Supporting Information for Deep Graph Kernel Learning for Material & Atomic Level Uncertainty Quantification in Adsorption Energy Prediction

Osman Mamun, Chenlu Yang, Shuwen Yue

Contents

S1 Computational Environment

This section details the hardware and software infrastructure utilized for conducting the computational experiments presented in the main manuscript.

S1.1 Hardware Specifications

All computational tasks, including model training and evaluation, were executed on high-performance computing nodes. The primary specifications of these nodes are as follows:

- **CPU:** 2 x AMD EPYC 7713 64-Core Processors
- **GPU:** NVIDIA A100-SXM4-40GB with 40 GB of High Bandwidth Memory (HBM2)
- **RAM:** 384 GB DDR4 System Memory
- **Storage:** 2 TB NVMe Solid State Drive for fast data access

This hardware configuration provided substantial parallel processing capabilities, essential for training deep learning models and handling large datasets efficiently.

S1.2 Software Environment

The core software stack employed for this research comprised the following components and versions:

- **Operating System:** Ubuntu 20.04 LTS
- **CUDA Toolkit:** Version 12.4 (NVIDIA Driver Version: 550.54.14), enabling GPU acceleration.
- **Python:** Version 3.10.16
- **PyTorch:** Version 2.4.0, the primary deep learning framework.
- **PyTorch Geometric (PyG):** Version 1.7.2, extending PyTorch for graph neural network implementations.
  - Dependencies: torch-scatter 2.1.2, torch-sparse 0.6.18
- **GPyTorch:** Version 1.14, a library for scalable Gaussian process inference within PyTorch.
  - Dependency: linear-operator 0.6
- **Ray (Ray Tune):** Version 2.42.1, utilized for distributed hyperparameter optimization.

- **Optuna:** Version 4.2.0, employed as the optimization engine within Ray Tune.
- **Scikit-learn:** Version 1.6.1, used for standard machine learning utilities and metrics.
- **NumPy:** Version 1.26.4, for numerical operations.
- **Pandas:** Version 2.2.3, for data manipulation and analysis.
- **Atomic Simulation Environment (ASE):** Version 3.24.0, for handling atomic structures.

Dependency management was handled using Poetry to ensure reproducibility.

S2 Dataset Details

To evaluate the performance of our developed DGKL framework, we utilize two datasets: Catalysis Hub,[1,2] and Open Catalyst 2020 (OC20),[3] both of which were developed to facilitate high-throughput catalytic materials research.

The Catalysis Hub dataset contains approximately 37,000 entries pertaining to adsorption energies of CH, CH_2, CH_3, OH, NH, and SH on approximately 2,035 distinct bimetallic combinations of transition metals. This dataset provides a comprehensive benchmark for evaluating catalytic performance across a diverse range of metal combinations.

In contrast, the OC20 dataset comprises approximately 470,000 adsorption energy data points across a broad spectrum of transition metal and adsorbate combinations. Due to the computational intensity required to train models on the complete dataset, we employ a selected subset of 46,000 adsorption energy data points. This subset was constructed by selecting structures containing catalytically important transition metals: Ru, Re, Pt, Pd, Cu, Ni, Fe, Co, Rh, Ir, Mo, W, Au, Ag, Cr, Mn, Zn, Al, Ti, Zr, and V. We also refined our selection to include only structures with adsorbates containing carbon, hydrogen, and oxygen atoms.

For all model training procedures throughout this study, we consistently employed a 70:10:20 ratio for train, validation, and test sets, respectively, ensuring robust evaluation and comparison of model performance.

S3 Model Architecture Details

Graph Neural Networks (GNNs) provided the foundational framework for the uncertainty quantification (UQ) methodologies explored in this work. This section outlines the specific architectures implemented.

S3.1 SchNet

SchNet is an MPNN that maintains invariance to atom indexing and translation, thereby conforming to the physical principle that intensive properties of materials, such as adsorption energy, should remain unaffected by permutation of identical atoms, reordering of atomic positions, or translation of atomic coordinates. The energy predicted by SchNet is also rotationally invariant, adhering to fundamental physical symmetry constraints.

In the SchNet architecture, atoms' nuclear charges are initially embedded through an atom-type embedding layer. These embeddings are subsequently processed through a series of interaction blocks designed to model interatomic interactions. A key innovation in this architecture is the utilization of continuous filters rather than discrete ones to capture the radial dependency of interactions. After processing through multiple interaction blocks, the resulting representation passes through several dense layers, and finally, the energy is predicted using an invariant operation such as sum pooling or mean pooling.

Our implementation of the SchNet architecture utilizes continuous-filter convolutional layers for learning atomic representations. Key architectural hyperparameters, optimized as detailed in Section S4, included:

- **Hidden Channels / Number of Filters:** Varied between 32 and 256, determining the dimensionality of feature representations.
- **Number of Interaction Blocks:** Ranged from 2 to 6, controlling the depth of message passing.
- **Dense Layers within CFConv / Interaction Blocks:** Configured with 2 to 6 layers for internal transformations.
- **Number of Gaussian Basis Functions:** Set between 25 and 75 for encoding interatomic distances.
- **Global Pooling Operation (Readout):** Employed either Sum or Mean pooling to aggregate atomic features into a graph-level representation.
- **Activation Functions:** Explored Shifted Softplus (ssp), ReLU, tanh, GELU, and SiLU for introducing non-linearity.

S3.2 PaiNN

While SchNet demonstrates excellent performance in various material property prediction tasks, it can be less data-efficient compared to fully equivariant methods. A primary limitation of SchNet is its handling of rotational symmetry, specifically the lack of equivariant representation, which necessitates learning rotational equivariance explicitly through data augmentation. The Polarizable Atom Interaction Neural Network (PaiNN) addresses this limitation by explicitly leveraging the directional properties of relative positions between atoms.

Similar to SchNet, PaiNN processes atomic embeddings through several layers of message and update blocks. However, PaiNN computes and updates both

scalar and vector properties in a manner that preserves rotational symmetry. Subsequently, these embeddings are processed through dense layers followed by a pooling operation, analogous to the approach in SchNet, to predict the target properties.

The PaiNN architecture, incorporating equivariant message passing to preserve physical symmetries, was also implemented. Key architectural hyperparameters, optimized via HPO (Section S4), were:

- **Number of Interaction Layers:** Varied from 2 to 6, defining the network depth.
- **Hidden Channels:** Ranged from 32 to 256, controlling the feature vector size.
- **Number of Radial Basis Functions:** Set between 25 and 75 for the radial basis expansion.
- **Global Pooling Operation (Readout):** Utilized either Sum or Mean pooling for graph-level feature aggregation.
- **Activation Functions:** Explored Shifted Softplus (ssp), ReLU, tanh, GELU, and SiLU.

S3.3 Deep Graph Kernel Learning (DGKL) Implementation

The DGKL framework integrates a GNN feature extractor with a Sparse Variational Gaussian Process (SVGP) for uncertainty-aware predictions.

S3.3.1 Feature Extractor

Either the SchNet or PaiNN architecture served as the feature extractor. It was modified to output a latent vector (dimensionality ranging from 32 to 256) representing the entire input graph (material system). Layer normalization was typically applied to the final latent features before input to the SVGP to improve stability.

S3.3.2 Sparse Variational Gaussian Process (SVGP)

The SVGP component was applied to the latent representations generated by the GNN. Inducing points were employed to ensure computational scalability to large datasets. Key SVGP parameters included:

- **Kernel Function:** Utilized either the Radial Basis Function (RBF) or Matérn(5/2) kernel to model correlations in the latent space.
- **Number of Inducing Points:** Ranged from 32 to 512. Initial positions were set randomly or via k-means clustering on a subset of latent features, and subsequently learned during training.

- **Variational Distribution:** Employed either a Cholesky decomposition-based or a mean-field approximation for the variational posterior.
- **Variational Strategy:** Utilized the standard coupled inducing point strategy.
- **Likelihood Function:** Assumed a Gaussian likelihood for the adsorption energy predictions.
- **Objective Function:** Training maximized either the Predictive Log Likelihood (PLL) or the Evidence Lower Bound (ELBO).
- **Feature Scaling:** Latent features were optionally scaled (e.g., to the range [-1, 1]) before input to the SVGP.

S3.4 DGKL-Atomic Implementation

This novel variant adapts the DGKL framework to provide atomic-level uncertainty estimates, offering finer-grained insights into prediction confidence.

- The GNN feature extractor was modified to produce *atomic-level* latent feature vectors, deferring the global pooling step.
- The SVGP operated directly on these atomic latent features, predicting per-atom means ($\mu_{\text{atomic},i}$) and variances ($\sigma^2_{\text{atomic},i}$).
- Material-level predictions were subsequently obtained by aggregating the atomic contributions. Assuming approximate independence between atomic contributions for variance aggregation:
  - Mean Prediction: $\mu_{\text{material}} = \sum_{i \in \text{graph}} \mu_{\text{atomic},i}$
  - Variance Prediction: $\sigma^2_{\text{material}} \approx \sum_{i \in \text{graph}} \sigma^2_{\text{atomic},i}$

This approach allows uncertainty to be attributed to specific atoms or regions within the material system.

In table **??**, we show the effect of conditioning on the whole covariance matrix vs. only the atomic uncertainties,

S4 Hyperparameter Optimization (HPO)

Systematic hyperparameter optimization was performed to identify effective configurations for the GNN architectures used within the Ensemble and MCD frameworks.

Table 1: Effect of atomic uncertainty covariance conditioning on CatHub SchNet predictions, averaged over five runs.

Method	**MAE (eV)** [↓]	**NLL** [↓]	**ENCE** [↓]	**Miscal. Area** [↓]	$\rho^{\star}_{\mathbf{SCC}}$ [↑]	**ROC AUC** [↑]
DGKL (Full Cov.)	0.35 ± 0.02	0.78 ± 0.12	0.42 ± 0.07	0.36 ± 0.05	0.38 ± 0.01	0.68 ± 0.01
DGKL (Diag. Cov.)	0.43 ± 0.04	**0.68 ± 0.09**	**0.07 ± 0.04**	**0.04 ± 0.02**	**0.51 ± 0.02**	**0.74 ± 0.01**

Notes:

[↑]: Higher is better. [↓]: Lower is better.

Bold values indicate the better result for each metric.

Full Cov.: full atomic covariance matrix; Diag. Cov.: diagonal elements only.

[⋆] SCC = Spearman Correlation Coefficient

S4.1 HPO Strategy

Hyperparameter optimization for the SchNet and PaiNN backbones was conducted using the Ray Tune library, leveraging the Optuna search algorithm for efficient exploration and the Asynchronous Successive Halving Algorithm (ASHA) for aggressive early stopping of unpromising trials.

- **Search Space Definition:** A comprehensive search space was defined, encompassing key architectural and training parameters. These included batch size, hidden channel dimensionality, number of interaction/filter layers, number of dense layers within blocks, number of radial/Gaussian basis functions, readout function (Sum/Mean), activation function, optimizer type (e.g., Adam, AdamW), learning rate scheduler type (e.g., ReduceLROnPlateau, CosineAnnealingLR), initial learning rate, and dropout probability (specifically for MCD models). The precise ranges explored are detailed in the accompanying source code repository.
- **Optimization Configuration:** The HPO process was configured as follows:
  - **Number of Trials:** 25 independent trials were executed for each combination of UQ method, dataset, and GNN architecture.
  - **Maximum Epochs per Trial:** Each trial was run for a maximum of 25 epochs.
  - **ASHA Scheduler Parameters:** Configured with a grace period (minimum epochs before stopping) of 10 epochs (or $\lfloor \text{max\_epochs}/2 \rfloor$) and a reduction factor of 2, promoting efficient resource allocation.
  - **Optimization Objective:** The primary objective was to minimize the validation loss (`metric="val_loss"`, `mode="min"`).
  - **Resource Allocation:** Each trial was typically allocated 4 CPU cores and 1 NVIDIA A100 GPU.
  - **Reproducibility:** Unique random seeds were utilized for each trial initialization to ensure reproducibility.
- **Output and Selection:** Upon completion of the HPO process, the hyperparameter configuration yielding the lowest validation loss was selected for the final training of single models (MCD, Evidential). For the Ensemble method, the top-K (where K=5 in this study) best-performing configurations based on validation loss were selected.

Note on DGKL/DGKL-Atomic HPO: Due to the increased complexity and computational cost associated with jointly optimizing the parameters of both the GNN feature extractor and the SVGP component, a full HPO search was not performed for the DGKL and DGKL-Atomic models. Instead, hyperparameters for these models were carefully selected based on prior experience and findings from preliminary experiments, as specified in the respective implementation scripts available in the code repository.

S5 Uncertainty Quantification Models

S5.1 Ensemble

The ensemble method is a widely used UQ technique across various molecular and material property prediction applications. It represents a simple yet reliable approach to UQ, as demonstrated by several benchmark studies[4–7]. Due to its consistent performance in ensembling/query-by-committee frameworks, it is frequently referred to as the 'gold standard' for uncertainty quantification.

Ensemble methods capture model uncertainty (epistemic uncertainty) by employing varied model architectures or different parameter initializations, allowing models to converge to distinct local minima on the energy landscape. The average prediction across these trained models can be conceptualized through a statistical mechanics analogy to ensemble averaging. The standard deviation among predictions from this ensemble provides a straightforward and effective means of quantifying uncertainty. Through full or partial exploration of the loss landscape, ensemble methods demonstrate robustness to noisy and out-of-distribution test samples[8].

While this approach offers simplicity and effectiveness, it presents two notable limitations: (1) the computational cost scales linearly with the number of constituent models, resulting in compute-intensive training, and (2) inference requires each test sample to be evaluated by every model in the ensemble, reducing inference speed.

For our ensemble of graph models, we conducted hyperparameter optimization across various model configurations, including learning rate, batch size, number of hidden channels, and number of layers, using the open-source Ray Tune library[9]. The five best-performing model configurations were selected for the final ensemble. Importantly, we used identical training and validation sets for hyperparameter optimization and final model training to prevent data leakage to the test set.

S5.2 Monte Carlo Dropout

Monte Carlo (MC) dropout represents a computationally efficient alternative to traditional ensemble approaches, enabling uncertainty quantification through a single model within a probabilistic framework. MC dropout was first formalized by Gal & Ghahramani[10] in their seminal 2016 paper, which established that applying dropout-probabilistically deactivating connections to different nodes during neural network training-can be interpreted as approximate Bayesian inference in deep Gaussian processes.

This training approach serves as an effective regularization technique, as it prevents the model from relying on any specific set of connections for making predictions. During standard inference, the weighted average of node predictions based on their assigned probabilities provides the mean estimation. However, for uncertainty quantification, one can generate multiple predictions by maintaining active dropout during inference time. These stochastic forward passes yield a

distribution of predictions from which standard deviation can be calculated, providing a measure of model uncertainty.

Conceptually, this process parallels the ensemble approach, as each stochastic forward pass effectively samples from a slightly different neural network architecture. The key advantage lies in computational efficiency: training requires only the resources needed for a single graph model, though inference still necessitates multiple forward passes to generate the prediction distribution.

For our implementation, we performed hyperparameter optimization over the same parameter space as used for the ensemble models, with the addition of dropout probability as an optimization parameter. After training the optimal model, we executed 10 stochastic forward passes per input to obtain robust uncertainty estimations.

S5.3 Evidential Regression

In contrast to sampling-based methods such as ensemble and Monte Carlo dropout, evidential regression represents a sampling-free approach to uncertainty quantification. Rather than directly parameterizing the target variable, evidential regression parameterizes an evidential distribution over the target. For our adsorption energy prediction task, the network is trained to learn the parameters of a Normal-Inverse-Gamma (NIG) distribution[11–13].

The loss function for evidential regression comprises two components: (1) the NIG loss, which encourages accurate prediction of the target distribution, and (2) a regularization loss that penalizes overconfident predictions, particularly for out-of-distribution samples. Given the learned NIG distribution parameters, we compute the epistemic uncertainty as:

$$\sigma^2_{\text{epistemic}} = \frac{\beta}{\nu(\alpha - 1)} \tag{1}$$

where α, β and ν are NIG distribution parameters learned during training. The parameter α represents the degrees of freedom (shape parameter) and must satisfy $\alpha > 1$ for the variance to be properly defined, while $\beta > 0$ serves as the scale parameter of the distribution.

Consistent with our approach for ensemble methods, we performed hyperparameter optimization for evidential regression using the same parameter space as applied to the ensemble models to ensure fair comparison across methods.

S6 Model Training and Evaluation Specifics

This section details the protocols used for training the various UQ models and evaluating their performance.

S6.1 Training Protocol and Data Splitting

All datasets (CatHub, OC20) were consistently partitioned into training (70%), validation (10%), and test (20%) sets using a fixed random seed for reproducibility across experiments. Models were trained for a predefined maximum number of epochs (`MAX_EPOCHS`), typically around 100 for Ensemble, MCD, and Evidential models, and between 200 and 400 for the DGKL variants, which often require longer convergence times. Early stopping was employed, monitoring the validation loss and terminating training if no improvement was observed for a specified patience window (typically 5 epochs), restoring the model weights from the epoch with the best validation performance.

S6.2 Ensemble Models

- **Training:** The top-K (K=5) distinct hyperparameter configurations identified via HPO (Section S4) were used to train K independent GNN models from different random initializations for the full `MAX_EPOCHS` (early stopping applied individually).
- **Evaluation:** For a given input sample, predictions were obtained from all K ensemble members. The final predictive mean (μ) was calculated as the average of the K individual predictions. The predictive uncertainty, quantified by the standard deviation (σ), was calculated as the standard deviation of the K predictions. This captures model uncertainty arising from different initializations and potentially different hyperparameter settings.
- **Loss Function:** Each individual model was trained by minimizing the Huber loss, providing robustness to outliers compared to standard Mean Squared Error.

S6.3 Monte Carlo Dropout (MCD) Models

- **Training:** The single best GNN model configuration identified via HPO was trained for `MAX_EPOCHS` with dropout layers activated. The dropout rate was determined during HPO and typically applied to the final dense layers before the output prediction.
- **Evaluation:** Crucially, dropout was kept active during the evaluation phase. For each input sample, multiple stochastic forward passes (e.g., $T = 10$ passes) were performed through the network. Due to the random nature of dropout, each pass yielded a slightly different prediction.
- **Prediction and Uncertainty:** The final predictive mean (μ) was taken as the average of the predictions from the T forward passes. The predictive uncertainty (σ) was calculated as the standard deviation of these T predictions, approximating the model's uncertainty.
- **Loss Function:** Huber loss was used during training.

S6.4 Evidential Deep Learning Models

- **Training:** The single best GNN model configuration from HPO was adapted for evidential regression. The network output layer was modified to predict the four parameters of an evidential Normal-Inverse-Gamma (NIG) distribution $(\gamma, \nu, \alpha, \beta)$ instead of a single point estimate. The model was trained for `MAX_EPOCHS`.
- **Evaluation:** The trained model directly outputs the parameters $(\gamma, \nu, \alpha, \beta)$ for each input.
- **Prediction and Uncertainty:** The predictive mean (μ) corresponds to the location parameter γ. The predictive variance (σ^2) is calculated from the NIG parameters as $\sigma^2 = \frac{\beta}{\nu(\alpha-1)}$. This formulation allows the model to explicitly learn uncertainty estimates.
- **Loss Function:** Training minimized a combination of the NIG negative log-likelihood (NLL) loss and an evidence regularizer term, encouraging the model to produce well-calibrated uncertainty estimates.

S6.5 Deep Graph Kernel Learning (DGKL/DGKL-Atomic) Models

- **Training:** A specific configuration defining both the GNN feature extractor (SchNet or PaiNN) and the SVGP parameters was employed (details in the code repository). The GNN and SVGP components were trained end-to-end.
- **Optimization Details:** Due to the nature of joint GNN-GP training, specific optimization strategies were crucial (see Section S8). This often involved using distinct learning rates for the GNN and SVGP parameters (typically a smaller learning rate for the pre-trained GNN backbone) and applying gradient clipping to prevent numerical instability.
- **Objective Function (Loss):** The training process aimed to maximize either the Predictive Log Likelihood (PLL) or the Evidence Lower Bound (ELBO), which are standard objectives for variational Gaussian processes.
- **Evaluation:** The trained DGKL model (comprising the `feature_extractor`, `svgp`, and `likelihood` components) was used in evaluation mode (`model.eval()`) to obtain the predictive mean (μ) and standard deviation (σ) directly from the SVGP posterior distribution for each input graph. For DGKL-Atomic, atomic-level means and variances were predicted and subsequently aggregated to the material level as described in Section S3.

S7 Evaluation Metrics

To evaluate the performance of uncertainty quantification methods, several metrics have been developed in the literature. These metrics often provide different and sometimes inconsistent insights about the performance of uncertainty quantification models.[14] Broadly, the evaluation metrics for UQ can be divided into two classes: (1) error-based approaches and (2) interval-based approaches.

Error-based approaches directly compare the predicted uncertainty to the observed error, providing a direct assessment of how well the uncertainty estimates correspond to actual prediction errors. In contrast, interval-based approaches examine how the observed errors are distributed within intervals of predicted uncertainties compared to ideal statistical behavior, such as evaluating whether approximately 68% of observations fall within $\pm 1\sigma$ of predictions as would be expected for a Gaussian distribution.

Below, we briefly discuss the some important evaluation metrics used in this work to assess the performance of our uncertainty quantification methods comprehensively.

S7.1 Negative Log-Likelihood (NLL)

Negative log-likelihood, an error-based approach, is a standard metric used to evaluate the performance of uncertainty quantification models, which can also be used as a loss function to minimize during training. It measures the likelihood of observing the true values given the predicted distribution, where the underlying assumption is that the prediction errors are described by a Gaussian distribution with predicted variances (σ^2). NLL accounts for both the error and the variance to measure the quality of uncertainty quantification. A lower NLL indicates better uncertainty quantification performance.

$$\text{NLL} = \frac{1}{2N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left[\ln(2\pi) + \ln(\sigma_i^2) + \frac{(y_i - \mu_i)^2}{\sigma_i^2} \right] \tag{2}$$

where μ_i and σ_i are the predicted mean and variance, respectively, and y_i is the true value.

For distributions with identical error magnitudes but different predicted variances, the uncertainties more representative of the actual error distribution will yield a lower NLL. It is important to note that a more accurate model (i.e., one having a lower mean absolute error) will generally result in a lower NLL, even if the uncertainties are poorly calibrated.[15] Furthermore, NLL is a relative metric, meaning its value is most meaningful when compared across different models. Due to these considerations, while a lower NLL indicates better uncertainty quantification performance, it is not sufficient in isolation to conclusively establish that a model is superior at quantifying uncertainty.

S7.2 Expected Normalized Calibration Error (ENCE)

The Expected Normalized Calibration Error (ENCE)—an interval-based approach—quantifies the deviation of a model's calibration from ideal behavior in the Root Mean Square Error (RMSE) versus Root Mean Variance (RMV) plot. In traditional error-based calibration approaches, comparing models based on the area between the ideal calibration curve and the model's calibration curve can lead to inconsistent comparisons, as different models often exhibit varying ranges of RMSE and RMV values that are not bounded within a standardized interval such as [0, 1].

To facilitate meaningful cross-model comparison, ENCE calculates the relative deviation by taking the mean absolute difference between the RMSE and RMV for each bin, normalized by the RMV[16]:

$$\text{ENCE} = \frac{1}{N_{\text{bin}}} \sum_{i=1}^{N_{\text{bin}}} \frac{|\text{RMSE}_i - \text{RMV}_i|}{\text{RMV}_i} \tag{3}$$

where RMSE_i and RMV_i are the RMSE and RMV for the i-th bin, respectively, and N_{bin} is the total number of bins used in the calibration analysis.

When comparing multiple uncertainty quantification models, the one with lower ENCE is preferred as it demonstrates smaller deviations from the ideal calibration behavior in the RMSE versus RMV relationship. A perfectly calibrated model would have an ENCE of 0, indicating that the predicted variances (RMV) precisely match the observed squared errors (RMSE) across all uncertainty bins.

S7.3 Miscalibration Area

The miscalibration area, another interval-based approach, quantifies the deviation between the expected and observed distributions of errors within confidence intervals. Specifically, it assesses how much the expected fraction of errors within each confidence interval differs from the observed fraction of errors that actually fall within those intervals.

This metric is computed as the absolute difference between the expected fraction of errors and the observed fraction of errors, effectively measuring the area between the diagonal line (representing perfect calibration) and the empirical calibration curve. The interpretation of this deviation has directional meaning: if the model's observed fraction of errors lies above the diagonal line, it indicates underconfident predictions (the model overestimates its uncertainty); conversely if the observed fraction lies below the diagonal, it indicates overconfident predictions (the model underestimates its uncertainty).

$$\text{Miscalibration Area} = \frac{1}{N_{\text{bin}}} \sum_{i=1}^{N_{\text{bin}}} |\text{Expected Fraction}_i - \text{Observed Fraction}_i| \tag{4}$$

where Expected Fraction$_i$ and Observed Fraction$_i$ are the expected and observed fractions of errors for the i-th confidence interval, respectively. This discrete summation approximates the continuous integral of differences across all possible confidence levels.

A smaller miscalibration area is desirable, as it indicates minimal deviation from ideal calibration. A perfectly calibrated model would have a miscalibration area of 0, signifying that across all confidence intervals, the expected fractions precisely match the observed fractions of errors.

S7.4 Reliability Diagram

A well-calibrated uncertainty quantification model should exhibit a one-to-one relationship between predicted variances and observed squared errors, meaning higher prediction errors should be reflected by correspondingly higher uncertainty estimates.[17] To obtain localized information about this relationship, the predicted uncertainties are ordered and divided into equal-sized bins. For each bin, the root mean square error (RMSE) and root mean variance (RMV) are computed as:

$$\text{RMSE} = \sqrt{\frac{1}{N_{\text{bin},i}} \sum_{j \in \text{bin}_i} (y_j - \mu_j)^2} \tag{5}$$

$$\text{RMV} = \sqrt{\frac{1}{N_{\text{bin},i}} \sum_{j \in \text{bin}_i} \sigma_j^2} \tag{6}$$

where $N_{\text{bin},i}$ represents the number of samples in the i-th bin, y_j is the true value, μ_j is the predicted mean, and σ_j is the predicted standard deviation for the j-th sample in that bin.

The reliability diagram plots RMSE against RMV for each bin. For a perfectly calibrated model, this plot should follow a straight line with a slope of 1 and an intercept of 0, indicating that the predicted uncertainties across all uncertainty levels accurately capture the magnitude of prediction errors.

S7.5 Calibration Plot

The calibration plot provides a visual representation of how well the uncertainty estimates constitute the true correctness likelihood of predictions.[18] This plot compares the observed fraction of errors against the expected fraction of errors within specified confidence intervals.

To construct this plot, prediction errors are first normalized into z-scores using the predicted uncertainties:

$$z_i = \frac{y_i - \mu_i}{\sigma_i} \tag{7}$$

where y_i is the true value, μ_i is the predicted mean, and σ_i is the predicted standard deviation for the i-th sample.

The expected fraction of errors is then computed as the cumulative distribution function (CDF) of the standard normal distribution evaluated at different confidence levels.

For a well-calibrated model, the calibration plot should closely follow the diagonal line, indicating that the observed fractions of errors align with the theoretically expected fractions across all confidence levels. Deviations above the diagonal suggest underconfidence (the model overestimates uncertainty), while deviations below indicate overconfidence (the model underestimates uncertainty).

S7.6 Spearman's Correlation Coefficient

Spearman's correlation coefficient (ρ) measures the monotonic relationship between predicted uncertainty and observed prediction errors. This error-based metric evaluates the model's ability to rank errors correctly from low to high values, regardless of the absolute magnitudes of the uncertainties.

The underlying premise is that data points with lower predicted uncertainties should generally correspond to lower observed errors, and conversely, higher predicted uncertainties should correlate with larger errors. Mathematically, Spearman's correlation coefficient is computed as the Pearson correlation between the rank variables of predicted uncertainties and observed absolute errors:

$$\rho = \frac{\text{cov}(R(|\hat{y} - y|), R(\sigma))}{\sigma_{R(|\hat{y}-y|)}\sigma_{R(\sigma)}} \tag{8}$$

where $R(|\hat{y} - y|)$ represents the ranks of absolute prediction errors, $R(\sigma)$ represents the ranks of predicted standard deviations, cov is the covariance, and $\sigma_{R(\cdot)}$ denotes the standard deviations of the rank variables.

While a higher ρ value (closer to 1) indicates better uncertainty quantification, it is important to note that perfect correlation ($\rho = 1$) is highly unlikely in practice. This is because even predictions with high uncertainty estimates can occasionally yield small errors due to the inherent stochasticity in the prediction process. A value of ρ substantially greater than zero indicates that the uncertainty estimates provide meaningful information about the expected magnitude of errors.

S7.7 Coverage

Coverage at different confidence intervals quantifies the proportion of true values that fall within specified prediction intervals. This interval-based metric is computed as:

$$\text{Coverage} = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \mathbf{1}_{y_i \in \text{CI}(x_i)} \tag{9}$$

where $\mathbf{1}_{y_i \in \mathrm{CI}(x_i)}$ is an indicator function that equals one if the true value y_i falls within the confidence interval $\mathrm{CI}(x_i)$ for input x_i, and zero otherwise. N_{test} represents the total number of test samples.

Coverage measures the model's empirical reliability by indicating the fraction of true values captured within the predicted confidence intervals [19]. For a well-calibrated model assuming Gaussian error distributions, approximately 68% of the data should fall within $\pm 1\sigma$ (one standard deviation) intervals, and approximately 95% should fall within $\pm 2\sigma$ intervals.

The deviation between the theoretical and observed coverage rates provides valuable insights into whether a model is overconfident (observed coverage lower than theoretical) or underconfident (observed coverage higher than theoretical). Ideally, the observed coverage should closely match the theoretical coverage across various confidence levels, indicating proper uncertainty calibration.

S7.8 ROC-AUC

The Area Under the Receiver Operating Characteristic curve (ROC-AUC) is an error-based approach that evaluates how well the predicted uncertainties discriminate between high and low prediction errors. This metric converts the continuous prediction errors into binary classifications based on a threshold value—in this work, we use the median of the observed errors as the threshold.

The ROC curve is constructed by plotting the true positive rate (sensitivity) against the false positive rate (1-specificity) at various uncertainty thresholds. Specifically, for each possible threshold on the predicted uncertainty:

- Predictions with uncertainties above the threshold with errors above the error median are counted as true positives.
- Predictions with uncertainties above the threshold with errors below the error median are counted as false positives.

The AUC (Area Under the Curve) value ranges from 0 to 1, where a value of 0.5 indicates random performance (uncertainties have no discriminative power for identifying large errors), and a value of 1 indicates perfect discrimination (uncertainties perfectly separate high and low errors). The ROC-AUC thus quantifies the model's ability to use predicted uncertainties as reliable indicators of potential prediction errors.

S8 DGKL Training Challenges and Mitigation Strategies

Training the DGKL and DGKL-Atomic models, which involve the joint optimization of deep GNN feature extractors and variational Gaussian processes, presented specific numerical and optimization challenges. These were addressed through the following mitigation strategies:

- **Mode Collapse Prevention:** To prevent the GNN feature extractor from collapsing due to uneven scaling, feature normalization techniques, particularly Layer Normalization applied to the GNN output features, were employed. This helps maintain diverse and informative latent representations for the SVGP.
- **Numerical Stability Enhancement:** Several techniques were combined to ensure stable training:
  - **Gradient Clipping:** Gradients were clipped to a maximum norm (e.g., 1.0 or 20.0, as specified in training scripts) to prevent exploding gradients, particularly during the initial stages of training or when encountering difficult samples.
  - **Kernel Matrix Jitter:** A small amount of jitter (1×10^{-6}) was added to the diagonal of kernel matrices within the SVGP calculations. This improves the numerical conditioning of the matrices, preventing issues during Cholesky decomposition or matrix inversion.
  - **Differential Learning Rates:** Distinct learning rates were often used for the GNN backbone and the SVGP parameters. Typically, the GNN (potentially pre-trained) used a significantly smaller learning rate (e.g., 100x smaller) than the SVGP parameters, allowing for fine-tuning of the feature extractor while the GP adapts more quickly.
  - **Adaptive Learning Rate Adjustments:** Training loops included checks for NaN (Not a Number) values in the loss or gradients. If detected, the optimizer state was rolled back, and the learning rate was reduced before retrying the step, preventing training divergence due to numerical overflows.

These strategies collectively contributed to the successful and stable training of the complex DGKL models.

S9 Supporting Parity Plots of Predictive Accuracy

Figure 1 provides a comprehensive visual comparison of the point prediction accuracy achieved by each uncertainty quantification (UQ) method across all evaluated dataset-GNN backbone combinations. These parity plots display the model-predicted adsorption energies versus the true Density Functional Theory (DFT) calculated values. Ideally, for a perfect model, all data points would lie on the $y = x$ diagonal. The scatter around this diagonal visually represents the prediction error.

Each panel in Figure 1 includes the Mean Absolute Error (MAE) and the coefficient of determination (R^2) to quantify the predictive performance. The MAE (in eV) indicates the average magnitude of the prediction errors, with lower values being better. The R^2 value (unitless, ranging from 0 to 1) represents the proportion of the variance in the true adsorption energies that is predictable from the model's predictions, with values closer to 1 indicating a better fit. The color intensity of the data points is scaled by the absolute prediction error, providing a quick visual guide to where larger errors occur.

As discussed in the main manuscript (Section 2.1, **Predictive Accuracy**), these plots generally show that Ensemble and Evidential methods achieve lower MAE and higher R^2 values, indicating superior point prediction accuracy. The DGKL and DGKL-Atomic methods often exhibit slightly higher MAEs, reflecting the trade-off inherent in their optimization objectives, which prioritize uncertainty calibration alongside accuracy. The plots for the OC20-SchNet combination (rightmost column) typically show greater scatter and lower R^2 values across all methods compared to the CatHub dataset, underscoring the increased challenge of predicting adsorption energies on this more diverse dataset.

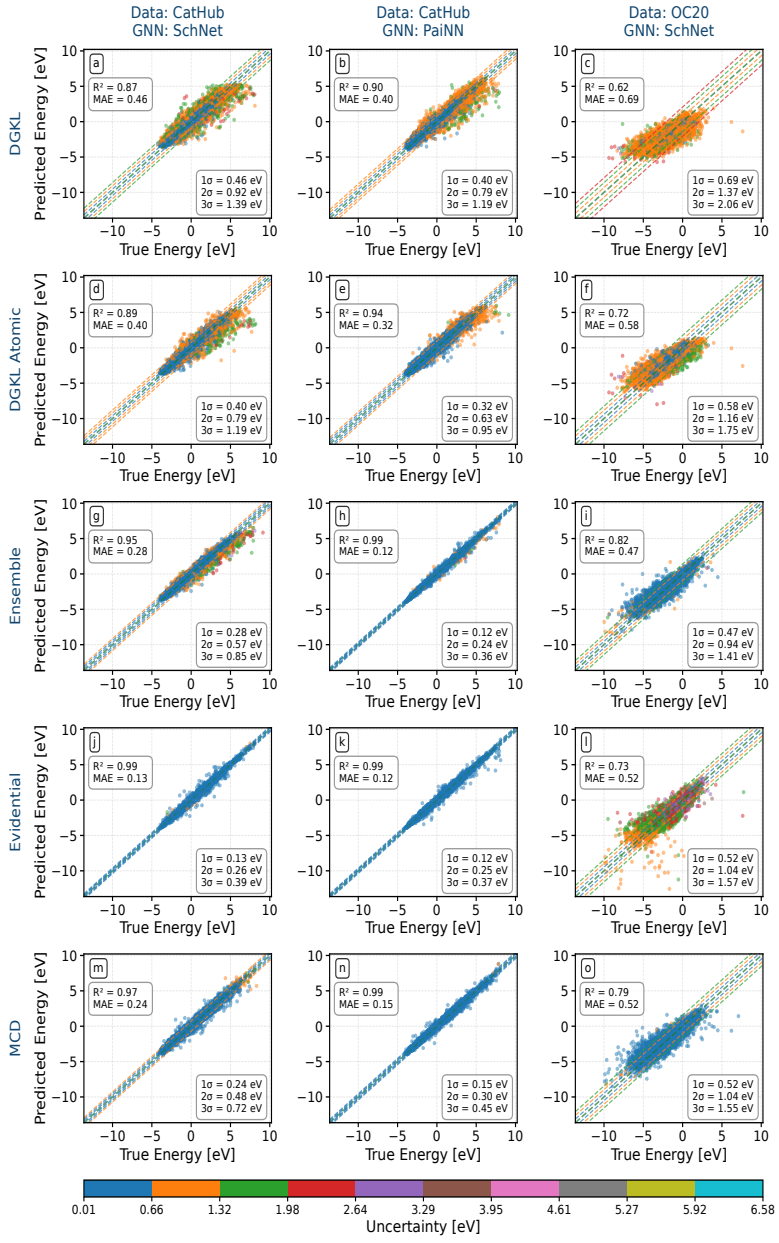

Figure 1: Comparative analysis of prediction accuracy for various GNN-based uncertainty quantification methods applied to adsorption energy prediction. The color intensity of data points corresponds to the magnitude of the absolute prediction error, as shown by the color bar. We also show the 1, 2, and 3 σ errors with diagonal lines around the parity line with the same color as the corresponding predicted uncertainty. Ideally, the points around these lines should have the same color as that line.

S10 Supporting Error versus Uncertainty Plots

Figure 2 presents a detailed visual analysis of uncertainty calibration for all evaluated UQ methods and dataset-GNN backbone combinations. These plots directly compare the magnitude of the absolute prediction error ($|\varepsilon| = |E_{\text{pred}} - E_{\text{true}}|$) against the model's predicted uncertainty for that prediction (quantified by the predicted standard deviation, σ). Both axes are in units of eV.

For a well-calibrated model, we expect a positive correlation: predictions with higher assigned uncertainty (σ) should, on average, exhibit larger absolute errors ($|\varepsilon|$). The dashed lines on each plot represent ideal calibration targets:

- The 1σ line ($|\varepsilon| = \sigma$): Approximately 68% of data points should ideally fall below this line if the errors are Gaussian and the uncertainty is perfectly calibrated.
- The 2σ line ($|\varepsilon| = 2\sigma$): Approximately 95% of data points should ideally fall below this line.

Points lying far above these lines, especially above the 2σ line, indicate instances where the model was overconfident (i.e., the actual error was much larger than the predicted uncertainty). Conversely, points far below might suggest underconfidence if the uncertainty is excessively large for small errors, though this is less common for the problematic cases. The density of points (log scale) indicates the prevalence of certain error-uncertainty pairings.

As discussed in the main manuscript (Section 2.1, **Error vs. Uncertainty Visualization**), the DGKL and DGKL-Atomic methods (top two rows) generally exhibit the most desirable behavior. Their data points show a clearer positive correlation between $|\varepsilon|$ and σ, and the uncertainties span a wider dynamic range, allowing the model to express varying degrees of confidence that align better with observed errors. This means that when DGKL predicts a large σ, it is a more reliable warning of a potentially large error.

In contrast, the Ensemble, Evidential, and particularly the MCD methods often show predicted uncertainties clustered at low σ values, even for predictions with large absolute errors. This indicates significant overconfidence. For instance, many points with high $|\varepsilon|$ (e.g., > 0.5 eV) may have a predicted $\sigma < 0.1$ eV for these methods, especially for MCD. Such behavior undermines the utility of the uncertainty estimates for practical decision-making in catalyst design, as a catalysis scientist cannot reliably trust a low predicted σ to mean a low actual error. These comprehensive plots across all configurations in Figure 2 provide strong visual support for the superior calibration quality of the DGKL framework discussed quantitatively in the main text.

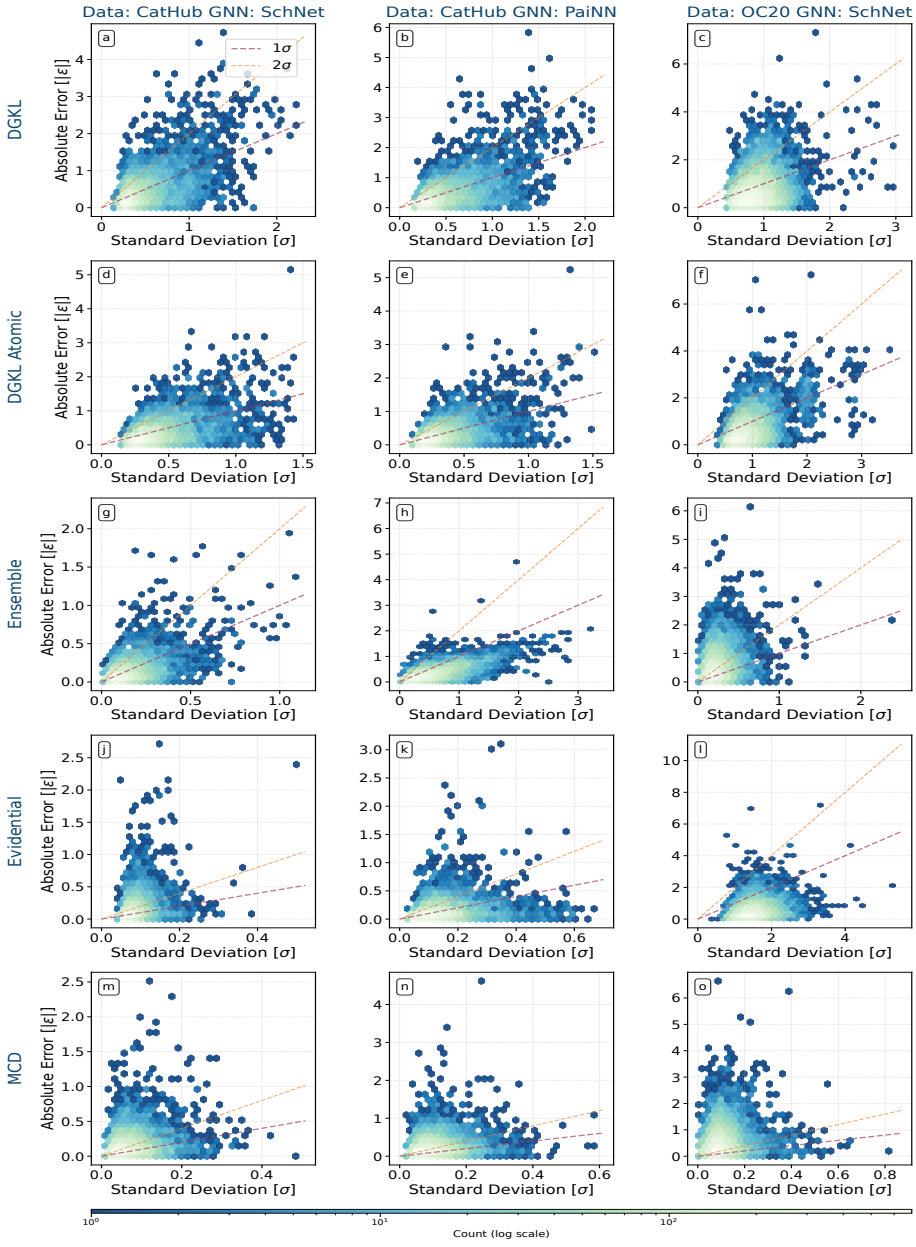

Figure 2: Assessment of uncertainty calibration quality across different UQ methods and model configurations. Each plot displays the absolute prediction error ($|\varepsilon|$) as a function of the predicted standard deviation (σ). Rows correspond to the UQ methodologies: (Top to Bottom) DGKL, DGKL Atomic, Ensemble, Evidential, and MCD. Columns represent the dataset-GNN combinations: (Left Column) CatHub-SchNet, (Middle Column) CatHub-PaiNN, (Right Column) OC20-SchNet. The dashed lines indicate perfect calibration targets, where absolute error equals the predicted standard deviation (1σ) and twice the standard deviation (2σ). The color intensity reflects the density of data points on a logarithmic scale, highlighting regions with high concentrations of predictions.

S11 Out-of-Distribution Detection Performance Analysis for DGKL-Atomic

Figure 3 presents a comprehensive evaluation of out-of-distribution detection performance through receiver operating characteristic (ROC) analysis across three distinct OOD scenarios: catalysis systems (OOD Cat), adsorption systems (OOD Ads), and their combination (OOD Both) for DGKL-Atomic. The analysis is conducted under two experimental conditions to elucidate the impact of structural similarity on detection efficacy. Due to the substantial computational expense associated with the computation of SOAP vectors, we restrict this analysis to randomly selected 400 molecules from each dataset, namely the test, OOD Cat, OOD Ads, and OOD Both datasets.

The left panel demonstrates baseline OOD detection performance when considering all atomic environments within the test datasets. Under these conditions, the model achieves moderate discrimination capability with area under the curve (AUC) values ranging from 0.756 to 0.763 across all three OOD categories. The relatively modest performance and overlapping confidence intervals suggest inherent challenges in distinguishing between in-distribution and out-of-distribution samples when structural heterogeneity is present within the OOD datasets.

The right panel reveals a substantial enhancement in detection performance following the exclusion of atoms exhibiting high structural similarity to in-distribution environments (SOAP cosine similarity ≤ 0.8). This filtering strategy yields remarkable improvements, with AUC values increasing to 0.902–0.907, representing an improvement of approximately 0.14–0.15 across all OOD categories. The steep initial rise in the ROC curves and sustained high true positive rates at low false positive rates indicate robust discrimination between genuinely novel atomic environments and familiar structural motifs.

The consistent performance enhancement across all three OOD scenarios validates our hypothesis that the bimodal uncertainty distribution observed in Figure 4d in the main text stems from the presence of structurally familiar atomic environments within nominally out-of-distribution materials. These results demonstrate that uncertainty-based OOD detection achieves near-optimal performance when applied to truly novel atomic environments, while highlighting the importance of considering local structural similarity in the interpretation of model predictions for materials discovery applications.

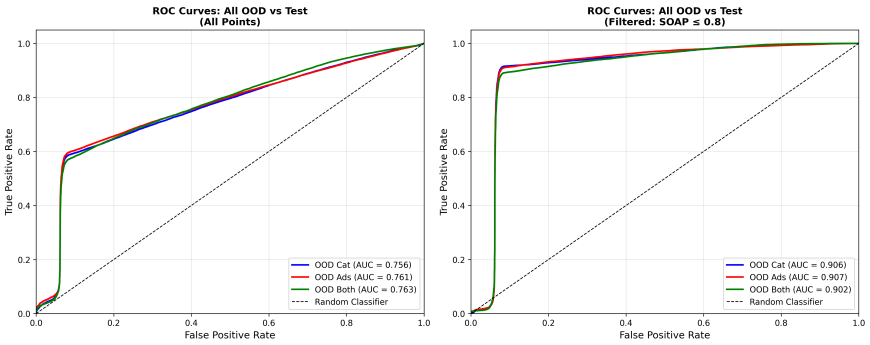

Figure 3: **Receiver operating characteristic (ROC) curves for out-of-distribution detection performance. (Left)** Baseline OOD detection performance considering all atomic environments in the test datasets, showing moderate discrimination capability with AUC values of 0.756–0.763 across three OOD scenarios: catalysis systems (OOD Cat, blue), adsorption systems (OOD Ads, red), and combined systems (OOD Both, green). **(Right)** Enhanced OOD detection performance after filtering out atoms with high structural similarity to in-distribution environments (SOAP cosine similarity ≤ 0.8), demonstrating substantial improvement with AUC values of 0.902–0.907. The dashed diagonal line represents random classifier performance (AUC = 0.5). The dramatic improvement in the filtered case validates the hypothesis that bimodal uncertainty distributions arise from the presence of structurally familiar atomic environments within nominally out-of-distribution materials.

S12 Code Availability

The source code developed for this study, encompassing the implementation of all models, training procedures, evaluation protocols, and analysis scripts, is publicly accessible on GitHub. This promotes transparency and facilitates reproducibility of the presented results.

`https://github.com/YueGroup/DGKL`

The repository contains the following key components:

- Python scripts for model definition, data loading, training, and evaluation for all implemented UQ methods (DGKL, DGKL-Atomic, Ensemble, MCD, Evidential).
- Configuration files or scripts detailing the specific hyperparameters used for the experiments reported in the manuscript.
- Necessary dependency files (e.g., `pyproject.toml` for use with Poetry) to reconstruct the software environment.
- Documentation or README files providing instructions on setting up the environment, running the code, and reproducing the key findings.
- Pre-trained model weights may also be provided within the repository or linked for download (refer to the repository documentation for availability).

Researchers are encouraged to utilize and build upon this codebase.

Notes and references

[1] K. T. Winther, M. J. Hoffmann, J. R. Boes, O. Mamun, M. Bajdich and T. Bligaard, *Scientific Data*, 2019, **6**, 75.

[2] O. Mamun, K. T. Winther, J. R. Boes and T. Bligaard, *Scientific Data*, 2019, **6**, 76.

[3] L. Chanussot, A. Das, S. Goyal, T. Lavril, M. Shuaibi, M. Riviere, K. Tran, J. Heras-Domingo, C. Ho, W. Hu, A. Palizhati, A. Sriram, B. Wood, J. Yoon, D. Parikh, C. L. Zitnick and Z. Ulissi, *ACS Catalysis*, 2021, **11**, 6059–6072.

[4] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan and J. Snoek, *Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift*, 2019, `https://arxiv.org/abs/1906.02530`.

[5] Y. Li, L. Kong, Y. Du, Y. Yu, Y. Zhuang, W. Mu and C. Zhang, *MUBen: Benchmarking the Uncertainty of Molecular Representation Models*, 2024, `https://arxiv.org/abs/2306.10060`.

[6] B. Lakshminarayanan, A. Pritzel and C. Blundell, Advances in neural information processing systems 30, 2017.

[7] Ensemble Methods in Machine Learning, Berlin, Heidelberg, 2000, pp. 1–15.

[8] S. Fort, H. Hu and B. Lakshminarayanan, *Deep Ensembles: A Loss Landscape Perspective*, 2020, `https://arxiv.org/abs/1912.02757`.

[9] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez and I. Stoica, *Tune: A Research Platform for Distributed Model Selection and Training*, 2018, `https://arxiv.org/abs/1807.05118`.

[10] Y. Gal and Z. Ghahramani, Proceedings of The 33rd International Conference on Machine Learning, New York, New York, USA, 2016, pp. 1050–1059.

[11] A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia and C. W. Coley, *ACS Central Science*, 2021, **7**, 1356–1367.

[12] A. Amini, W. Schwarting, A. Soleimany and D. Rus, *Deep Evidential Uncertainty*, 2020, `https://openreview.net/forum?id=S1eSoeSYwr`.

[13] A. Amini, W. Schwarting, A. Soleimany and D. Rus, *Deep Evidential Regression*, 2020, `https://arxiv.org/abs/1910.02600`.

[14] C. J. Gruich, V. Madhavan, Y. Wang and B. R. Goldsmith, *Machine Learning: Science and Technology*, 2023, **4**, 025019.

[15] M. H. Rasmussen, C. Duan, H. J. Kulik and J. H. Jensen, *Journal of Cheminformatics*, 2023, **15**, 121.

[16] J. Dai, S. Adhikari and M. Wen, *Reviews in Chemical Engineering*, 2024.

[17] D. Levi, L. Gispan, N. Giladi and E. Fetaya, *Sensors*, 2022, **22**, year.

[18] T. Gneiting and M. Katzfuss, *Annual Review of Statistics and Its Application*, 2014, **1**, 125–151.

[19] B. Kompa, J. Snoek and A. L. Beam, *Entropy*, 2021, **23**, year.