrics to quantify the accuracy of the various models: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Median Absolute Error (MDAE), Mean Absolute Relative Percent Difference (MARPD), R^2 correlation coefficient (R^2), and Pearson product-moment correlation coefficent (Pearson's). MARPD values were calculated with

Equation 1

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

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. 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, R^2, and Pearson's because they provide normalized measures of accuracy that may be more interpretable for those unfamiliar with adsorption energy measurements in eV. The ensemble of these metrics provide a more robust view of accuracy than any one metric can provide alone.

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.[27]. 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.[27] 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 esti-

mates. This idea of having small uncertainty estimates is called "sharpness", and Kuleshov et al.[27] 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 the magnitude of the sharpness values.

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

Results

1. Table/figure of metrics
2. Plots:
   (a) parity
   (b) calibration
   (c) sharpness
3. Blocking results on best candidates (maybe put in SI)
4. Cost of computing each method (if its there)
5. Human overhead and difficulty

Conclusions

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[26] 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) Gardner, J. R.; Pleiss, G.; Bindel, D.; Weinberger, K. Q.; Wilson, A. G. **2018**,

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