A Machine Learning Based Approach to Accelerate Catalyst Discovery

Asif Jamil Chowdhury

A Machine Learning based Approach to Accelerate Catalyst Discovery

by

Asif Jamil Chowdhury

Bachelor of Science
Bangladesh University of Engineering and Technology, 2007

Submitted in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2020

Abstract

Computational catalysis, in contrast to experimental catalysis, uses approximations such as density functional theory (DFT) to compute properties of reaction intermediates. But DFT calculations for a large number of surface species on variety of active site models are resource intensive. In this work, we are building a machine learning based predictive framework for adsorption energies of intermediate species, which can reduce the computational overhead significantly. Our work includes the study and development of appropriate machine learning models and effective fingerprints or descriptors to predict energies accurately for different scenarios. Furthermore, Bayesian inverse problem, that integrates experimental catalysis with its computational counterpart, uses Markov chain Monte Carlo (MCMC) methods to refine the uncertainties on the quantities-of-interest such as turnover frequency. However, large number of forward simulations required by MCMC can become a bottleneck, especially in computational catalysis, where the evaluation of likelihood functions involves finding the solution to microkinetic models. A novel and faster MCMC method is proposed to reduce the number of expensive target evaluations and to shorten the burn-in period by emulating the target along with using a better informed proposal distribution.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning (ML) is a subfield of computer science that has intersecting areas with fields such as statistics, data mining, artificial intelligence, and it has proved to be an immensely useful tool for business, health, science and engineering. Machine learning operates by trying to find patterns in the data and can be used to make predictions that can save cost, time and effort. With the availability of more data in recent years, ML has been successfully applied to accelerate drug discovery [1], find patterns in genetic data [2], make accurate predictions for experimental design in molecular and materials science[3], detect objects in images and videos [4], understand natural language to classify documents [5]. My work focuses on using and extending machine learning models to develop predictive models for faster computational catalyst discovery.

Effective catalysts can speed up a chemical reaction by providing a lower energy pathway between the reactants and the products. Based on the phase of the catalyst compared to the reactants, catalysis can be divided into two areas: heterogeneous and homogeneous. Heterogeneous catalysts operate in a different phase from that of the reactants, i.e, the catalyst is a solid material whereas the reactants are in liquid or gas phase [6]. An important reason for the extensive use of heterogeneous catalysts in industry is that they are easily separable from the reactant and the products. These catalysts are typically porous materials so that the chemical reactions have more surface area on which to take place. The discovery of an effective heterogeneous catalyst is a complex process, and if done experimentally in a trial and error fashion, can be very time consuming. That is where lies the importance of computational catalysis.

Computational catalyst screening has the potential to significantly accelerate heterogeneous catalyst discovery. Typically this involves developing microkinetic reactor models (set of elementary reactions which are thought to be of relevance for a complete chemical transformation) that are based on parameters (such as rate constants) obtained from density functional theory (DFT) and transition state theory (TST). As analytical solution to many body Schrødinger equation is intractable, approximate methods like DFT has to be used instead. Still, there is a large computational overhead associated with the DFT calculations of different adsorption and transition state energies on various active site models. The uncertainty captured by composite probabilistic model for DFT energies is propagated through the microkinetic model to quantities-of-interest (QoI) such as turnover frequency (a measure of catalytic activity) or apparent activation energy using Monte Carlo simulations. This is the forward problem of the Bayesian inference. The inverse problem happens when we are given experimental measurements on QoIs such as TOF, and want to refine or reduce the uncertainties associated with the energy calculations. This step typically makes use of the Markov chain Monte Carlo (MCMC) algorithms. The MCMC methods, however, are difficult to use if the likelihood function is computationally expensive to evaluate. My work is centered on making these parts of the catalyst discovery workflow, i.e, the calculation of adsorption energies and the use of MCMC methods, more efficient.

In figure 1.1, the high level workflow for catalyst discovery is shown. The top and bottom flowcharts show the workflows with and without the use of machine learning, respectively. The key differences between the workflows lie in two areas: first, instead of calculating and using a full database of all the reaction intermediates, machine learning is applied to predict significant portion of the energies so that only a partial database suffices; second, although both the workflows use MCMC to solve Bayesian

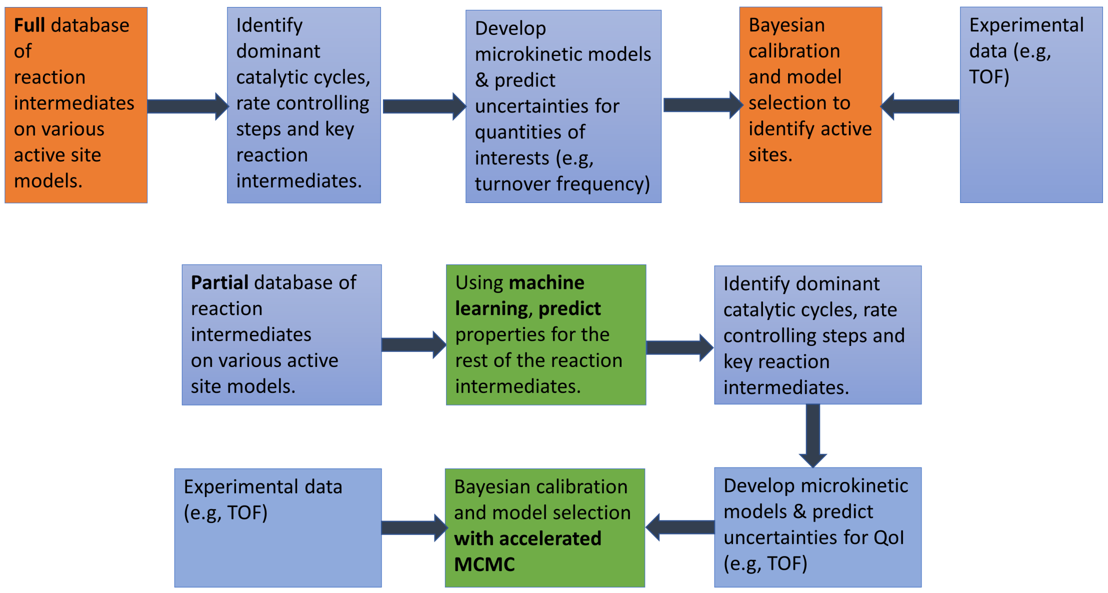

Full database of reaction intermediates on various active site models.
Identify dominant catalytic cycles, rate controlling steps and key reaction intermediates.
Develop microkinetic models & predict uncertainties for quantities of interests (e.g, turnover frequency)
Bayesian calibration and model selection to identify active sites.
Experimental data (e.g, TOF)
Partial database of reaction intermediates on various active site models.
Using **machine learning, predict** properties for the rest of the reaction intermediates.
Identify dominant catalytic cycles, rate controlling steps and key reaction intermediates.
Experimental data (e.g, TOF)
Bayesian calibration and model selection **with accelerated MCMC**
Develop microkinetic models & predict uncertainties for QoI (e.g, TOF)

Figure 1.1: Workflow of the heterogeneous catalyst discovery with and without our proposed improvements. The top flowchart shows the plain workflow without application of ML with our target areas highlighted. The bottom one shows the workflow with the application of ML. Here, the areas our current work focuses on are highlighted - instead of using full database of adsorption energies, ML is used to predict large number of them; and Bayesian inverse problem is solved with a new, accelerated MCMC method.

inverse problem (refinement of uncertainties using experimental measurements), our approach uses an improved and accelerated version of plain Metropolis-Hastings algorithm to reduce the number of expensive forward simulations. The areas where the current work investigates and improves upon are highlighted in the bottom workflow (works for prediction of adsorption energies using ML are discussed in chapters 2 and 3, whereas the improvement in the inverse problem is discussed in chapter 4) .

In order to reduce the computational cost for calculation of adsorption and transition state energies of all possible surface states on a large number of catalyst models, previous works have developed linear scaling relations for surface intermediates and transition states that only depend on a few, typically one or two metal descriptors such as the carbon atom adsorption energy. As a result, only the descriptor values have to be computed for various active site models to generate volcano curves for activity or selectivity. Unfortunately, for more complex chemistries the predictability of linear scaling relations is unknown. Also, the selection of descriptors is essentially a trial and error process. In our work, we have tested the effectiveness of non-linear machine learning models compared to the linear scaling relations when predicting the adsorption energy for various species on a metal surface based on data from the rest of the metal surfaces. Our results showed that linear scaling perform as good as the advanced ML models when the training dataset contains a complete set of energies for all the species on various metal surfaces. However, when the training dataset is incomplete, namely contains a random subset of species energies for each metal, molecular representations of the species have to be used as the descriptors along with the metal descriptors; and non-linear ML models significantly outperform linear models. To improve upon the trial and error process for the discovery of appropriate metal descriptors, we proposed an approach for automatic discovery based on principal component analysis (PCA). This part of the work has been published in a peer-reviewed journal [7] and the complete discussion is presented in chapter 2.

The work of chapter 2 was extended to a different scenario which warranted novel methods to achieve satisfactory results. This work is presented in chapter 3. For complex surface chemistries, the number of reaction intermediates can be very large and the cost of calculating the adsorption energies by DFT for all surface intermediates even for one active site model can become prohibitive. In our next work, we identified appropriate descriptors and machine learning models that can be used to predict a significant part of these adsorption energies given data on the rest of them. Moreover, our investigations also included the case when the species data used to train the predictive model is of different size relative to the species the model tries to predict - this is an extrapolation in the data space which is typically difficult with regular machine learning models. Due to the relative size of the available datasets, we attempted to extrapolate from the larger species to the smaller ones in our work. A neural network based predictive model was developed that combines an established additive atomic contribution based model with the concepts of a convolutional neural network that, when extrapolating, achieves a statistically significant improvement over the previous models. This work has also been published in a peer-reviewed journal [8].

Next, we move to the second part of the proposed improvement on the catalyst discovery workflow. To reduce uncertainties associated with the computational catalysis, we update them with the measurements from experimental catalysis. We propose an enhanced Metropolis-Hastings (MH) algorithm which we call MHGP, that requires less number of expensive posterior function evaluation, has shorter burn-in period, and uses a better and informed proposal distribution. The main innovations include the use of Bayesian optimization to reach the high density region quickly, emulating the target distribution using Gaussian processes (GP), and using Laplace approximation of the GP to build a proposal distribution that captures the underlying correlation better. An initial version of the work has been presented in a conference and the full version is in preparation to be submitted to a peer-reviewed journal. MHGP is discussed in chapter 4.

Chapter 2

Prediction of Adsorption Energies using Machine Learning

2.1 Introduction

Heterogeneous catalyst discovery using computational catalyst screening typically involves the development of a microkinetic reaction model that is based on DFT and TST [6]. To reduce the large computational cost of computing different adsorption and transition state energies on various active site models, linear scaling relations for surface intermediates and transition states have been developed [9, 10]. Linear scaling relations typically use a few easily computable descriptors which are computed for a variety of active site models. Then a volcano curve in activity or selectivity is generated as a function of the descriptors. However, the effectiveness of linear scaling relations is unknown for more complex chemistries. Besides, the descriptor selection process typically involves trial and error. In this paper, working on a predictive framework for the most stable ground state adsorption energies (without zero point correction) across a group of intermediate species and metal surfaces for the decarboxylation and decarbonylation of propionic acid [11], we propose an automatic process to discover efficient metal descriptors. We also compared the effectiveness of linear scaling with that of advanced machine learning (ML) models in predicting adsorption energies of surface intermediates in various scenarios. Specifically, when working with a set of metal surfaces (here the closed-packed surfaces of Ni, Pt, Pd, Ru, Rh, Re, Cu, Ag) and the adsorption energies for a set of intermediate species on those surfaces, we can think of a metal-species table where each row of the table contains adsorption energies of various species on a metal surface while each column contains adsorption energies of a species on different metal surfaces. Each cell in this table requires an expensive DFT calculation to obtain the value of the cell - the adsorption energy for an intermediate species for a particular metal. Our goal is to minimize the number of these calculations by predicting part of the table given training data on the other part. In this paper, we discuss two approaches for dividing the table into training set and prediction set.

One approach is to predict across metals - given energies for all intermediate species for some of the metal surfaces, we predict energies for all the species for the remaining metals. In other words, our training set is comprised of all the columns for some of the rows in the metal-species table, and the prediction set contains the rest of the rows. This is the approach that is commonly used in the catalysis community where linear scaling relations are used to predict adsorption energies [12] for species on a new metal surface. The typical choice of descriptors in this case is some combination of the adsorption energies of carbon, oxygen etc. In this paper, we use a more systematic approach that facilitates automatic descriptor discovery. Principal component analysis (PCA) [13] with varimax rotation [14] is used to find the best minimal set of adsorption energies that can be used as metal descriptors for a given data set. Our results show that the combination of descriptors obtained by this approach outperforms conventional descriptors like the adsorption energies of atomic carbon, hydrogen and oxygen. Also, the prediction results obtained by linear scaling with the discovered descriptors were compared with the predictions from non-linear machine learning models such as kernel based models and neural networks [15]. We found none of these advanced ML models to perform better than linear scaling when predicting across the metals.

The second approach is to choose surface-species pairs randomly for training and then predict on the rest. In terms of the metal-species table, the training set in this case consists of random cells from the entire table and the prediction set contains the rest of the cells. Thus, each row and each column of this table is only partially filled by the training data and the predictive model should fill in the missing ones. As this is a prediction not only across the metal surfaces but also across the species, we need additional species descriptors along with the metal descriptors from the first approach. Although this is not the conventional approach to catalyst discovery, we study it in this paper because first, for large number of intermediate species this approach can require fewer training points and second, this allows us to work with a general predictive framework for adsorption energies with a complete set of metal descriptors as well as the species descriptors. Finding appropriate species descriptors for predicting different chemical properties [16] is an active research area. Species descriptors have been used with ML models to predict atomization energies and other chemical properties as substitute for expensive DFT calculations [17, 18, 19, 20]. The descriptors that have been proposed range from a simple bond count or bond order to more complex Coulomb matrix or bag-of-bonds techniques [19, 20]. In the current work, we have studied different species descriptors along with the metal descriptors for predicting adsorption energies of random metal-species data points. Simple descriptors are desirable both because they do not require the geometry and coordinates of the species and surface atoms and also because they are less prone to overfitting. Our results show that a very simple descriptor like bond counts, when combined with the metal descriptors discovered in the first approach, has no statistically significant difference in prediction accuracy compared to more sophisticated descriptors.

Finally, the choice and calibration of the machine learning models is also studied. Unlike in the first approach, linear models proved inferior in predicting across metal and species compared to complex ML models. The highly time consuming process of

DFT calculations of the intermediate species across the metal surfaces means that in this problem domain, the size of the data set will not always be large and hence the machine learning algorithm has to approximate the underlying function with a relatively small training set - a challenge we expect to be common to many catalysis and materials science problems. Previous research on predicting the chemical properties had successfully used kernel ridge regression [18] and artificial neural networks [15]. In our predictive analysis of the adsorption energies we found that kernel based methods such as support vector regression, Gaussian process [21] and kernel ridge regression all worked well with prediction Mean-absolute-error (MAE) around 0.13 eV once their hyper-parameters were properly tuned. An additional benefit of the Gaussian process is that we can obtain the uncertainties around the predictions which is useful for uncertainty quantification in later stages of the calculation of the macroscopic quantities of interest (QoIs) such as catalyst's turnover frequency [22, 23]. Linear methods with regularizers had an MAE of around 0.28 eV, significantly higher than kernel based methods. Neural network with extensive hyper-parameter tuning had MAE a little over 0.2 eV, which is clearly an improvement over the linear models but not as good as the kernel based methods. With small data sets as in this case, neural nets can be prone to overfitting.

2.2 Methodology

We ran our predictive analysis on a data set comprising of the adsorption energies of a group of species of relevance for the decarboxylation and decarbonylation of propionic acid on eight different metal surfaces. In this section, details of the methods and algorithms are presented. We begin with a description of the data preparation. Then, we discuss prediction across metals - training the predictive model on all species energy for some of the surfaces and then predict for the rest of the surfaces. Here, we compare linear scaling with advanced ML models and also present the automatic discovery process for metal descriptors. Finally, we elaborate on prediction across species and metals - training on random points in the metal-species data space and predicting the rest of them. Again, a comparison among linear scaling and different types of ML models for this approach are presented. Here, we also discuss the feature engineering process of coming up with effective species descriptors.

2.2.1 Data Collection and Data Preparation

Adsorption energies are highly dependent on the metal surface structure [6]. In the current work, we have confined our predictive analysis on similar, i.e., closed-packed metal surface structures: *Pd(111), Pt(111), Ni(111), Rh(111), Ag(111), Cu(111), Re(0001), and Ru(0001).* Data were obtained from VASP calculations with PW91 functional for these metal surfaces and for each intermediate species in the microkinetic model of the decarboxylation and decarbonylation of propionic acid identified in our prior work [11, 24, 25] and illustrated in Figure 2.1. The adsorption energies as well as the geometry used in VASP calculations are considered in our predictive analysis. The geometric data was converted to a Coulomb matrix [19] and bag-of-bonds [20] to be used as features for the ML algorithms.

The energy data are prepared to have the same reference values. For example, the adsorption energy for an intermediate surface species $C_xH_yO_z$ was calculated as:

$$E_{C_xH_yO_z} = E^{DFT}_{C_xH_yO_z} - E^{DFT}_{*} - xE_C - yE_H - zE_O \tag{2.1}$$

where

$$\begin{aligned} E_C &= E^{DFT}_{CH_4(g)} - 2E^{DFT}_{H_2(g)} \\ E_H &= \frac{1}{2}E^{DFT}_{H_2(g)} \\ E_O &= E^{DFT}_{H_2O(g)} - E^{DFT}_{H_2(g)} \end{aligned}$$

Here, E^{DFT}_{*} is the energy of the free site (clean slab) and E^{DFT}_{X} denotes the adsorption energy of the species X from the DFT calculations.

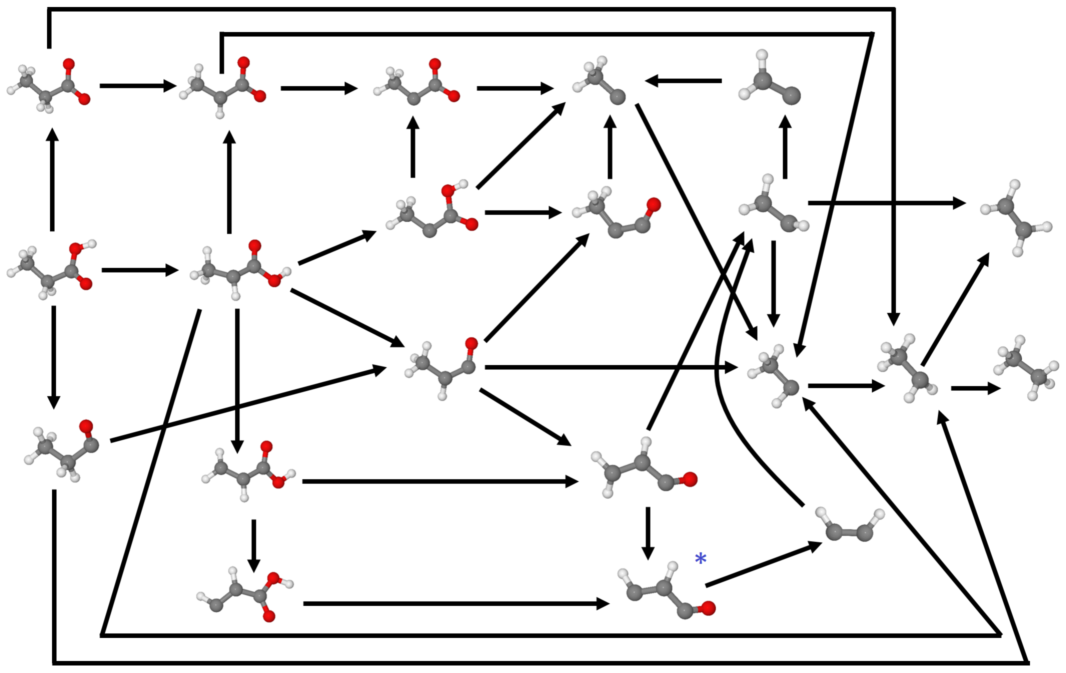


Figure 2.1: Reaction network for the decarboxylation and decarbonylation of propionic acid. The larger species among the metal descriptors ($CHCHCO$) is marked on the figure. The other descriptor (OH), along with $COOH$, CO_2, CO, H_2O and H, is not included in the figure for clarity.

2.2.2 Prediction Across Metals

Based on the d-band model [6] and past studies on the adsorption of small molecules on transition metal surfaces, the adsorption energies of different intermediate species are expected to show a linear scaling relationship against carefully chosen descriptors [12]. For each intermediate species in our data set, adsorption energies for all metal surfaces were plotted against various descriptors, i.e., the adsorption energies of all surface species in the reaction mechanism and other commonly used descriptors such as the carbon and oxygen adsorption energy. While the data show a trend, the standard deviations of the actual values from the best linear fits with these single descriptors were greater than 0.3 eV for many of the intermediate species as shown in Figure 2.2.

Considering the adsorption energies of the 26 intermediate species as a feature and then running Principal Component Analysis (PCA) [13] on the data reveals

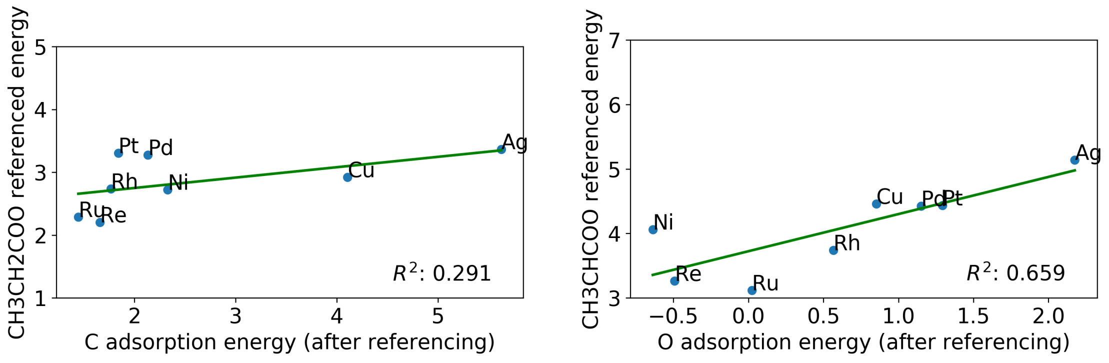


Figure 2.2: Two of the many cases where linear scaling with single descriptor does not yield a satisfactory result. Left: adsorption energies (after referencing) of CH_3CH_2COO against the C adsorption energy. Right: adsorption energies (after referencing) of CH_3CHCOO against the O adsorption energy.

that the first, second, and third principal components explain approximately 93%, 5%, 1% of the variance of the data, respectively. Hence, approximately 98% of the variance is explained by two factors. PCA reduces the dimensionality of the data but the descriptors are not directly identifiable from this because PCA learns a linear combination of the original variables as its components. So, we applied varimax rotation [14] which searches for a rotation of the factors from PCA and associates each original variable with one or a small number of factors, and thus, helps to find the most relevant original variables that best capture the variance in the data. With this approach, we found that adsorption energies of two species - $CHCHCO$ and OH aligns best with the first two principal components or factors. The study also reveals that when including C, H and O adsorption energies in the database, the adsorption energy of carbon becomes a dominant factor. So, in our predictive analysis, when using two metal descriptors, we used the adsorption energies of $CHCHCO$ and OH, and when using three metal descriptors we added the adsorption energy of carbon as the third descriptor. We predicted adsorption energies for all the species with seven of the eight metals for training and the other one for testing. Combinations of one, two, three, and four intermediate species adsorption energies were tried as descriptors. The results agree with our analysis with the best result of MAE 0.12 eV being obtained when using $CHCHCO$, OH and C as descriptors. Other good set of descriptors based on low MAE during linear scaling were: ($CHCHCO$, OH, CO), ($CHCHCO$, OH, O), ($CHCHCO$, OH, H), ($CHCHCO$, OH). The commonly used descriptor set in catalysis (C, O adsorption energies) resulted in an MAE of 0.2 eV approximately.

As we will show below, prediction using advanced machine learning methods such kernel based methods did not show a statistically significant improvement over linear scaling when predicting across metals. This result confirms the hypothesis that the adsorption energies show a linear scaling relationship. Our analysis shows that a combination of properly chosen descriptors are required in this case. The methodical approach of applying PCA and varimax rotation can help us identify the proper descriptors efficiently.

Automatic discovery of chemical descriptors have been done in previous research. One approach that has been proposed is a clustering-ranking-modeling method that ranks all candidate descriptors for each cluster based on their performance by running regression [26]. Another approach was to use LASSO as a feature selection technique [27]. This is a supervised learning way of finding appropriate features requiring exploration of a large number of possible descriptors. Since in our case the target variable and the feature variables are both adsorption energies of some species, the dimensionality of the whole data table can be reduced in an unsupervised manner with the application of PCA and then the varimax rotation provides the interpretability that PCA lacks [28] (as each principal component usually is a linear combination of a number of original variables) but LASSO is good at [29]. Moreover, with our approach we are able to come up with an appropriate number of descriptors automatically without the need to try out a large number of candidate descriptors. Nevertheless, we also ran LASSO feature selection with different sets of descriptors and found that the largest non-zero valued descriptors identified by LASSO match with the descriptors found by our approach.

2.2.3 Prediction Across Species and Metal

To build a predictive model which predicts not only across metals but also across species as it might be necessary for a complex/large reaction network where it is extremely time intensive to compute the energy of every possible surface species, we need some descriptors or features that can work as representations of the species molecules. These representations should include the interaction among the atoms of the species molecule as well as the interaction between the molecule and the metal surface. Combinations of these features are then fed to machine learning algorithms. The algorithms that we used can be broadly divided into three subclasses: linear models with L1 or L2 regularizers, kernel based methods, and artificial neural network.

Feature Engineering

Feature extraction for the representation of molecules and their interaction with the surface was performed by using the geometric data from VASP calculations. These descriptors need to be used in addition to the ones we were already using to predict across metal surfaces as in this case one wants to use information from one species for predicting the adsorption energy of another species. We computed the pairwise distance between the atoms of the intermediate species and the surface to find the number of C-C, C-H, C-O, O-H, C-M, and O-M bonds (single, double or triple bonds were not differentiated). Whenever the sum of the covalent radius [30] of the atoms involved in the bond was larger than the bond distance, we assigned a bond there. We also used species-only bond counts, i.e., the number of C-C, C-H, C-O, and O-H bonds. This has the advantage that no coordinate data is required and the descriptor values can be filled up with only pen-and-paper chemistry. In general, coordinate-free descriptors are preferred as they do not require any DFT or semi-empirical calculations once the model has been trained.

Then, we tried more sophisticated descriptors. The Coulomb matrix (CM) and bag-of-bonds (BoB) techniques have been used in previous research [31] for representing molecules. We have used these as descriptors to differentiate among the intermediate species structures. The diagonal entries in the Coulomb matrix are given by

$$C(i,i) = 0.5 * Z_i^{2.4} \tag{2.2}$$

where Z represents the atomic number. The off-diagonal (i,j)-th entry of the Coulomb matrix is given by

$$C(i,j) = \frac{Z_i * Z_j}{r} \tag{2.3}$$

where r represents the distance between the atoms in Angstrom. The sorted eigenvalues of the matrix are then used as the descriptor. The bag-of-bond method works with only the off-diagonal elements of the Coulomb matrix by placing the entries for each pair of atoms inside a bag and thus building a long vector [20]. The dimensionality of the descriptor vector increases with the number of atoms. If there are n atoms involved, the dimensionality of the CM is n and for BoB it is $\frac{n(n-1)}{2}$. In our case, three metal descriptors are added to these counts.

Machine learning models are prone to overfitting for high dimensional cases [32] if the training data are not large. For our propionic acid database, the largest intermediate species molecule has 11 atoms. To account for the surface interactions with the various sites on the metal surface, we need to consider at least the top two layers which means in our case 24 metal atoms. Our calculations were done with and without the metal atoms. Leaving out metal atoms would certainly deprive the predictor of important insights of the surface interactions, but it would also keep the dimensionality lower and stop the algorithm from overfitting.

When including the metal surface in our calculations, we removed the metal-metal atom interactions to reduce the dimensionality. We also introduced a cutoff distance to make the C(i,j)-th entry zero when the distance between the atoms involved exceeds the cutoff value (in Angstrom). As discussed later, we tested different cutoff values and different powers of the cutoff value c and that of r - the distance measure in the Coulomb matrix. After these modifications the (i,j)-th entry had the form

$$C(i,j) = \frac{Z_i * Z_j}{r^{power}} - \frac{Z_i * Z_j}{c^{power}} \tag{2.4}$$

where c is the cutoff distance. If one of the atoms involved in the (i,j)-th entry was the metal atom, we tried taking the square root or natural logarithm of the metal atomic number to come up with its Z value. This made sure that the entries involving species-metal atom pairs did not have too large values compared to the entries for species-species atom pairs. Here, we also tried approaches such as re-scaling or standardization to reduce the magnitude of the metal atomic number. However, our initial results found the square root or natural log to be superior to these and hence, square root and natural log were used for the rest of the investigations.

Machine Learning Models

Several machine learning algorithms were applied and are briefly described below - linear models such as linear regression, ridge regression, lasso; kernel based methods such as support vector regression (SVR), kernel ridge regression (KRR), Gaussian process (GP); and artificial neural networks (ANN). Kernel based methods consistently achieved best results. Of these kernel based methods, Gaussian process is of particular interest as it achieves results as good as SVR or KRR and at the same time provides the uncertainty information which can be useful for subsequent steps of macroscopic quantities of interest estimations, i.e., if one is interested in how the uncertainty of a species energy affects the turnover frequency.

A Gaussian Process (GP) is a collection of random variables such that any finite subset of those variables has a multivariate normal distribution [33]. A Gaussian Process' behavior is primarily governed by the covariance function $k(x, x^*)$ it uses. The covariance function or the kernel defines the relation between any pair of data points. The Gaussian process prior is zero mean with a valid covariance function k (which means the matrix obtained by applying this function on each pair of points must be positive definite [34]). The input error is normally distributed with variance σ^2 and the training set consists of input-output pairs (X, Y) and the test inputs X^* yield outputs Y^* in the form

$$Y^*|Y, X, X^* \sim \mathcal{N}(\mu_p, \Sigma_p) \tag{2.5}$$

where μ_p is the posterior mean,

$$\mu_p = K(X^*, X)(K(X, X) + \sigma_n^2 I)^{-1} Y \tag{2.6}$$

and Σ_p is the posterior variance,

$$\Sigma_p = K(X^*, X^*) + \sigma_n^2 I - K(X^*, X)(K(X, X) + \sigma_n^2 I)^{-1} K(X, X^*) \tag{2.7}$$

where σ_n^2 is the noise variance. Each entry of the covariance matrix K contains the kernel function evaluation for each pair of points. Of the many available kernel functions, we used the most commonly used Gaussian kernel:

$$k(x_i, x_j) = \sigma_y^2 exp(-\frac{(x_i - x_j)^2}{2l^2}) \tag{2.8}$$

which can be extended to multi-dimensional scenarios. Here, σ_y^2 is the kernel variance and l is the length scale. The length scale can be the same for all the dimensions or a different length scale is learned for each of the dimensions - which is called Gaussian Process with Automatic Relevance Determination (GP-ARD). In our case, we used the standard GP as the initial results showed it to outperform GP-ARD for our dataset.

SVR, another kernel based method, is the extension of the classical support vector classification (SVC) to solve regression problems. It uses a subset of the training data (which are called support vectors) to make predictions. As with GP, there is a choice of kernels that depends on the prediction task at hand. In our case, through the use of 5-fold cross-validation [35] we found that the Gaussian kernel worked best in our case. Although the Laplacian kernel, which uses the Manhattan distance between the input vectors instead of the squared one as in the Gaussian kernel, has been shown [36] to work well in predicting different molecular properties, our results indicate that the Gaussian kernel achieves slightly better results in predicting adsorption energies compared to the Laplacian. Another kernel based method, KRR is similar to SVR, but uses a different loss function, learns a non-sparse model and its estimation can be done in closed form. Again, we found through 5-fold cross-validation that the Gaussian kernel is superior for our dataset.

ANN, also known as Multilayer Perceptron, is a model that can learn highly non-linear functions but suffers from a non-convex loss function and hence is prone to get stuck in poor local minima or plateau [37]. With careful initialization and proper hyperparameter (such as the number of hidden layers, number of units or iteration) tuning, this problem can be minimized. The expressiveness of the model comes from the fact that the hidden layers learn progressively more complex representations of the input data. However, an ANN typically requires larger training data in order to find a good generalization, which in our case is not available. For our dataset we found that kernel based methods achieve better results than ANN. We applied different regularization schemes such as L2 regularization, dropout, and early stopping to keep the ANN model from overfitting. Other models, such as ridge, SVR, and KRR also uses L2 regularization whereas LASSO uses L1 regularization to tackle overfitting.

Table 2.1: Results of linear scaling using different sets of metal descriptors to predict across metals.

Descriptor Combination	MAE (eV)	SD of AE (eV)	SD of MAEs of Metals (eV)
$CHCHCO, OH, C$	0.120	0.110	0.037
$CHCHCO, OH$	0.138	0.149	0.052
C, OH, O	0.149	0.144	0.050
C, OH	0.153	0.143	0.058
C, O	0.195	0.184	0.077
C, H, O	0.203	0.190	0.080

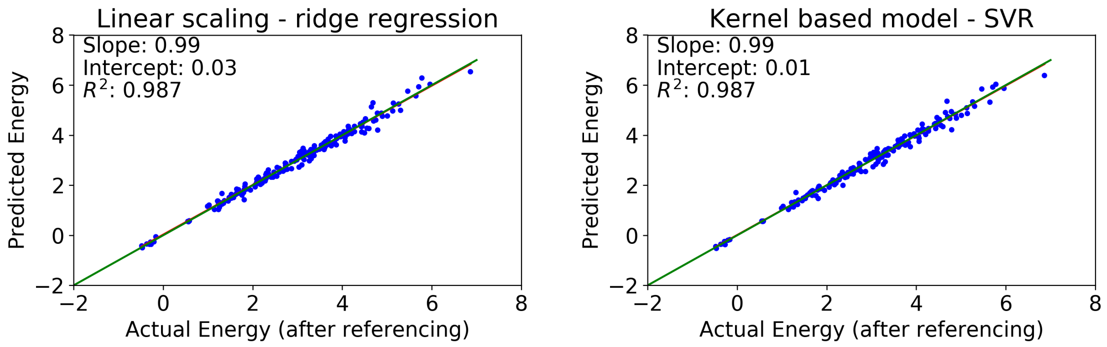


Figure 2.3: Predicted energy (after referencing) vs actual energy (after referencing) for predictions across metals, i.e., predicting adsorption energies of all intermediate species for a metal given the energies for all the species for the rest of the metals. Left: using linear scaling (linear ridge regression on the metal descriptors). Right: using non-linear Method (KRR on metal descriptors). In both cases the the adsorption energies of $CHCHCO$, OH and C were used as metal descriptors. The plots look very similar. We performed a statistical test to see if there is any statistically significant difference between the means of the absolute errors of the two methods. A p-value of over 0.2 confirmed that there is no statistically significant difference between the results of linear scaling and sophisticated machine learning methods (such as kernel ridge regression in this case) when predicting across metals.

2.3 Results and Discussion

The data for the decarboxylation and decarbonylation of propionic acid contain information on 26 intermediate species across 8 metal surfaces making the total size of the data set 208. While predicting across metals, data for seven of the metals were used for training and the eighth was used for testing. The process was repeated for each of the metals. To test linear scaling, we used different combinations of adsorption energies of molecules as descriptors. Not all combinations of descriptors achieved good results as can be see in Table 2.1. Here, we see a comparison among different sets of metal descriptors. Clearly, not all descriptor sets produced near-best results. The set of descriptors (adsorption energies of $CHCHCO$, OH, C) that we found in our analysis through principal component analysis (PCA) and varimax rotation shows superior performance than other sets. The reported MAEs and the standard deviations of the absolute errors (AE) are calculated on all 208 data points by training on the data for seven metals and predicting on the eighth metal and repeating for each of the eight metals. The statistical comparison between the second row (subset of the descriptors obtained from our analysis) and the fifth row (common descriptors, not including the first two optimum descriptors found in our analysis) has a p-value of less than 0.001 which establishes a statistically significant difference between the results. We followed a systematic machine learning approach to find the best descriptors - by applying Principal Component Analysis (PCA) on the data and then performing varimax rotations. This procedure gave us the optimum set of descriptors: adsorption energies of $CHCHCO$, OH and C. We applied linear regression with L-2 regularizer (also known as ridge regression) with the set of optimum descriptors. We also applied advanced non-linear machine learning methods such as SVR and KRR with this descriptor set. The best MAE was approximately 0.12 eV for both linear regression and for non-linear models. We also combined these metal descriptors with species descriptors such as bond counts, Coulomb matrix and bag-of-bonds. The results of non-linear models or the inclusion of species descriptors did not show any statistically significant difference when compared to the linear scaling (with a p-value of over 0.1). A comparison between the predictions of linear scaling and a non-linear method is shown in Figure 2.3. Results for predictions across metals for linear ridge regression and the kernel based methods are shown in Table 2.2. The first three rows shows results for linear and non-linear models when using only these metal descriptors. For

Table 2.2: Results of prediction across metals, i.e., given adsorption energies of all intermediate species for seven metals, we tried to predict the adsorption energies for all intermediate species of the remaining metal.

Method	MAE (eV)	SD of AE (eV)	SD of MAEs of Metals (eV)
Ridge Regression with metal descriptors	0.120	0.110	0.037
SVR with metal descriptors	0.120	0.109	0.046
KRR with metal descriptors	0.127	0.118	0.050
GP with BoB, bond counts and metal descriptors	0.134	0.136	0.063
SVR with BoB, bond counts and metal descriptors	0.136	0.129	0.057
KRR with BoB, bond counts and metal descriptors	0.137	0.168	0.075

each species, training on data for seven metals and testing on the eighth and repeating the process for each of the metals and each intermediate species gives us predicted adsorption energies for each species on each surface. Absolute error (AE) for each case is obtained by taking the absolute difference between the predicted and the real energies. The mean and standard deviation of these absolute errors are shown in the second and the third columns, respectively. Testing on each metal surface also provided us a mean-absolute-error (MAE) for each metal. The standard deviation of these MAEs are shown in the fourth column. The last three rows show results when species descriptors such as CM, BoB, bond counts are included along with the metal descriptors. In this case, the training set contained energies for all species for seven metal surfaces and testing on all the species of the eighth and then repeating the process for each metal. The results show that linear ridge regression with just appropriate metal descriptors performs as good as the kernel based methods such as kernel ridge regression (KRR) or support vector regression (SVR). Even with other descriptors such as bag-of-bonds (BoB) or bond counts, the results are not better than linear scaling with carefully chosen metal descriptors.

Predicting across metals and species requires some additional descriptors that capture the representation of the species and the interaction between the species and the surface. For this, we used simple descriptors like bond counts and more complex descriptors such as Coulomb matrix (CM) and bag-of-bonds (BoB). We used different

Table 2.3: Prediction across metals and species with non-linear and linear machine learning methods with different sets of descriptors.

Species Descriptors	Metal Descriptors	ML method	Mean of MAEs (eV)	SD of AEs (eV)	SD of MAEs (eV)
Bond Counts (Species and Metal)	$CHCHCO, OH$	SVR	0.133	0.129	0.023
Bond Counts (Species and Metal)	$CHCHCO, OH$	KRR	0.134	0.128	0.025
BoB incl metal, sqrt, cutoff 6Å, power 5	$CHCHCO, OH, C$	GP	0.128	0.120	0.022
BoB incl metal, ln, cutoff 4.5Å, power 4	$CHCHCO, OH, C$	GP	0.129	0.131	0.023
BoB power 5 + Bond Counts (Species and Metal)	$CHCHCO, OH, C$	GP	0.139	0.140	0.027
BoB power 1 + Metal Bond Counts	$CHCHCO, OH, C$	KRR	0.140	0.159	0.029
BoB power 4 + Bond Counts (Species and Metal)	$CHCHCO, OH, C$	SVR	0.142	0.134	0.024
CM power 1 + Bond Counts (Species and Metal)	$CHCHCO, OH$	KRR	0.145	0.155	0.026
Bond Counts (Species Only)	$CHCHCO, OH, C$	GP	0.185	0.186	0.030
Bond Counts (Species and Metal)	$CHCHCO, OH$	ANN	0.214	0.205	0.064
BoB incl metal, ln, cutoff 6Å, power 2	$CHCHCO, OH$	Ridge	0.277	0.245	0.042
BoB power 2 + Bond Counts (Species and Metal)	$CHCHCO, OH$	Lasso	0.293	0.272	0.041
CM power 4 + Bond Counts (Species and Metal)	$CHCHCO, OH, C$	Elastic	0.294	0.274	0.038
Bond Counts (Species and Metal)	$CHCHCO, OH$	Ridge	0.347	0.295	0.045

combinations of these descriptors along with the metal descriptors that we already have. We also tried different values of power and cutoff (in equation (2.4)). In this scenario, the non-linear ML methods fared much better than the simple linear models (see Table 2.3). Non-linear ML models used here are artificial neural network (ANN) and Kernel based methods like support vector regression (SVR), kernel ridge regression (KRR), and Gaussian process (GP). Kernel based methods outperform ANN. The top row contains the result for the same case as the left image of Figure 2.5. The tenth row contains the result for ANN. The 3rd and the 4th rows show the results for BoB with metal atoms included, with different cutoff values and different methods (natural log and square root) to minimize the difference between metal and species atoms' atomic number. The ninth row contains the result for the species only bond counts which can be obtained by pen-and-paper chemistry. The bottom four rows present the results for linear machine learning models. These models vary in terms of the regularizer that they use - in case of ridge it is L2 regularizer, in case of lasso it is L1 regularizer and for elastic it is a combination of both regularizers [38]. Comparing the prediction errors with that of the non-linear models in the top rows of the table clearly shows the value of advanced ML methods in a full predictive model, i.e., when we try to predict not only across the metals but across the species as well. In order to get an unbiased estimate of the prediction error, we randomly permuted the data 100 times and at each time we split it into training and testing sets (with 160 and 48 data points for training and testing, respectively) and get a Mean Absolute Error (MAE). The mean and the standard deviation of these MAEs are reported in the fourth and the sixth column, respectively (since each run has the same number of testing points, the mean of MAE equals the mean of the absolute errors over 100 runs). The fifth column presents the standard deviation over all the absolute errors. Here, metal bond counts refer to the carbon-metal and oxygen-metal bond counts. Power refers to the exponent in the denominator of equation (2.4). Cutoff refers to the base of the denominator of the second term of equation (2.4). We tried with powers 1 to 5 and cutoff values from 2.5Å to 6.0Å with 0.5Å intervals. While the kernel based methods achieved an MAE of approximately 0.13 eV, linear regression with L-2 regularizer had an MAE exceeding 0.28 eV. These results signify the importance of advanced ML techniques for a full predictive model of adsorption energies of various species on different surfaces.

The predictions of machine learning algorithms are highly dependent on an appropriate choice of the hyperparameters. In our case, different sets of descriptors for an ML method would require different hyperparameters. So for each ML method and each descriptor set, we first divided the randomly shuffled data into training and testing sets; then, we ran five-fold cross validation on the training set to obtain optimal hyperparameters. After that, we obtained the prediction errors on the testing set using those hyperparameters. This process was repeated 100 times with the data set randomly shuffled each time to obtain an unbiased estimate of the prediction error for each machine learning method and each descriptor set. Out of the 208 data points, we used 160 for training. The prediction errors tend to decrease with an increase in training size. However, this rate of decrease diminishes as more training points are added as is evident in Figure 2.4 and it is not clear that the MAE could be reduced significantly below 0.1 eV by increasing the training set size.

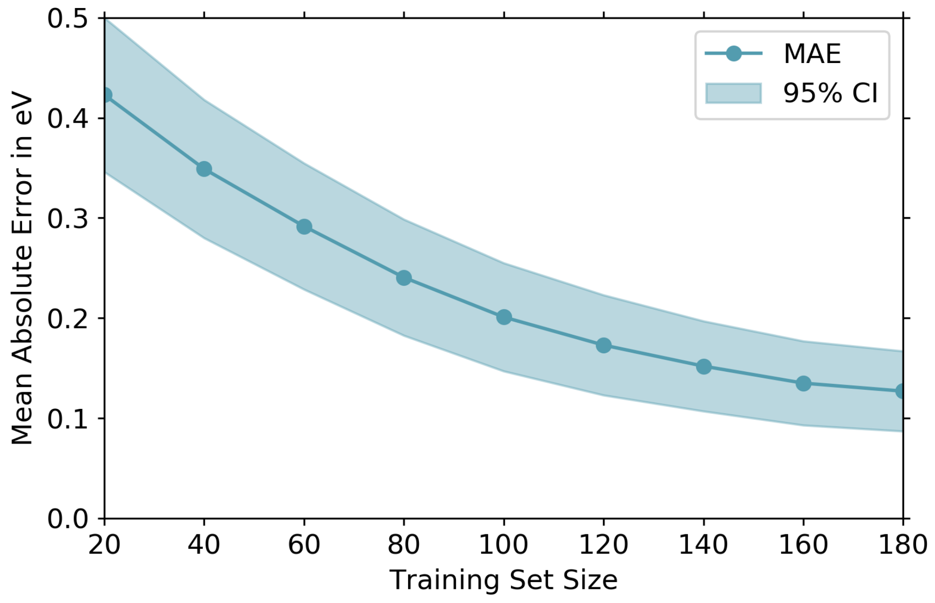


Figure 2.4: Mean Absolute Error (MAE) of prediction vs training set size while predicting across metals and species, i.e., we are testing on a set of randomly chosen rows from the full dataset (which can contain rows from any metal or species) given the rest of the rows for training. The ML model used here was the Gaussian process (GP) and the descriptors were the bond counts (carbon-carbon, carbon-oxygen, carbon-hydrogen, oxygen-hydrogen, carbon-metal and oxygen-metal bond counts) and the metal descriptors (adsorption energies of $CHCHCO$, OH and C). We found similar trends for other ML models and different combinations of descriptors. For each training set size, data was randomly permuted 100 times and split into training and testing set. The means of these 100 runs are shown here. The standard deviation of the MAEs for each training set size provides the 95% confidence interval which is shown in the shaded region.

A comparison between the prediction errors of the non-linear ML model and linear regression is presented in Figure 2.5. The linear case has many more points deviating from the ideal prediction line compared to the non-linear method. Some better results on different descriptor sets for non-linear methods such as the ANN and kernel based methods - SVR, KRR, and GP are shown in Table 2.3. For predictions from GP, we have extra information about the uncertainties around the prediction points. This uncertainty information allowed us to calculate 95% confidence intervals around each prediction point for GP. We found each of the actual energies to lie within this interval which indicates that GP captured the uncertainty well. Comparing the results of the kernel based models after hyperparameter tuning, we found no statistically significant difference among themselves. As we ran predictions hundred times for each case, the table shows the mean and standard deviation of the absolute errors of all runs and the standard deviations of the MAEs of those runs.

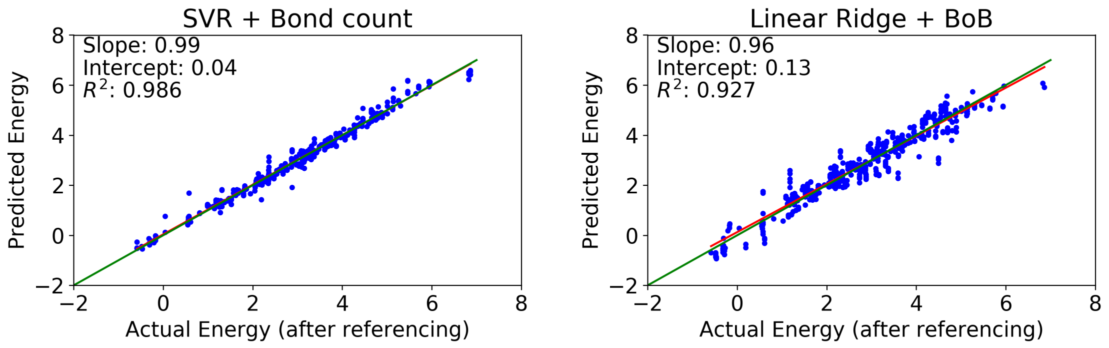


Figure 2.5: Comparison between the predictions from kernel based ML models (on the left) and linear scaling (on the right) while predicting across metals and species, i.e., we are predicting a test set which is chosen randomly from the whole dataset and thus, it can contain rows from any combination of metal or species. The rest of the data is used for training. We used 160 data points for training and the rest for testing. For each machine learning method and each descriptor set, we split the data into training and testing set after randomly shuffling it, then we performed 5-fold cross-validation on the training set to find optimized hyperparameters which we then used to get the MAE on the testing set. We repeated the process one hundred times to get an unbiased estimate of the prediction error. On the left: results from running support vector regression (SVR) with bond counts (C-C, C-O, C-H, O-H, C-M and O-M counts) and metal descriptors (adsorption energies of $CHCHCO$, OH). On the right: results from running linear ridge regression with bag-of-bonds (BoB) and same metal descriptors. Unlike Figure 2.3, here we see the kernel based method predicted much closer to the ideal line consistently and the statistical test that we performed showed that the prediction errors from advanced ML methods were lower than those from linear scaling. When predicting only across metals, we can use just the metal descriptors for which linear scaling works well; but for a full general predictive model like this, we need additional descriptors to represent the species and then we need more sophisticated ML models.

The results in Table 2.3 show that simple descriptor sets like bond count (with appropriate metal descriptors) worked nearly as well as the more complex descriptors such as CM and BoB. The bond counts contain the number of bonds inside the species and between the species and the surface. If we omit the C-M and O-M which are the metal bond counts, we are left with a species-only-bond-count that can be obtained by just pen-and-paper chemistry without any geometry data. As can be seen in Table 2.3, this case has an MAE less than 0.19 eV. The table also shows that when using geometry based methods such as bag-of-bonds, including metal surface with the BoB computation (the 3rd and the 4th rows) yield slightly better result than when using the BoB on the species alone and then incorporating the C-M, O-M bond counts (the sixth row), but still no statistically significant improvement is found compared to simple bond counts (top two rows). The results for linear methods are shown at the bottom four rows of the table. Again, the advantage of using advanced ML models in this case is evident from the results.

2.4 Conclusion

Effective prediction of adsorption energies on heterogeneous catalyst surfaces requires beyond a database, both a proper set of descriptors and proper choice and calibration of the machine learning model. We have studied both of these in the current work. An automatic method to discover effective metal descriptors is presented based on PCA and varimax rotation. Our comparative study has illustrated that when predicting adsorption energies for the species on a metal surface given the energies of those species on other surfaces, linear scaling with appropriate metal descriptors holds well with MAE of approximately 0.12 eV. In this case, we found no statistically significant difference between the performances of the regularized linear regression and that of the advanced ML models. However, we have shown that an appropriate choice of descriptors, which can be obtained by our proposed method, is necessary and the results of commonly used descriptors can be significantly inferior compared to the predictions from an optimum descriptor set. We also studied an uncommon scenario in catalyst screening where we predict random species on random metal surfaces given random training data. While such a scenario is currently rarely used, we believe it to become more relevant when more complex reaction mechanisms are studied on surfaces and preliminary mechanisms are found to be incomplete, i.e., there is a desire to extend the mechanism in a more ad hoc fashion. In this case, where we combined other species descriptors along with the metal descriptors, we found the non-linear ML models to significantly outperform linear models with an MAE of 0.13 eV. Interestingly, comparing this result with that of the prediction across metals suggests that given data on a sufficient number of metal surfaces for a species, information from other species, even with full optimized coordinate information, do not add much to the learning of the energy for that species on a new metal surface. However, information from other species becomes useful when very few data for that species on different metal surfaces are available (a likely case for random training data). Another key outcome of these studies is that advanced machine learning models work well in any scenario of predicting adsorption energies on the surface. Our investigations with different descriptors show that for ML models to succeed, it is not necessary to use advanced (geometric) coordinate based descriptors; simple descriptors such as bond count can provide satisfactory results. As many catalysis and materials science problems require significant time to generate each data point, in many cases the ML models would need to work with a relatively small sized dataset. This requires careful tuning of hyperparameters using cross-validation, use of regularization to account for overfitting, and reducing the dimensionality of the descriptor space - all of which have been studied in the present work. Our studies and investigations have shown that it is possible to predict the adsorption energies using machine learning with reasonable accuracy when all these constraints are properly addressed.

Chapter 3

A Multiple Filter Based Neural Network Approach to the Extrapolation of Adsorption Energies on Metal Surfaces for Catalysis Applications

3.1 Introduction

Computational catalyst discovery typically requires the development of a microkinetic model based on parameters determined by density functional theory (DFT) calculations [6] of all reaction intermediates [39]. To minimize the cost of calculating the energies for each reaction intermediate and transition state on different active site models, linear scaling relations [9, 10, 12] have been proposed which use a few easily computable descriptors, such as the carbon atom adsorption energy, on different active site models to generate volcano curves on catalyst activity [40]. However, even the DFT computations for only the intermediate species on a number of surfaces require, for a large reaction network with many intermediates, a significant number of expensive calculations. In this work, our goal was to build a predictive framework that would train on the energies of some of the surface species and predict on the rest, which can significantly reduce the computational overhead when working with a complex microkinetic model with a large number of surface species. In addition to that, we also investigated appropriate predictive models for extrapolation of adsorption energies in terms of the size of the species, i.e, when the training data and the prediction set contain different sized molecules. This is typically challenging because machine learning models, while performing satisfactorily during interpolation (when training and testing set come from the same area of the feature space), do not work as well for extrapolation [41] (when training and testing set come from non-overlapping regions of the feature space).

In this paper, we have worked with two data sets of adsorption energies, both containing reaction intermediates consisting of carbon, oxygen, and hydrogen atoms. One of them contains 247 larger C4 species, i.e, molecules with at least four carbon atoms and variable numbers of oxygen and hydrogen, obtained from the hydrodeoxygenation of succinic acid on Pt(111). The other contains 29 smaller C2 and C3 species, i.e, molecules made up of two or three carbon atoms along with some oxygen and/or hydrogen, obtained from a reaction network of decarboxylation and decarbonylation reactions of propionic acid on Pt(111) [11]. All calculations were done using the PBE-D3 functional. Two types of predictive analysis were performed - interpolation on the bigger C4 dataset, i.e, training on some of the C4 species and predicting on the rest of them; and extrapolation from the C4 data set to the C2 and C3 data set, i.e, training on the full C4 data and predicting the adsorption energies for the C2 and C3 species. While extrapolation to longer chain molecules is in principle most relevant, we do not possess a C5 dataset and the C2 and C3 datasets are too small to be used for extrapolation to C4 species. Nevertheless, extrapolation from C4 to C2 and C3 is technically as challenging as extrapolation to longer chain molecules and we expect all of our conclusions to also be valid when extrapolating to longer chain molecules provided these do not contain any chemical fragment that is non-existent in the smaller training molecules.

Predicting properties of some chemical entity using machine learning [16, 17] involves solving two related subproblems - discovery of effective features or descriptors, and using a proper machine learning model that, together with the chosen descrip-

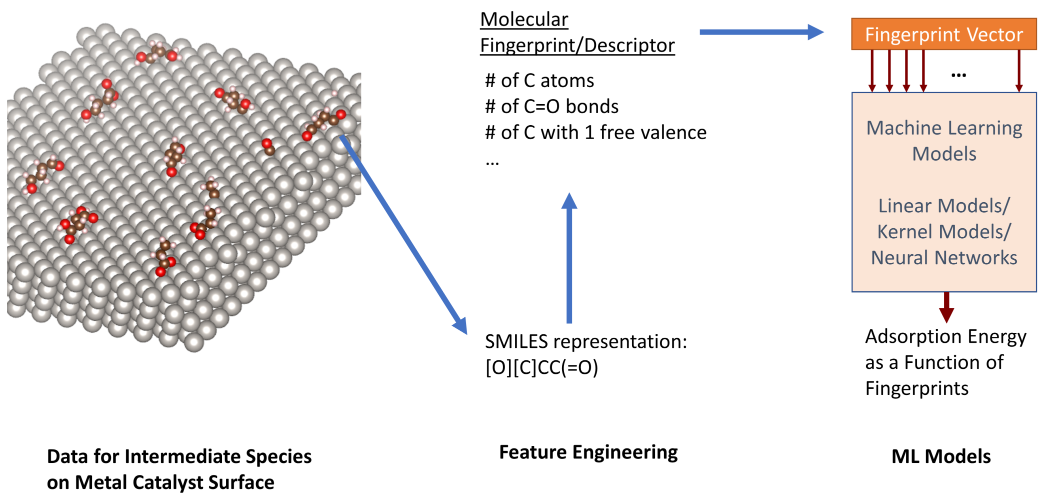

Molecular Fingerprint/Descriptor
# of C atoms
# of C=O bonds
# of C with 1 free valence
…
SMILES representation: [O][C]CC(=O)
Fingerprint Vector
…
Machine Learning Models
Linear Models/ Kernel Models/ Neural Networks
Adsorption Energy as a Function of Fingerprints
Data for Intermediate Species on Metal Catalyst Surface
Feature Engineering
ML Models

Figure 3.1: An overview of the application of machine learning for prediction of adsorption energies. Structures of the intermediate species are used to obtain a suitable fingerprint, which is fed to ML models that learn the adsorption energy as a function of the fingerprint vector.

tor, works best for the specific task at hand [7]. A high-level workflow for applying machine learning for this process is shown in Figure 3.1. Here, we are trying to predict the adsorption energies of surface intermediates and hence, the descriptors are essentially some form of molecular fingerprints. Many different kinds of fingerprints or fingerprint generation schemes have been proposed in previous studies - Coulomb matrix [19] and bag-of-bonds [20] using distance measures between the atomic coordinates of the species; atom centered radial or angular symmetry functions [42, 43, 44, 45]; non-coordinate based fingerprints that take into account features of a molecule which can be extracted from the chemical formula or SMILES notation [46, 47, 48, 49]; generation of fingerprints from molecular graph structure [50, 51] where the atoms and the bonds are considered as the nodes and edges of a graph, respectively, and fingerprints corresponding to a target property learned using back-propagation etc. Fingerprints based on SMILES or graph have the desirable property over the coordinate based descriptors that any DFT or other semi-empirical methods need to be applied only on the training data; for the rest of the data for which the adsorption energies are unknown, their molecular notation is all that the predictive model would need to make the predictions. In contrast, the coordinate based methods would need reliable atomic coordinates even for the species on the prediction set which would require some form of expensive calculations - ones we wish to minimize in the first place. Most commonly used machine learning models have been kernel based models such as kernel ridge regression [18] and different neural network based models such as graph convolution [50, 51], recurrent neural network [52], 3D convolutional neural network which reads the 3D spatial coordinates of the molecule [53], or additive atomic contribution through atomic subnetworks [16, 42].

In our investigations for interpolation of adsorption energies, we have studied both the coordinate based and SMILES based descriptors along with a variety of machine learning models. Our results indicate that simple molecular descriptors that capture the nearest neighbor information across the species from the SMILES notation, paired with kernel based models can perform as good as coordinate based descriptors such as Coulomb matrix or bag-of-bonds with a mean absolute error (MAE) of 0.14 eV. However, for extrapolation, the choice of descriptor is more complex - descriptors based on pairwise or triplet atomic distances such as Coulomb matrix or bag-of-bonds have the disadvantage that they are not size extensible. For data with different sized species, smaller ones have to be padded with zeros that make the learning difficult. In this case, constant sized molecular fingerprints [54] are more suitable. Our results, however, suggest that the predictive errors are still quite high for these descriptors. A different kind of approach, which is atom centered and where each atom's neighborhood information (its pairwise and triplet distances from other atoms) is treated as the atomic fingerprint and fed to a small neural network where these subnetworks from each atom share their weights, and then all of the atoms' contributions are added up to get the final energy, is size extensible. We found this method to work better than the other methods for extrapolation with MAEs of around 0.4 eV. However, the error is still large and we have sought ways to improve upon this model. One improvement was to include SMILES based atomic fingerprints over the coordinate based ones, and the second contribution, that helped to get significantly smaller extrapolation errors, was to treat the small atomic subnetworks like a filter of a convolution neural network and use multiple of these filters. This method had extrapolation MAE of 0.23 eV.

3.2 Methodology

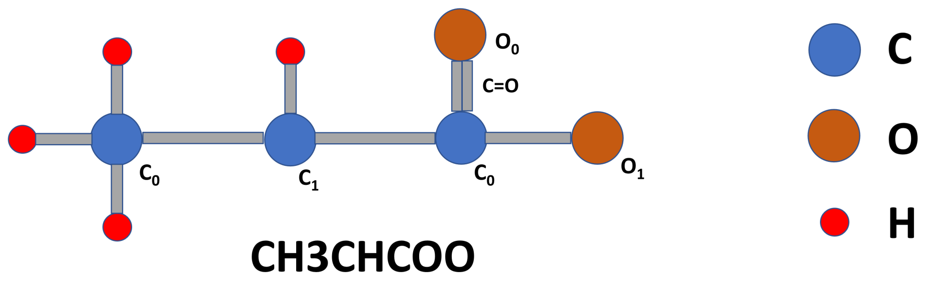
Molecular fingerprints used in our investigations can be categorized into three classes: first, coordinate based Coulomb matrix and bag-of-bonds that use pairwise distances between the atoms in the molecule to generate the fingerprint; second, flat fingerprints based on the number of different bond counts inside the molecule that can be read from the chemical formula or SMILES notation; third, atom centered fingerprints that are based on the local neighborhood around an atom which is calculated either by distance measures between the atomic coordinates or by the number of different bond types for the atom that can be read from the molecular notation. Machine learning models that we have used can also be divided into three categories: generalized linear models such as linear regression, ridge regression (which uses L2 regularizer), LASSO (which uses L1 regularizer); kernel based models such as kernel ridge regression (KRR), support vector regression (SVR), Gaussian processes (GP); and artificial neural networks (ANN).

3.2.1 Coulomb Matrix and Bag-of-bonds

The Coulomb matrix (CM) method first creates a symmetric matrix where the off-diagonal element $C(i, j)$ is a function of the distance measures between the i-th and j-th atoms and also their atomic numbers. The diagonal elements are a function of the atomic number of the corresponding atoms. The sorted eigenvalues of the matrix forms the molecular fingerprint. The Bag-of-bond (BoB) method takes the off-diagonal lower triangle of the symmetric matrix formed in CM and puts the entries corresponding to each atom type pair in a bag, sorts the entries inside each bag and concatenates the bags to form the fingerprint vector. We have found these methods to typically work well for interpolative predictions among the same sized species. However, for data with variable sized species, one needs to pad the entries of the matrix for smaller species with zeros. This limits their usefulness for size-extrapolation predictions. Detailed methods for building CM and BoB are described in the supporting information.

3.2.2 Flat Molecular Fingerprint

Fingerprints generated from the SMILES notation of the adsorbed species encode the connectivity among the atoms inside the molecule. The encoding can capture the number of different types of atoms or bonds by looking into the nearest neighbors of each atom or upto some specified distance. In our study, we have built a simple scheme, similar to previous work on constant-sized descriptors [54], that looks into the nearest neighbors of the atoms and counts the number of different atom types an atom is connected to, and then accumulates the results in a fingerprint vector. The proposed fingerprint is shown in Figure 3.2. Here, atom types are divided into subclasses by the number of free valencies an atom has, e.g, instead of just looking at how many carbon-carbon bonds are present, the fingerprint captures how many saturated carbon atoms are single bonded to a carbon with one free valence, or how many oxygens are double bonded to a saturated carbon atom and so on. This is a constant sized molecular descriptor as the length of the vector remains the same for smaller or bigger molecules.



C_0	C_1	C_2	C_3	O_0	O_1	C-H	C=O	O-H	C-O_0	C-O_1
2	1	0	0	1	1	4	1	0	0	1

C_0-C_0	C_0-C_1	C_0-C_2	C_0-C_3	C_1-C_1	C_1-C_2	C_1-C_3	C_2-C_2	C_2-C_3	C_3-C_3
0	2	0	0	0	0	0	0	0	0

Figure 3.2: Molecular fingerprint for the surface species $CH3CHCOO$. Here, C_0 denotes a saturated carbon (no free valence). C_1, C_2, and C_3 denote carbon atoms with one, two, and three free valencies, respectively. Similarly, O_0 is a saturated oxygen whereas O_1 is an oxygen atom with one free valence. The fingerprint vector (shown at the bottom of the image) contains the number of different saturated or unsaturated atoms, and the number of bonds between them.

As will be discussed later, this method works well for interpolation and works better than CM or BoB for extrapolation, but the extrapolation error was still quite large. Both, the flat molecular fingerprint and CM/BoB, can be fed to any regular ML model such as linear model, kernel based model or fully connected feed forward neural network. In each case, the ML model takes as input the molecular fingerprint vector and outputs the target real value (in our case, adsorption energy). We have also tried the extended connectivity based fingerprint (ECFP) [46] which produces fixed length vectors from the SMILES notations of the molecules.

3.2.3 Additive Subnetwork Model

The atomic fingerprint based additive subnetwork model is a size extensible model. The atomic fingerprint originally used [42] for this model were the symmetry func-

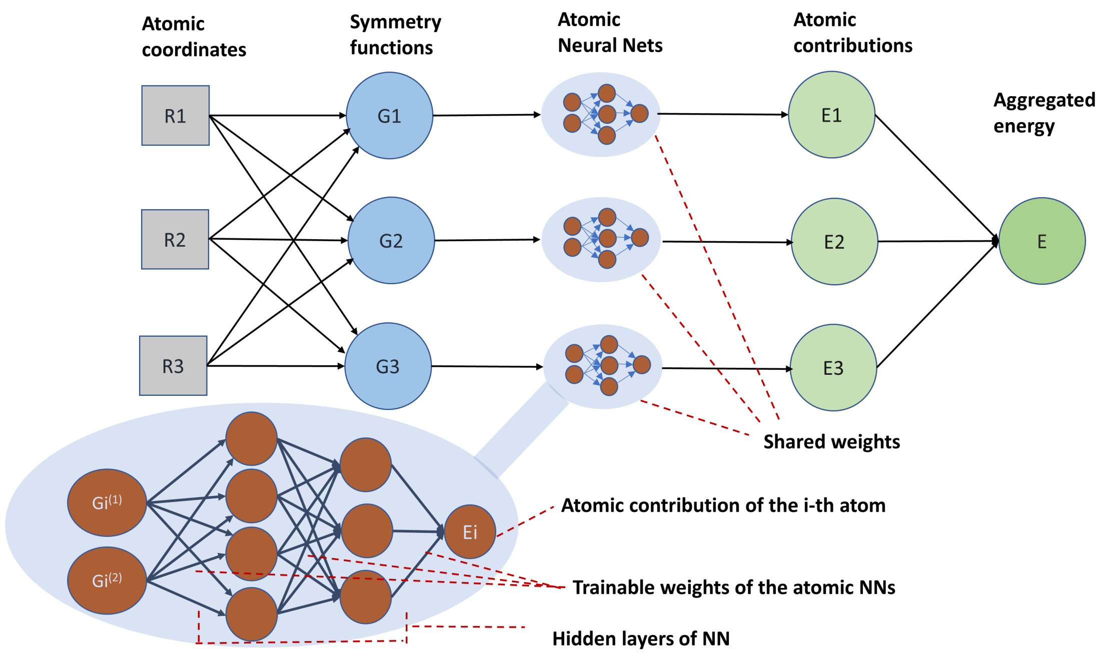
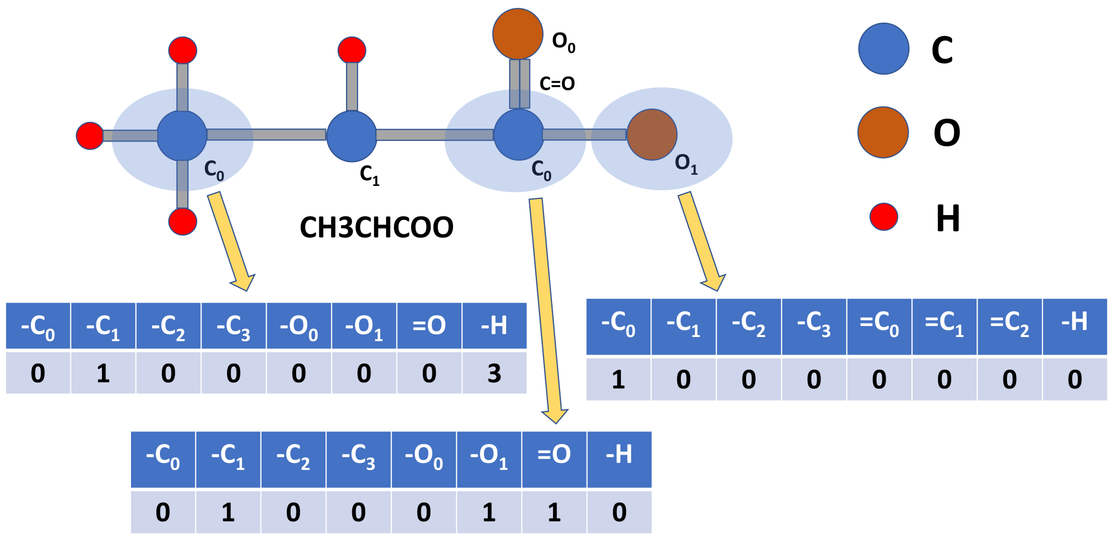


Figure 3.3: The network for the atomic contribution method. First, symmetry functions are calculated from the atomic coordinates of all the atoms in the molecule. Typically, two symmetry functions are used: one that aggregates the pairwise distance information centered around each atom; the other that combines the angular distance information from a triplet of atoms. Other symmetry functions can be devised, too. Here, Gi denotes the vector containing the symmetry function values for the i-th atom. For each atom, its corresponding G vector is fed to a neural network (NN). The networks for all the atoms share weights which makes the method to work with any ordering of atoms. Each atomic NN learns the energy contribution of the corresponding atom to the total energy of the species. All the atomic contributions are summed to get the predicted energy. The structure of the atomic NN can be adjusted as shown in the bubble at the bottom of the figure.

tions calculated from the atomic coordinates of all the atoms in the molecule. Two commonly used symmetry functions are: one that aggregates the pairwise distance information centered around each atom; the other that combines the angular distance information from a triplet of atoms (equations for these distance measures are given in the supporting information). Other symmetry functions can be devised, too. The model is shown in Figure 3.3.

Fingerprints for each atom are fed to a small neural network. These subnetworks learn the energy contribution of the current atom to the total energy as a function of



$-C_0$	$-C_1$	$-C_2$	$-C_3$	$-O_0$	$-O_1$	$=O$	$-H$
0	1	0	0	0	0	0	3

$-C_0$	$-C_1$	$-C_2$	$-C_3$	$=C_0$	$=C_1$	$=C_2$	$-H$
1	0	0	0	0	0	0	0

$-C_0$	$-C_1$	$-C_2$	$-C_3$	$-O_0$	$-O_1$	$=O$	$-H$
0	1	0	0	0	1	1	0

Figure 3.4: Our proposed non-coordinate based atomic fingerprints for the atomic subnet based method. These fingerprints can be obtained directly from the SMILES notation of the molecule without the need of any atomic coordinates. Three sample fingerprint vectors for two carbon atoms and one oxygen atom are shown. The vectors contain the information about the number of different types of bonds for an atom. For example, the vector item $-C_1$ contains the number of single bonds the current atom has with carbon atoms that contain one free valence. The 5th to 7th positions of the fingerprint vectors have different meaning for carbon and oxygen atoms, e.g, in the 7th position, for carbon, $= O$ denotes how many oxygen the current carbon atom is bonded to by double bonds, whereas for oxygen, $= C_2$ encodes the number of carbon with two free valencies that the current oxygen atom is connected to by double bonds.

the fingerprints. Aggregated energy contributions from all the subnetworks yield the final energy. Subnetworks for all the atoms of an atom type share their weights. The weights can also be shared across the atom types. We have found the latter approach to work better for our case. The weight sharing ensures that the model is invariant to the ordering of the atoms. In contrast to other models, this one can only work with neural networks because it gives the flexibility of a hierarchical structure through the use of the back-propagation method to learn the network weights. In order to avoid the computation of reliable coordinates for the prediction set, we prefer the SMILES based fingerprints. We have developed such an atomic fingerprint, shown in Figure 3.4, that is similar to the flat molecular fingerprint described above, but is centered on an atom and encodes the connectivity information for that atom. We have found this model to work better for extrapolation compared to CM, BoB or flat fingerprints, but the errors were still quite large which warranted a further improvements to the model.

3.2.4 Proposed Multiple Filter based Additive Subnetwork Model

To make the additive subnetwork based model generalize better to an unseen testing data, we propose to treat the shared weights of the atomic subnetworks as a filter in a convolutional neural network (CNN). This type of deep learning model is commonly used for image data where different sets of weights (called convolutional filter) scan through the image patches and learn to detect various basic image features such as edges or corners [55] which are combined in subsequent layers to detect higher level objects such as a face or a car or a digit [4]. These filters are analogous to the local receptive field in biological visual systems [56, 57]. In CNN, the filters are usually 2D or 3D matrices of weights. A filter is placed on a patch of the image and a cross-correlation operation between the filter weights and the input plane pixels is performed - this is the output of that filter for that image patch. Then, the filter is moved to the next adjacent patch (which may or may not overlap with the previous patch). A key observation here is that there is not one but a number of filters that are used because each filter learns different features (through back-propagation) [58]. Moving to our problem and the atomic subnetworks, we can think of a subnetwork as a filter in CNN. Since all of the subnetworks share their weights, they can be considered as one filter scanning each of the atoms in the molecule one by one.

Unlike CNN, however, in our case the subnetworks compute a non-linear function of its inputs instead of the cross-correlation, which makes sense since predicting adsorption energies is a regression problem and we want each subnetwork to learn the

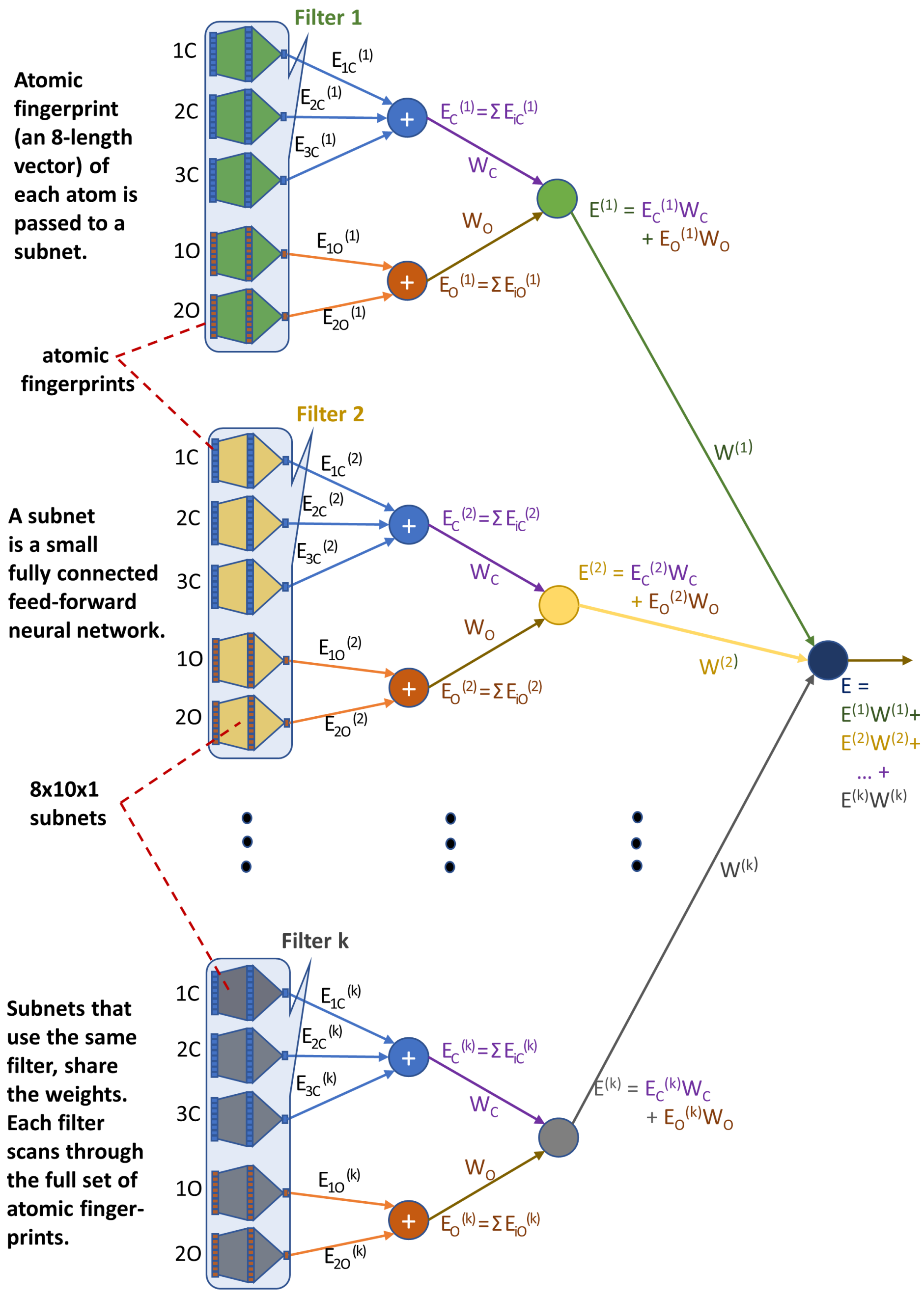

Filter 1
Atomic fingerprint (an 8-length vector) of each atom is passed to a subnet.
atomic fingerprints
Filter 2
A subnet is a small fully connected feed-forward neural network.
8x10x1 subnets
Filter k
Subnets that use the same filter, share the weights. Each filter scans through the full set of atomic finger-prints.
1C
2C
3C
1O
2O
E_C^(1) = Σ E_iC^(1)
E_O^(1) = Σ E_iO^(1)
W_C
W_O
E^(1) = E_C^(1)W_C + E_O^(1)W_O
E^(2) = E_C^(2)W_C + E_O^(2)W_O
E^(k) = E_C^(k)W_C + E_O^(k)W_O
W^(1)
W^(2)
W^(k)
E = E^(1)W^(1)+ E^(2)W^(2)+ ... + E^(k)W^(k)

Figure 3.5: Our proposed Model. A species with 3 carbon and 2 oxygen atoms is passed through k filters. Each filter is an 8 by 10 by 1 neural network. For each filter, outputs of networks for each of the three C atoms is summed up, same is done for the two O atoms. The weighted sum of these two sums is the output for one filter. The final output is the weighted sum of all the filter outputs. The parameters of the network that are learned through back-propagation are: $W^{(i)}$s, W_C, W_O, and network weights for each filter.

energy contribution of an atom. Also, unlike CNN, here we do not need multiple layers of filters as our learning objective is to find the individual atom contribution to the total energy.

However, the aspect of CNN that can be incorporated in our networks and that can lend the additive subnetworks a better representational ability is to use multiple filters instead of one. Here, we should clarify that using multiple filters does not mean using separate filters for different atom types. Whether we use different shared weights for different atom types or not, by 'multiple filters' we are referring to completely separate sets of filters (in each set, there may be one filter if all atom types share the weights, or more than one if weights are shared only inside each atom type). In our proposed model, each of the separate set of filters would scan each atom of the molecule and the weighted sum of all the filtered values should yield the final output energy.

The proposed model is shown in Figure 3.5. The atomic fingerprints are the 8-length vectors from Figure 3.4. During the training of the network, at each iteration of the gradient descent, there is a forward pass that starts from the fingerprints at the left of the figure and moves to the right. The gradient of the error in the energy obtained at the right-most node is then back-propagated through the network which makes it learn the appropriate weights to fit the data. Nonlinearity in the computation comes from the nonlinear activation functions used in the hidden layers of the filters. The error function (that the gradient descent tries to optimize) for neural networks is non-convex [59] - there can be many local minima. This means the output of a neural network is sensitive to the starting points of its weights - starting from different points in the hyperspace can result in different ending locations. Since the weights of each filter are randomly initialized [60, 61], each of them is likely to end up with different weights than others even though all of them are fed the same set of atomic fingerprints, i.e, each of them learns different functions of their inputs, just like each CNN filter learns to detect different image features. Let us assume there are K filters and the functions that they learn are denoted as $f^{(1)}, f^{(2)}, ..., f^{(K)}$ and the output of the i-th filter (after combining outputs of that filter for each atom in the species) is $E^{(i)}$. Then, the overall energy output of the networks, E is

$$E = \sum_{i=1}^{K} E^{(i)} W^{(i)} \tag{3.1}$$

where $W^{(i)}$ is the weight of the contribution for the i-th filter and

$$E^{(i)} = \sum_{a=1}^{T} E_a^{(i)} W_a \tag{3.2}$$

where T is the number of atom types in the species that have fingerprints (in our case, it is 2, for C and O; since those two atom types can describe all the bonds inside the species, we have not included fingerprints for H), $E_a^{(i)}$ and W_a are the summed contribution for all the atoms of atom type a when passed through filter i, and the weight for that atom type which is shared across the filters, respectively. So,

$$E_a^{(i)} = \sum_{n=1}^{N_a} E_{a_n}^{(i)} \tag{3.3}$$

where $E_{a_n}^{(i)}$ is the output when the n-th atom of atom type a is passed through filter i. The last equation means the atomic contribution of all the atoms for an atom type for a filter are directly summed and not weighted (which can be treated like a constant, non-learnable weight of 1). This ensures that a change in the relative ordering of the atoms (inside the set of atoms of an atom type) does not change the overall result. If the atomic fingerprint for the atom a_n is X_{a_n}, then $E_{a_n}^{(i)}$ is a non-linear function of X_{a_n}:

$$E_{a_n}^{(i)} = f^{(i)}(X_{a_n}) \tag{3.4}$$

Here, the fingerprint is passed to the filter which, in our case, is a small fully connected feed-forward neural network (NN). The output of each layer in the NN is computed by multiplying the weight matrix between the current and the previous layer with the output vector of the previous layer and then passing the obtained vector to a non-linear activation function [62, 63, 64].

Table 3.1: Interpolation results with following methods used: Coulomb matrix, bag-of-bonds, flat molecular fingerprint, and additive atomic subnetwork model (See discussions for details).

Method	ML model	MAE (eV)	SD of AE (eV)
Coulomb matrix	GP	0.230	0.218
Bag-of-bonds	KRR	0.139	0.136
Bag-of-bonds	Ridge	0.219	0.279
ECFP	SVR	0.165	0.179
Flat molecular fingerprint (from SMILES)	SVR	0.148	0.129
Flat molecular fingerprint (from SMILES)	KRR	0.141	0.122
Flat molecular fingerprint (from SMILES)	Ridge	0.196	0.166
Additive atomic subnetwork	ANN	0.398	0.202
Proposed model (from coordinates, 1 filter)	ANN	0.347	0.259
Proposed model (from coordinates, 4 filters)	ANN	0.309	0.231
Proposed model (from SMILES, 1 filter)	ANN	0.190	0.164
Proposed model (from SMILES, 6 filters)	ANN	0.142	0.120

3.3 Results and Discussion

Table 3.2: Extrapolation results with following methods used: Coulomb matrix, bag-of-bonds, flat molecular fingerprint, and additive atomic subnetwork model (See discussions for details).

Method	ML model	MAE (eV)	SD of AE (eV)
Coulomb matrix	SVR	2.392	1.015
Bag-of-bonds	KRR	2.046	0.422
ECFP	SVR	2.961	0.760
Flat molecular fingerprint (from SMILES)	KRR	2.342	0.625
Additive atomic subnetwork	ANN	0.441	0.214
Proposed model (from coordinates, 1 filter)	ANN	0.324	0.212
Proposed model (from coordinates, 4 filters)	ANN	0.282	0.190
Proposed model (from SMILES, 1 filter)	ANN	0.434	0.314
Proposed model (from SMILES, 5 filters)	ANN	0.227	0.143

In our investigations, we have used Coulomb matrix, bag-of-bonds, flat molecular fingerprints (non-coordinate based, calculated from the SMILES), and additive atomic subnetwork models for both interpolation and extrapolation. Key results are shown in Table 3.1 and Table 3.2, respectively. The supporting information contains full tables of all results. For interpolation, for the first seven rows, for each of the methods, we ran the following ML models: ridge regression, LASSO, kernel ridge regression (KRR), support vector regression (SVR), Gaussian processes (GP). The rest of the rows used artificial neural networks (ANN). Some of the results from each category are shown. The first and second columns show the descriptor method and the ML model used, respectively. The third column contains the mean absolute error (MAE) of the predicted adsorption energies, and the fourth column presents the standard deviations of the absolute errors. Data for the 247 C4 species were randomly permuted and 215 were used for training and the rest for testing. The process was repeated 100 times (data being permuted randomly each time) to obtain an unbiased estimate of the MAE. For extrapolation, for the first four rows, for each of the methods, we ran the following ML models: ridge regression, LASSO, kernel ridge regression (KRR), support vector regression (SVR), Gaussian processes (GP). The rest of the rows used artificial neural networks (ANN). Some of the best results for each method are shown. The first and second columns show the descriptor method and the ML model used, respectively. The third column contains the mean absolute error (MAE) of the predicted adsorption energies, and the fourth column presents the standard deviations of the absolute errors. Data of 247 C4 species were used for training and data of 29 C2 and C3 species were used for testing. Here, the table for interpolation shows that non-coordinate based molecular fingerprints with kernel based ML models perform as good as coordinate based descriptors with the same ML models. The additive atomic subnetwork based on SMILES with multiple filter also worked well. For this model, we also used a coordinate based atomic fingerprint of length 5 - aggregated pairwise distance measures from an atom to each of the four atom types involved (carbon, hydrogen, oxygen and top two layers of metal catalyst surface), plus the triplet distance measure.

For extrapolation, both the Coulomb matrix and bag-of-bonds performed poorly. This is not unexpected since these methods are not size extensible and require padded zeros to make them work for different sized molecules. From this point of view, the flat molecular fingerprint comes as an attractive alternative as it is a constant sized descriptor (size of the molecule does not effect the size of the vector; no zero padding is required). But our results show that it performs no better than CM or BoB for extrapolation. However, the method that we found to be most promising was the additive atomic subnetwork. Since this method adds up the atomic contributions to the total energy, it is naturally size extensible. The initial predictive error obtained using this method (with the length 5 fingerprint discussed above) was approximately 0.4 eV. As a neural network ends up in a different location of its parameter hyperspace on different runs (because of randomly initialized parameters), we ran the model multiple times and our final result was an ensemble of these runs - for each target species, its predicted adsorption energy was the mean of its predicted values of all the runs. This yielded an extrapolation error of approximately 0.32 eV. The ensemble method was used in all of our following models.

The next step was to replace the coordinate based atomic fingerprints with SMILES based ones (Figure 3.4). However, only replacing the atomic fingerprints in the additive subnetwork model actually increased the predictive errors to over 0.4 eV. We improved this model by using multiple filters as discussed before (Figure 3.5). The predictive errors went down significantly as more filters were used. The rate of improvement, however, gradually subsided and after incorporating a certain number of filters, the predictive errors change very little, as can be seen in Figure 3.6. We used this multi-filter approach with the coordinate based atomic fingerprints as well, and the extrapolation error there went down to 0.28 eV from over 0.32 eV.

Here, we should note that, the number of filters is essentially a hyperparameter to our model and needs to be tuned for specific problems. Tuning of hyperparameters for machine learning models is typically done by setting aside a portion of the training set as validation set and choosing the hyperparameters for which the model performs

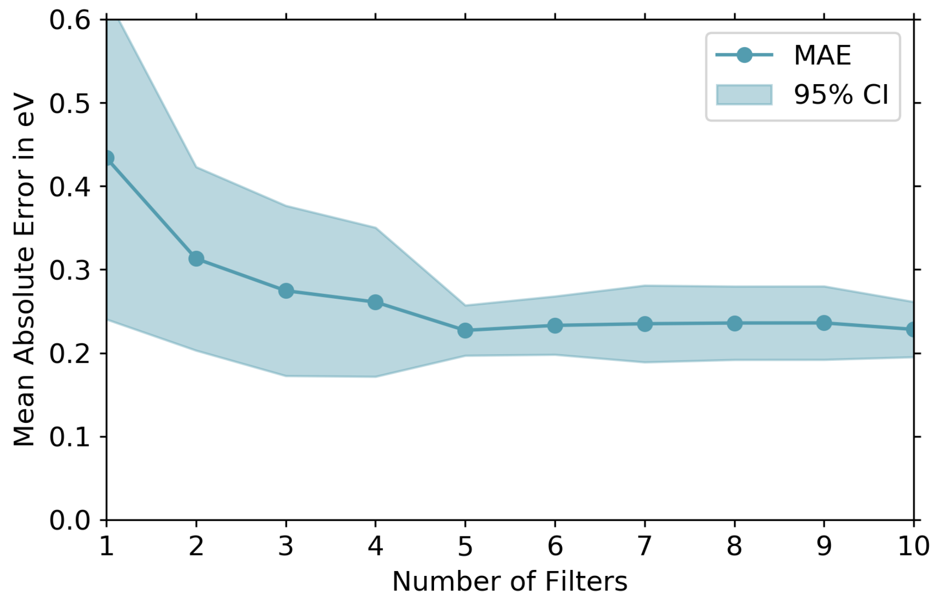


Figure 3.6: Extrapolation errors decreased sharply with the use of more filters. At one point, however, it reaches a state where adding more filters does not make any significant improvement. The model used here is our proposed model shown in Figure 3.5 and the atomic fingerprints were the ones shown in Figure 3.4.

best on the validation set. The chosen model is then used to run on the testing set. We also used this approach. This works well for interpolation problems where the training (which includes the validation set) and testing sets reside in the same region of the parameter space. But in case of extrapolation, this might not work.

Through our investigations, we have seen that for extrapolation, the error on the validation set (which is part of the training set, containing C4 species) went to an approximate minimum value when 6 filters were used and then remained more or less constant. But the extrapolation error on the testing set (containing C2 and C3 species), after the network was trained with different numbers of filters, reached minimum with 5 filters. This can occur for other 'pure extrapolation' settings where a low validation error does not always correspond to a low test error. In this case, if a small amount of data from the test space (which, in our case, were C2 and C3 species) can be obtained, that can be included in the validation set to tune the

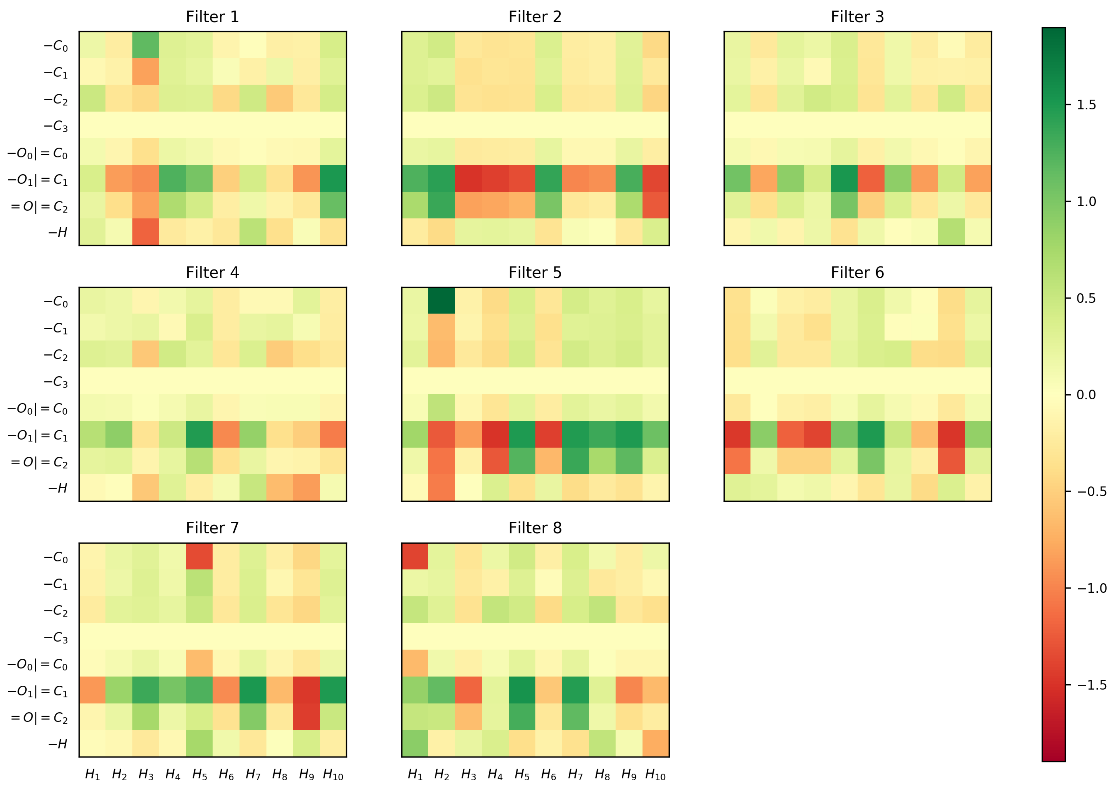


Figure 3.7: Weight matrices for the 8 filters learned by running our proposed model. Each of the eight 8-by-10 matrices contains the values of the weights that connect each of the 8 atomic fingerprint values (which, according to Figure 3.4 are the number of different bond types an atom is connected to) to the 10 hidden units denoted as H_1 to H_{10} (the left half of each subnetwork or filter shown in Figure 3.5).

hyperparameter more effectively. The problem setting, however, would no longer be a 'pure extrapolation' as small amount of data points from the test space are included during the training phase.

Figure 3.7 shows the values of the learned weights between the input layer and the hidden layer for each filter when 8 filters were used. For each matrix, a row corresponds to the 10 weights going out of one fingerprint value (see Figure 3.4). A column corresponds to the 8 values going into a hidden layer unit. There are three key observations here. First, each filer learns a different function of its inputs. Second, the sixth and seventh row contain weights with high absolute values. This is because the weights were shared across the atom types, i.e, fingerprints from carbon and oxygen atoms were fed to the same filters; and according to Figure 3.4, some of the entries at the same location of the fingerprint vector for carbon and oxygen carry a different meaning. According to this, the fifth, sixth, and seventh entries have to encode more information and hence the network learned higher valued weights for some of those. And finally, the fourth row for each of the filters learned close-to-zero weights. The fourth entry in the atomic fingerprint is the count of the number of carbon atoms that the current atom is bonded to where the carbon atom has three free valencies. In our training set, there was no such species. So, the network did not learn any significant value for those weights. This also signifies that if any species in the target set contains such a structure, its prediction would be inaccurate. Indeed, we found that two species, CH_2C and CH_3C (not included in the 29 species used as our target set), both containing this type of structure, had very high prediction errors (around 1 eV). This observation signifies a fundamental limitation in machine learning based models - the predictions can be at most as good as the data that is fed to train the model. The model learns from the training data the adsorption energy as a function of the basic building blocks or fragments that make up a chemical species, such as the information on how many free valencies an atom has, or how many unsaturated carbon atoms an atom is connected to, etc. The effectiveness of the model, hence, is dependent on how well the encoded structural information in the fingerprint can describe the differences between the target property (here, adsorption energy) among different chemical species. Fragments that are absent in the training data, such as aromatic rings, are not learnt by the model and thus, the model will likely fail for species containing such fragments.

3.4 Conclusion

In this paper, we have performed a detail investigation on a predictive model for both interpolation and extrapolation of adsorption energies of hydrocarbon species on Pt(111) catalyst surface. We have compared the effectiveness of different fingerprints and ML models. For interpolation, our results indicate that a simple SMILE based fingerprint calculated from nearest neighbors with kernel based ML models perform very well for interpolation of adsorption energies with an MAE of 0.14 eV. However, when predicting adsorption energies of species of different size from that of the training set (extrapolation), only an additive atomic contribution based model works reasonably well. To improve upon this method, we have developed a multi-filter based weighted additive model that combines the established additive model with the concept of filters from a convolutional neural network. Our findings show that this approach is highly generalizable compared to other models and leads for extrapolation of adsorption energies to an MAE of 0.23 eV. The proposed model also worked well when applied to interpolation with no statistically significant difference with the best models. The model has the potential to be applicable in other problems if the hyper-parameters of the model are adjusted according to the task. In the current work, all species were chain structures and of size between C1 and C4. The model was able to successfully extrapolate from larger to smaller species as long as the target species had similar chemical fragments as those in the training set. However, it should be noted that for distant extrapolation from small species to large species such as enzymes and proteins, the relative number of atoms bonded to the surface might be different and the model needs further refinement for such scenarios.

Chapter 4

An Accelerated Metropolis-Hastings based on Bayesian Optimization and Gaussian Process Approximation

4.1 Introduction

Markov Chain Monte Carlo (MCMC) algorithms, widely applied in numerous fields of science, engineering and statistics [65], use Markov chain to sample from a probability distribution. In this paper we focus on the Metropolis-Hastings (MH) algorithm, one of the premier algorithms under this class. MH Typically runs for a large number of iterations where, for each iteration, a new point on the parameter space is proposed using a proposal distribution; and the target function needs to be evaluated for the proposed point, based on which the algorithm decides whether to move to the proposed point or to stay in the same location. Thus, the target function needs to be evaluated many times. For many physics based models, the forward simulations are expensive and performing these for each iteration incur large computational cost and becomes a performance bottleneck. Moreover, the initial samples of the Markov chain usually follow a distribution that is different from the target, and have to be discarded until the convergence to the target - termed as burn-in period - which is wasteful. In this work, we propose an enhanced MH algorithm, named MHGP that addresses these problems.

Different approaches have been proposed to make MCMC faster [66] - using parallelization through multithreading [67] or distributed algorithm to achieve better performance [68]; or reducing the computational cost of the accept/reject step by using smaller fraction of data [69]; using log-likelihood estimator to work with random subset of the observations [70]; reducing dimensionality of the underlying computation model for efficient convergence [71]. These methods, while, trying to reduce the cost of the target evaluation, do not actually reduce the number of times the target is evaluated. Gaussian Process model has been proposed to off-load some of the computational work in Hybrid Monte Carlo [72, 73]; approximation methods have been proposed [74] where acceptance probabilities are calculated on a local approximation and the actual target is only evaluated once the proposal has been accepted. However, it would still require large number of target evaluations once the chain reaches high density region as more of the proposals are accepted there. Gaussian Process approximation of the target distribution was used to decrease expensive function calls [75]. But, as more numbers are added to the GP, the computational cost increases as the time complexity for GP is $O(N^3)$. Methods such as sparse Gaussian process [76] can be used to limit these computational overheads. To improve the proposal distribution in MCMC, adaptive approach has been used [77, 78] where information from simulation is utilized to adapt the proposal distribution. A multi-step proposal distribution was proposed [79] which discusses an adjustable proposal to speed up convergence. The adaptive approach has the limitation that for high dimensional space the stationary distribution tends to be biased. Although there exist many nonlinear low-dimensional models, this certainly reduces the domain of the set of problems where this approach can be applied.

In this paper we propose an enhanced MH method. It uses Bayesian optimization to speed up the burn-in process and quickly reach high density region. Next, continuing with the GP obtained from the Bayesian optimization, Laplace approximation of the GP is taken around the peak to get the covariance matrix for an informed proposal distribution; and then, guided by this proposal, sample-generating iterations run as more training points are added to the GP, which continues to gain better approximation of the unnormalized target distribution. Due to positivity of probability density functions (pdf), GP is used to approximate the log of the target pdf instead of using it directly to the original pdf. The uncertainty measure of the GP predictions provides the uncertainty for the acceptance rate, which is then used to decide whether the objective function needs to be evaluated or it can be read from the GP approximation - resulting in fewer forward simulations as the iterations progress. Local Gaussian process [80, 81] was used to avoid expensive calculations involving cumulative sampled points. The proposed algorithm was evaluated for different benchmark problems, two of which are presented here. The obtained samples are compared with those from plain MH and DRAM methods which reveal that samples from MHGP have no statistically significant difference from the established methods but is able to achieve that with far fewer target evaluations.

4.2 Methodology

4.2.1 Vanilla Metropolis-Hastings Algorithm

The pseudocode for the plain Metropolis-Hastings algorithm [82] is given in Algorithm 1.

There are some points to note here. First, the target distribution $p(x^*)$ has to be computed N times. The iteration number N is usually quite large - in the range of hundreds of thousands or millions. The quality of the generated sample improves with the number of iterations. So for complex target distribution the value of N has to be large. Each of these iterations requires the objective function to be evaluated. Evaluating this involves the computation of the likelihood function, which, for large data sets, such as physics-based model simulations, is computationally expensive [73].

Algorithm 1 Plain Metropolis-Hastings algorithm

```
1: procedure Metropolis–Hastings
2:     Initialize $x^{(0)}$
3:     for $i = 0$ to $N$ - 1 do
4:         Sample $u \sim U_{[0,1]}$
5:         Sample $x^* \sim q(x^*|x^{(i)})$
6:         $acceptanceRatio = \frac{p(x^*)q(x^{(i)}|x^*)}{p(x^{(i)})q(x^*|x^{(i)})}$
7:         $A(x^{(i)}, x^*) = min\{1, acceptanceRatio\}$
8:         if $u < A(x^{(i)}, x^*)$ then
9:             $x^{(i+1)} = x^*$
10:        else
11:            $x^{(i+1)} = x^{(i)}$
12:        end if
13:    end for
14: end procedure
```

So it is desirable to keep the number of costly function evaluation to minimum while still obtaining good quality sample set. Secondly, the proposal distribution used with the plain MH algorithm does not reflect any knowledge obtained from the structure of the distribution. A better proposal distribution is bound to propose samples from more relevant regions and thus achieving better results. Lastly, although the Markov chain eventually converges to the target distribution, the initial samples can follow quite a different distribution, more so in the case of a starting point chosen in the low density region of the function space. This requires a number of initial samples to be thrown away, which is quite wasteful if each iteration has heavy computational expense. Our proposed method, MHGP, addresses these issues.

4.2.2 Starting with Bayesian Optimization

MHGP starts with initiation of a Gaussian Process for Bayesian Optimization, which is a sequential approach to optimize an objective function by balancing between exploitation and exploration, controlled by an acquisition function [83, 84]. The Gaussian process provides uncertainty estimates on the parameter space. The regions of high uncertainty or exploration, and the regions of high model estimate or exploitation, are balanced using the acquisition function. In our case, it enables MHGP to reach the high density region of the target propability distribution with a handful number of iterations.

4.2.3 An Informed Proposal Distribution

The optimized point and the GP provided by Bayesian optimization is used in the next step to come up with an informed proposal distribution that captures and approximate shape of the target distribution. We calculate Hessian of the GP at the mode of the distribution to apply Laplace approximation and thus obtain a multivariate Gaussian distribution with mean at the optimized point and a covariance matrix that we will use as the covariance of our proposal distribution for the following stages.

As Bayesian Optimization often requires only a few steps to reach the optimized region, the Gaussian Process may not be good enough to approximate the target distribution at the end of the first phase and the covariance of the proposal may not be positive semi-definite. In order to get a better approximation, a random walk, governed by some isotropic Gaussian proposal, is initiated starting from the optimized point obtained in the first phase of the algorithm. New points are added to GP by evaluating the objective function. The number of steps this needs to go on can be pre-specified or can be adaptively controlled by checking the uncertainty in the GP estimation on subsequent steps.

4.2.4 Sample Generation using Gaussian Process

Next, MHGP enters its sample generating iterations. At each iteration, a new point is proposed centering at the current point with a covariance obtained from the previous phase. If the GP prediction at the proposed point has high uncertainty, it needs to be evaluated and added to the GP training set, otherwise it is read from GP. A high-level pseudo-code of MHGP is presented in Algorithms 2 and 3.

If x is the last sampled point and x^* is the new proposed point that the proposal distribution has given, the acceptance probability, a can be written as:

$$a = \frac{p(x^*)}{p(x)}$$
$$\ln a = \ln p(x^*) - \ln p(x)$$

We build our GP on the log of the target pdf. So each of the values $\ln p(x^*|D)$ and $\ln p(x|D)$ are Gaussian distributed. This means if we take the log of the acceptance probability, it is the subtraction of two Gaussian which makes it Gaussian, too.

Thus we have: $\ln a \sim \mathcal{N}(\mu, \sigma^2)$.

This makes the probability of the acceptance rate a log-normal distribution: $a \sim \ln \mathcal{N}(\mu, \sigma^2)$.

As new points are evaluated for the target posterior distribution they are added to the training set of the GP. Each time a new point x^* is proposed from the proposal distribution, we measure how certain our GP is about the acceptance probability there. The measurement is done by computing $\sqrt{Var[a]}/E[a]$ and see if it is larger than some threshold value. Based on the value of the computation being larger than the threshold or not, we decide whether to read the $p(x^*)$ from the GP or to evaluate the target distribution.

Algorithm 2 Proposed MHGP Method

```
1: procedure MHGP
2:     Initialize $x^{(0)}$
3:     Run *BayesOpt* starting from $x^{(0)}$; it returns $GP$
4:     Set proposal distribution $Q$ to be the covariance obtained from Laplace approximation of $GP$
5:     Set $x^{(0)}$ to be the optimized point
6:     for $i = 0$ to $N$ - 1 do
7:         Sample $u \sim U_{[0,1]}$
8:         Sample $x^* \sim Q(x^*|x^{(i)})$
9:         $\mu, \Sigma = \textsc{getTargetValue}(x^*, x^{(i)})$
10:        $acceptanceRatio = e^{\mu^* - \mu + \frac{\sigma^2_{xx} + \sigma^2_{x^*x^*} - 2\sigma^2_{xx^*}}{2}}$
11:        $A(x^{(i)}, x^*) = min\,\{1, acceptanceRatio\}$
12:        if $u < A(x^{(i)}, x^*)$ then
13:            $x^{(i+1)} = x^*$
14:        else
15:            $x^{(i+1)} = x^{(i)}$
16:        end if
17:    end for
18: end procedure
```

Algorithm 3 Get Predicted or Evaluated Target Value

```
1: procedure GETTARGETVALUE($x^*, x^{(i)}$)
2:     $\mu, \Sigma = LocalGP(predict(x^*, x^{(i)}))$
3:     if $\sqrt{e^{\sigma^2_{xx}+\sigma^2_{x^*x^*}-2\sigma^2_{xx^*}} - 1} > threshold$ then
4:         if $p(x)$ was evaluated then
5:             Evaluate $p(x^*)$
6:             add to GP training set
7:         else
8:             Evaluate $p(x)$
9:             add to GP training set
10:            if ratio still greater than threshold then
11:                Evaluate $p(x^*)$
12:                add to GP training set
13:            else
14:                get $p(x^*)$ from LocalGP's prediction
15:            end if
16:        end if
17:    else
18:        get $p(x^*)$ from LocalGP's prediction
19:    end if
20:    Return predicted or evaluated $p(x^*)$
21: end procedure
```

To calculate $\sqrt{Var[a]}/E[a]$ for the log-normal distribution, we use the standard mean and variance formula for log-normal, which are $e^{\mu+\sigma^2/2}$ and $(e^{\sigma 2} - 1)e^{2\mu+\sigma^2}$, respectively. This gives us $\sqrt{e^{\sigma 2} - 1}$ as our desired ratio. So we need to calculate the value of σ^2. For current point x and new proposed point x^* the GP prediction gives us the covariance matrix containing $\sigma^2_{xx}, \sigma^2_{x^*x^*}, \sigma^2_{xx^*}$ where σ^2_{xx} is the mean-squared error at point x, $\sigma^2_{x^*x^*}$ is the mean-squared error at point x^* and $\sigma^2_{xx^*}$ is the covariance between x and x^*. Since $\ln a$ was a subtraction of two Gaussian, the value of σ^2 for the log-normal will be $\sigma^2_{xx} + \sigma^2_{x^*x^*} - 2\sigma^2_{xx^*}$. As the acceptance ratio in our case is a log-normal random variable, we use its mean as the measure for the acceptance ratio. The mean formula of $e^{\mu+\sigma^2/2}$ for log-normal distribution gives $e^{\mu^*-\mu+\frac{\sigma^2_{xx}+\sigma^2_{x^*x^*}-2\sigma^2_{xx^*}}{2}}$.

GP regression is a function approximation process that not only predicts the function but also gives a measure of the uncertainty associated with the prediction [85]. For each of the desired prediction points we get the mean prediction and also the measure of how certain the algorithm is about its mean prediction. As more data points are added to the training set, the GP model becomes more certain about the structure of the function. It is more certain in the area where more training points reside and less certain where there are few training points. GP can use any kernel for the similarity measure provided that the kernel satisfies some condition [85]. Here we have used the well known squared exponential (SE) kernel. To limit the time required to train the GP on all the accepted points after each evaluation, we instead used local Gaussian process before making a prediction that considered only the points in the vicinity of the current and the proposed points. The covariance for the proposal (obtained from Laplace approximation) was scaled down by a configurable parameter.

4.2.5 Approximate Detailed Balance

MHGP is an approximate MCMC sampling technique since it uses the GP approximation of the target. So we cannot use the regular detailed balance property and prove mathematically that the generated sample chain converges to the target. Instead we use a new approximate detailed balance equation to show that MHGP converges to the GP approximation of the target. This proof follows the proof of the property for the standard MH [86]. The plain MH creates a Markov chain while generating samples from $p^*(x)$. The detailed balance property, which a chain satisfies if $p(x'|x)p^*(x) = p(x|x')p^*(x')$, is a sufficient condition for p^* to be the stationary distribution of the chain. MHGP generates samples from the GP-approximation $f^*(x)$ of the original target distribution $p^*(x)$. So the condition for the chain to satisfy detailed balance property, after replacing the target distribution with the approximation and rearranging, will be:

$$\frac{f^*(x')}{f^*(x)} = \frac{p(x'|x)}{p(x|x')}$$

MHGP algorithm deals with the mean of the ratio of $f^*(x')$ and $f^*(x)$ as the aspect ratio. So what we really want to show is the expected value of the ratio of $f^*(x')$ and $f^*(x)$ equals the right hand side of the last equation:

$$E\left[\frac{f^*(x')}{f^*(x)}\right] = \frac{p(x'|x)}{p(x|x')} \tag{4.1}$$

The argument for this to work is that given enough training points the Gaussian Process would converge to the target distribution and the uncertainty around the mean would be lower. To prove the equation 4.1 let us define $r(x'|x)$ to be the probability of accepting x' given that this point has been proposed when the chain was at x; and $\alpha(x'|x)$ to be

$$\alpha(x'|x) = \frac{f^*(x')q(x|x')}{f^*(x)q(x'|x)}$$
$$E\left[\alpha(x'|x)\right] = E\left[\frac{f^*(x')}{f^*(x)}\right]\frac{q(x|x')}{q(x'|x)}$$

Assuming $f^*(x)q(x'|x) < f^*(x')q(x|x')$ (proof for the opposite is symmetric), we have $r(x'|x) = E\left[\alpha(x'|x)\right]$ and $r(x|x') = 1$. The probability of going from current point x to another point x' is the probability of proposing that point times the probability of accepting the proposed point:

$$
\begin{aligned}
p(x'|x) &= q(x'|x)r(x'|x) \\
&= q(x'|x)E\left[\frac{f^*(x')}{f^*(x)}\right]\frac{q(x|x')}{q(x'|x)} \\
&= E\left[\frac{f^*(x')}{f^*(x)}\right]q(x|x') \qquad (4.2)
\end{aligned}
$$

The backward probability is given by $p(x|x') = q(x|x')r(x|x') = q(x|x')$. Putting this result in equation 4.2 and rearranging brings us to equation 4.1. Thus we can say MHGP would converge to the mean of the GP.

4.3 Experimental Results

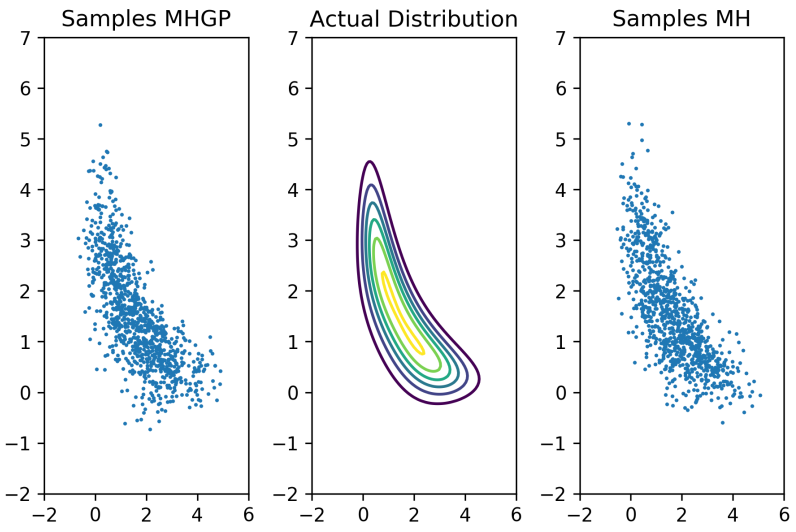


Figure 4.1: Random samples taken from the generated samples for both MHGP and plain MH along with the actual banana distribution in the middle.

The comparison between the plain MH and the proposed MHGP was done using a number of experiments, Three of which are presented here. The model for the first experiment was the banana distribution. Both the algorithms were run 15000 iterations to generate samples from the same banana distribution. For plain MH, each iteration needs one target evaluation. MHGP, in contrast, needed less than 200 evaluations

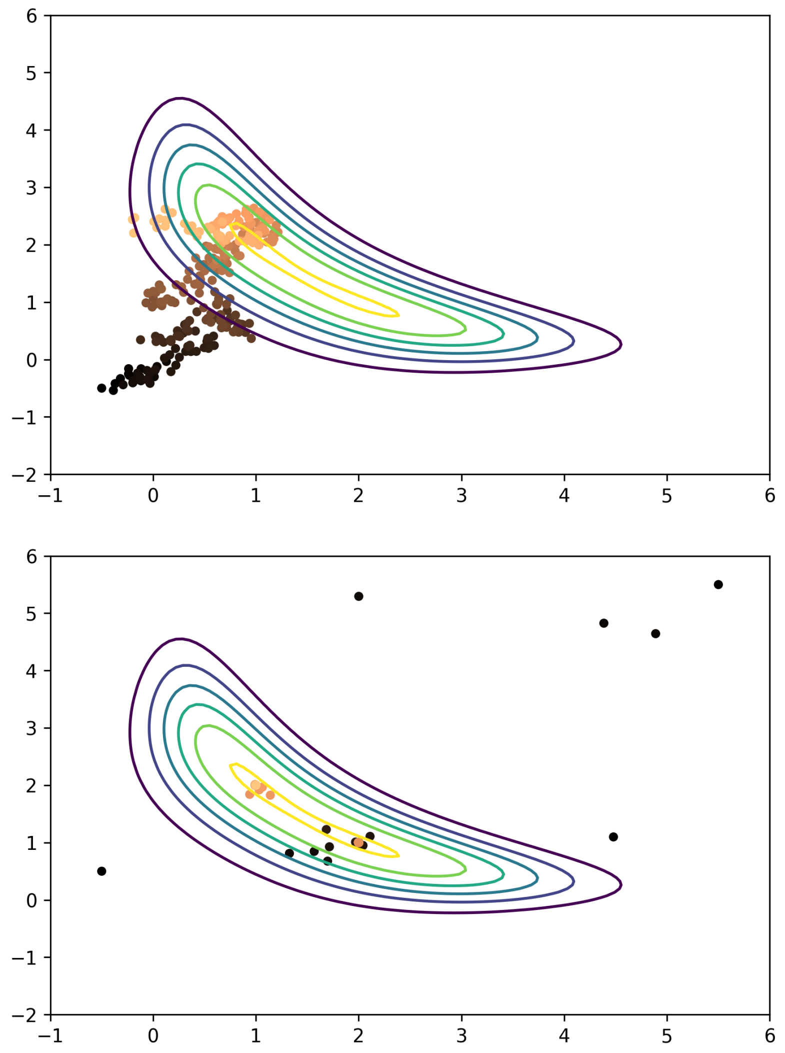


Figure 4.2: Comparison between time taken to reach high density region for plain MH versus the Bayesian optimization phase of MHGP. Top: showing first 250 iterations of plain MH for sampling banana distribution. Bottom: points explored by Bayesian optimizer for the same problem.

during all these iterations with 50 additional evaluations during the Bayesian optimization phase. 500 random samples generated by both the methods along with the actual distribution, shown in Figure 4.1, illustrates that MHGP achieves very similar results as MH but with far less computational cost. The performance gain from using Bayesian Optimization is evident from Figure 4.2. Both the algorithms started far from the high density region. The plot shows that plain MH required significantly larger number of evaluations compared to the Bayesian optimizer in MHGP.

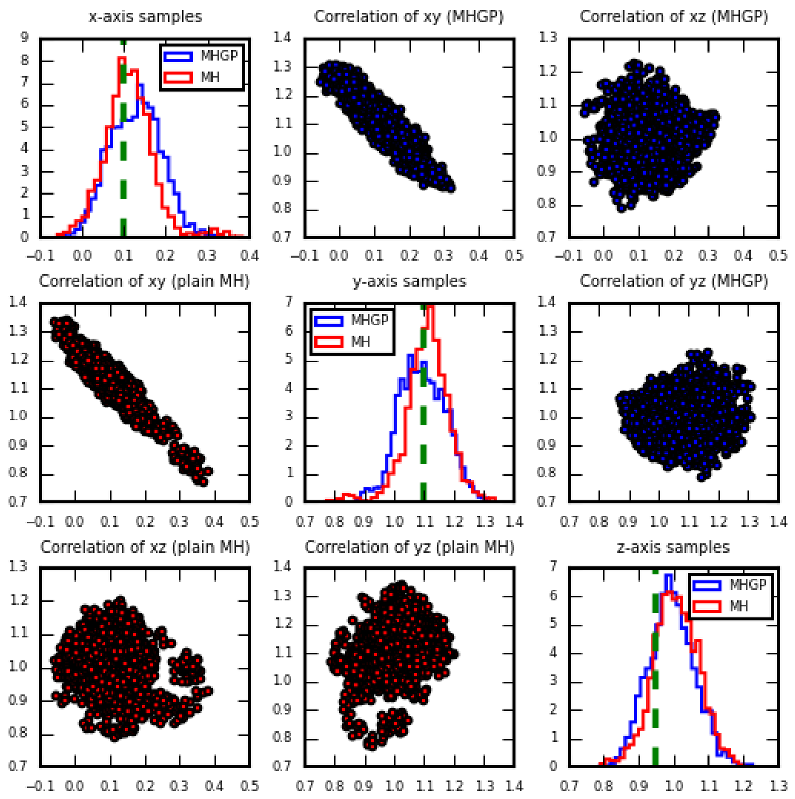


Figure 4.3: Stacked view of results from plain Metropolis-Hastings and the proposed MHGP. The green dotted lines show the actual values. The proposed model gives similar result to the original one but with fewer calls to the target distribution. Plain MH had 10,000 calls whereas MHGP had less than 300 calls to reach the same result.

The second example is a Lorenz system [87]. Starting with an initial condition and parameters the system reaches some new location after some specified amount of time. The model consists of three ordinary differential equations:

$$\frac{dx}{dt} = s(y-x) \quad , \quad \frac{dy}{dt} = x(p-z) - y \quad , \quad \frac{dz}{dt} = xy - bz$$

where x, y, and z together are the system state, t is the time, and s, p, and b are the system parameters. Our goal is to infer the initial location based on the data for the final location using prior and likelihood. We can model the system like this:

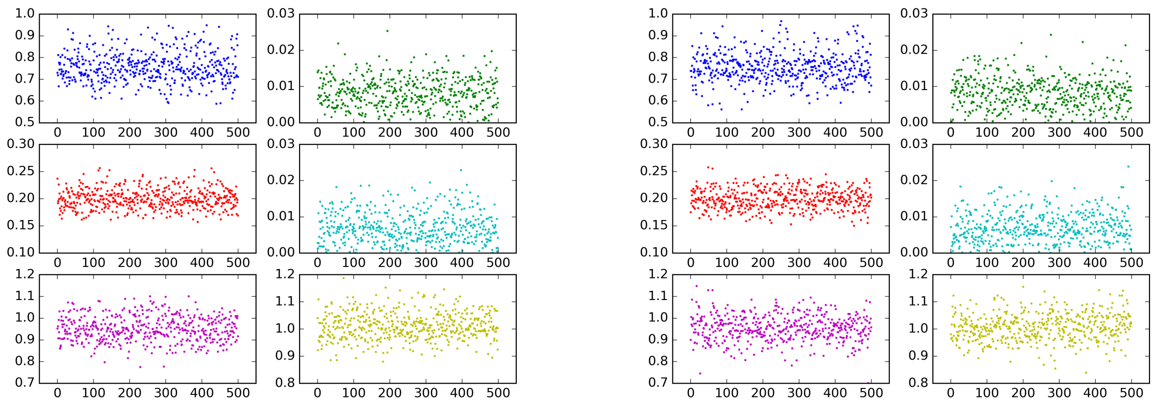


Figure 4.4: Samples along each dimension of the kinetics example. Left: samples generated by MCMC DRAM and Right: samples from MHGP. For both sides, the six plots read from left to right and top to bottom are for the first reaction rate, first activation energy, second reaction rate, second activation energy, and A's initial concentration for the first and second batch, respectively.

$$\begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} = L^t \left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \right) + \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \end{bmatrix}$$

where L^t stands for the Lorenz system running for time t; x_f, y_f, z_f make up the initial coordinate; x_f, y_f, z_f make up the final coordinate after the system is run for time t. The ϵ_x, ϵ_y, and ϵ_z are the error measures with each of them being normally distributed with zero mean and some variance which, in our experiment, was 0.01. We specify some prior which contains our prior knowledge of the initial location, if any. The likelihood measures the likelihood of the value of the final location given the starting location. We sample from the posterior distribution which we get from the prior and the likelihood. We do this for both the plain MH algorithm and our proposed enhanced MHGP algorithm. The results from both the algorithms showed similarity but with fewer number of calls to the target distribution in the enhanced algorithm (Figure 4.3). The figure has nine plots. The diagonal plots show the comparison of the samples between plain MH and the MHGP along each of the axes using histograms. The non-diagonal plots contain scatter plots showing the pairwise correlation among the samples for each of the axes. Plots in the upper triangle are from the MHGP, plots from the lower triangle (each of which has its symmetric counterpart in the upper triangle) are from the plain MH. As this is a very chaotic system, the samples for both the algorithms are somewhat off-target but again they show the similar trend. And again the important point is the fact that MHGP required much less than one-tenth of the function evaluation required by plain MH to achieve very similar result.

The Model for the third example was more complex real world ODE system problem of chemical kinetics. Here, a two step reaction A $\longrightarrow$ B $\longrightarrow$ C was considered with temperature dependent reaction rates. The dataset consists of two batches of data for two temperature settings where both the batches contain the relative concentrations of A and B over different time steps. There are six unknowns in the model: two reaction rate parameters, two activation energies, and for both the batches - the initial concentration of A. The priors for all the unknowns were set to be uniformly distributed in the acceptable range of values. The likelihood function measures the likelihood of the observed concentrations of A and B given the model prediction. For both MHGP and MCMC DRAM, 500 random samples out of the accepted samples along each of the six dimensions (unknowns) are shown in Figure 4.4. The plots at the top row show the distribution of samples for the reaction rate parameter and the activation energy for the first reaction, the middle row shows these parameters for the second reaction, and the bottom row presents the initial concentrations of A for both the batches.

The samples from MHGP again has similar distribution to that of the established method. We performed statistical test based on the energy distance [88] measures between the two sets of samples generated by the two methods to find if the samples indeed come from the same distribution. The energy distance test works by first obtaining pooled samples from both sets and calculate their energy distance ε_n. Then resample from these pooled samples some number of times, calculating the energy distance ε_m each time. For a desired significance level α, the null hypothesis is rejected if ε_n exceeds $1 - \alpha$ of ε_ms. No statistically significant difference was found between the two sets with p-value of 0.12 for the kinetics example, 0.15 for the banana distribution, and 0.16 for the Lorenz system. Note, here the null hypothesis is that there is no statistically significant difference between the two sets of samples and hence, high p-values mean that the energy distance test did not find any significant difference between the samples obtained from MHGP and existing methods.

The fact that MHGP, driven by the uncertainty measurement from Gaussian process, requires less and less target evaluation as the algorithm advances through the iterations, can be observed in Figure 4.5. GP starts with high uncertainty and many of the initially proposed points need to be evaluated. But gradually it gains a better approximation of the target and very few evaluations are needed in later stages.

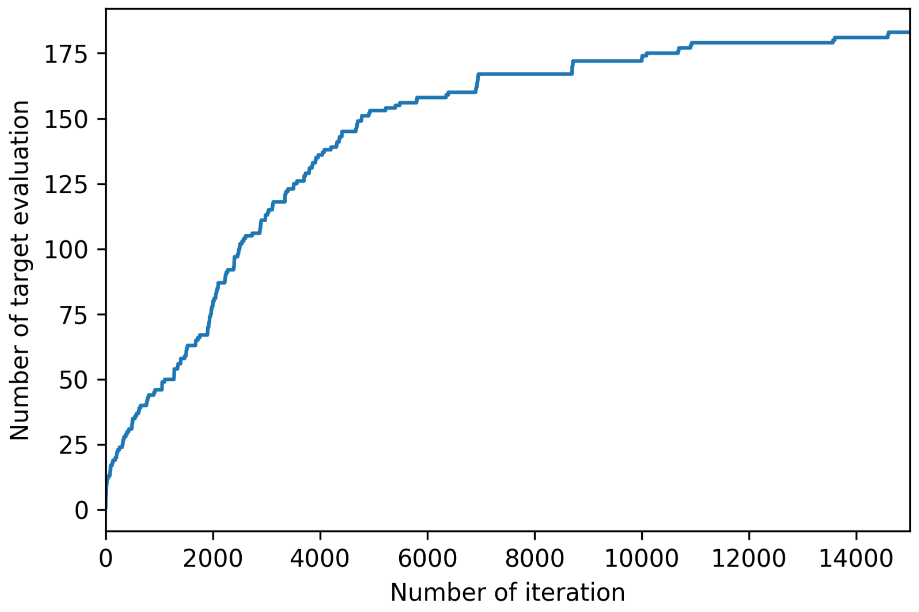


Figure 4.5: Target evaluations of MHGP.

4.4 Conclusion

The key challenge in this work was to reduce the number of costly evaluations while ensuring efficient convergence to the target distribution. As our experiments and corresponding comparative study have indicated, MHGP offers an efficient alternative to the plain Metropolis-Hastings. It has short burn-in period with the help of Bayesian optimization, an informed proposal distribution using Laplace approximation, and fewer target evaluations due to Gaussian process with quantified uncertainty. The method is based on the Metropolis-Hastings algorithm and hence, will face the same challenges as the original algorithm when faced with multi-modal distributions, for example. Moreover, as Gaussian process is used to approximate the target, the limitations of GP [89] on handling very high dimensional data can affect the method. Making GP to better handle this kind of scenario is an active research area [90, 91, 92]. Also, since the method is based on a GP approximation of the target, adherence to the detailed balance property cannot be established. Nevertheless, we believe the method can have significant practical value for different areas of science and engineering where forward simulations are expensive.

Chapter 5

Conclusion

Computational catalysis is an active research area with huge economic and environmental impact. In the current work, we have investigated, formulated and proposed improvements to the catalyst discovery workflow using machine learning. We have identified key bottlenecks in the conventional process of computational catalyst screening and tried to improve upon those. The regular process involves computation of complete database containing the adsorption and transition state energies for all intermediate species on different surfaces; then using these to identify dominant catalytic cycles, rate controlling steps and key reaction intermediates; which in turn are used to develop microkinetic models and perform the forward problem of predicting uncertainties for quantities-of-interests such as turnover frequency; and finally, these uncertainties are refined by Bayesian inverse problem using the experimental measurements on QoIs. Our work has focused on reducing the computation of adsorption energies and accelerating the MCMC method used in inverse problems.

In order to reduce the number of expensive density functional theory (DFT) calculations, we have built a predictive framework for adsorption energies. Our initial investigation focused on comparing non-linear machine learning models with linear scaling. We found that linear scaling works well when predicting energies for species on some metal surface given energies for those species on other metal surfaces. Our studies showed that appropriate choice of metal descriptors was important in this case. To do this, we proposed an automatic discovery process for metal descriptors. We also found that when our model used an incomplete energy dataset for training, species descriptors need to be included and advanced ML models outperform linear models in this case. This study has been published in Ref. [7]. Next, we investigated the case when we have data for a set of species for one metal surface and the goal is to predict energies for other species on the same surface. In this case, learning is done solely based on the species descriptor. Also, we wished to extrapolate on adsorption energies - when training set contains small species and test set contains larger species and vice versa. Our experiments suggested that regular machine learning models did not achieve satisfactory results for this type of prediction. We developed a novel multiple filter based neural network model that has been shown to outperform traditional models for extrapolation of adsorption energies. This work has also been published in Ref. [8].

To accelerate the Bayesian inverse problem, that is refining the uncertainties of the quantities-of-interests based on experimental measurements, we have proposed an improved Metropolis-Hastings MCMC algorithm that reduces the number of forward simulations containing expensive likelihood function calculations by using an approximate model of the target function using Gaussian process. We compared our method with standard MH for several simple models and this initial work had been published in a conference proceedings [75]. The work has since been enhanced. It focuses on reducing the burn-in period using Bayesian optimization and introduces a new and improved proposal distribution that uses Laplace approximation through the Hessian of the Gausian process. The method has been tested on larger, real world problems and the experiments indicate that it is able to achieve similar quality of samples as the original method, but with far fewer target evaluations.

Bibliography

[1] Y. Jing, Y. Bian, Z. Hu, L. Wang, and X. S. Xie, "Deep Learning for Drug Design: an Artificial Intelligence Paradigm for Drug Discovery in the Big Data Era," *AAPS Journal*, vol. 20, no. 3, p. 58, 03 2018.

[2] M. W. Libbrecht and W. S. Noble, "Machine learning applications in genetics and genomics," *Nature Reviews Genetics*, vol. 16, pp. 321 EP –, May 2015, review Article.

[3] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[5] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger, "From word embeddings to document distances," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 957–966.

[6] J. K. Nørskov, F. Studt, F. Abild-Pedersen, and T. Bligaard, "Fundamental concepts in heterogeneous catalysis," in *Fundamental Concepts in Heterogeneous Catalysis*. Hoboken, New Jersey: John Wiley and Sons, 2014, ch. 2, pp. 17–19.

[7] A. J. Chowdhury, W. Yang, E. Walker, O. Mamun, A. Heyden, and G. A. Terejanu, "Prediction of adsorption energies for chemical species on metal catalyst surfaces using machine learning," *The Journal of Physical Chemistry C*, vol. 122, no. 49, pp. 28 142–28 150, 2018.

[8] A. J. Chowdhury, W. Yang, K. E. Abdelfatah, M. Zare, A. Heyden, and G. A. Terejanu, "A multiple filter based neural network approach to the extrapolation of adsorption energies on metal surfaces for catalysis applications," *Journal of Chemical Theory and Computation*, vol. 16, no. 2, pp. 1105–1114, 2020, pMID: 31962041.

[9] J. K. Nørskov, F. Abild-Pedersen, F. Studt, and T. Bligaard, "Density functional theory in surface chemistry and catalysis," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, no. 3, pp. 937–943, 2011.

[10] M. Busch, M. D. Wodrich, and C. Corminboeuf, "Linear Scaling Relationships and Volcano Plots in Homogeneous Catalysis - Revisiting the Suzuki Reaction," *Chemical Science*, vol. 6, no. 12, pp. 6754–6761, Dec 2015.

[11] J. Lu, S. Behtash, M. Faheem, and A. Heyden, "Microkinetic modeling of the decarboxylation and decarbonylation of propanoic acid over pd(111) model surfaces based on parameters obtained from first principles," *Journal of Catalysis*, vol. 305, pp. 56 – 66, 2013.

[12] F. Abild-Pedersen, J. Greeley, F. Studt, J. Rossmeisl, T. R. Munter, P. G. Moses, E. Skúlason, T. Bligaard, and J. K. Nørskov, "Scaling properties of adsorption energies for hydrogen-containing molecules on transition-metal surfaces," *Physical Review Letters*, vol. 99, p. 016105, Jul 2007.

[13] D. Lopez-Paz, S. Sra, A. J. Smola, Z. Ghahramani, and B. Schölkopf, "Randomized nonlinear component analysis," in *Proceedings of the 31st International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–1359–II–1367.

[14] H. Abdi, "Factor rotations in factor analyses," in *Encyclopedia of Social Science Research Methods.* Thousand Oaks, California: Sage Publications, 2003.

[15] M. Szaleniec, M. Witko, R. Tadeusiewicz, and J. Goclon, "Application of artificial neural networks and dft-based parameters for prediction of reaction kinetics of ethylbenzene dehydrogenase," *Journal of Computer-Aided Molecular Design*, vol. 20, no. 3, pp. 145–157, Mar 2006.

[16] J. Behler and M. Parrinello, "Generalized neural-network representation of high-dimensional potential-energy surfaces," *Physical Review Letters*, vol. 98, p. 146401, Apr 2007.

[17] F. Pereira, K. Xiao, D. A. R. S. Latino, C. Wu, Q. Zhang, and J. Aires-de Sousa, "Machine learning methods to predict density functional theory b3lyp energies of homo and lumo orbitals," *Journal of Chemical Information and Modeling*, vol. 57, no. 1, pp. 11–21, 2017, pMID: 28033004.

[18] M. Rupp, R. Ramakrishnan, and O. A. von Lilienfeld, “Machine learning for quantum mechanical properties of atoms in molecules,” *The Journal of Physical Chemistry Letters*, vol. 6, no. 16, pp. 3309–3313, 2015.

[19] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, “Fast and accurate modeling of molecular atomization energies with machine learning,” *Physical Review Letters*, vol. 108, p. 058301, Jan 2012.

[20] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, “Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space,” *The Journal of Physical Chemistry Letters*, vol. 6, no. 12, pp. 2326–2331, 2015.

[21] M. Seeger, “Gaussian processes for machine learning,” *International Journal of Neural Systems*, vol. 14, no. 02, pp. 69–106, 2004.

[22] E. A. Walker, D. Mitchell, G. A. Terejanu, and A. Heyden, “Identifying active sites of the water-gas shift reaction over titania supported platinum catalysts under uncertainty,” *ACS Catalysis*, vol. 8, no. 5, pp. 3990–3998, 2018.

[23] E. Walker, S. C. Ammal, G. A. Terejanu, and A. Heyden, “Uncertainty quantification framework applied to the water-gas shift reaction over pt-based catalysts,” *The Journal of Physical Chemistry C*, vol. 120, no. 19, pp. 10 328–10 339, 2016.

[24] J. Lu, S. Behtash, and A. Heyden, “Theoretical investigation of the reaction mechanism of the decarboxylation and decarbonylation of propanoic acid on pd(111) model surfaces,” *The Journal of Physical Chemistry C*, vol. 116, no. 27, pp. 14 328–14 341, 2012.

[25] J. Lu, M. Faheem, S. Behtash, and A. Heyden, “Theoretical investigation of the decarboxylation and decarbonylation mechanism of propanoic acid over a ru(0001) model surface,” *Journal of Catalysis*, vol. 324, pp. 14 – 24, 2015.

[26] B. Meredig and C. Wolverton, “Dissolving the periodic table in cubic zirconia: Data mining to discover chemical trends,” *Chemistry of Materials*, vol. 26, no. 6, pp. 1985–1991, 2014.

[27] L. M. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, “Big data of materials science: Critical role of the descriptor,” *Physical Review Letters*, vol. 114, p. 105503, Mar 2015.

[28] I. T. Jolliffe and J. Cadima, “Principal Component Analysis: a Review and Recent Developments,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, no. 2065, p. 20150202, Apr 2016.

[29] L. Melkumova and S. Shatskikh, “Comparing ridge and lasso estimators for data analysis,” *Procedia Engineering*, vol. 201, pp. 746 – 755, 2017.

[30] C. E. Housecroft and A. G. Sharpe, “Inorganic chemistry,” in *Inorganic Chemistry*. Pearson Prentice Hall, 2008, pp. 31, 1013, 1014.

[31] R. Ramakrishnan and O. A. von Lilienfeld, *Machine Learning, Quantum Chemistry, and Chemical Space*. John Wiley & Sons, Inc., 2017, pp. 225–256.

[32] M. Verleysen and D. François, *The Curse of Dimensionality in Data Mining and Time Series Prediction*. Berlin, Heidelberg: Springer, 2005, pp. 758–770.

[33] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Berlin, Heidelberg: Springer, 2004, pp. 63–71.

[34] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” *Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, 2008.

[35] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI’95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 1137–1143.

[36] M. Rupp, “Machine learning for quantum mechanics in a nutshell,” *International Journal of Quantum Chemistry*, vol. 115, no. 16, pp. 1058–1073, 2015.

[37] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, Y. W. Teh and M. Titterington, Eds., vol. 9. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[38] H. Zou and T. Hastie, “Regularization and variable selection via the elastic net,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[39] C. Bo, F. Maseras, and N. López, "The role of computational results databases in accelerating the discovery of catalysts," *Nature Catalysis*, vol. 1, no. 11, pp. 809–810, 2018.

[40] J. P. Greeley, "Theoretical heterogeneous catalysis: scaling relationships and computational catalyst design." *Annual Review of Chemical and Biomolecular Engineering*, vol. 7, pp. 605–35, 2016.

[41] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1, pp. 151–175, May 2010.

[42] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," *The Journal of Chemical Physics*, vol. 134, no. 7, p. 074106, 2011.

[43] T. Morawietz and J. Behler, "A density-functional theory-based neural network potential for water clusters including van der waals corrections," *The Journal of Physical Chemistry A*, vol. 117, no. 32, pp. 7356–7366, 2013, pMID: 23557541.

[44] J. Behler, "Perspective: machine learning potentials for atomistic simulations," *The Journal of Chemical Physics*, vol. 145, no. 17, p. 170901, 2016.

[45] Z. W. Ulissi, M. T. Tang, J. Xiao, X. Liu, D. A. Torelli, M. Karamad, K. Cummins, C. Hahn, N. S. Lewis, T. F. Jaramillo, K. Chan, and J. K. NÃÿrskov, "Machine-learning methods enable exhaustive searches for active bimetallic facets and reveal active site motifs for co2 reduction," *ACS Catal.*, vol. 7, no. 10, pp. 6600–6608, 2017.

[46] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *Journal of Chemical Information and Modeling*, vol. 50, no. 5, pp. 742–754, 2010, pMID: 20426451.

[47] Y. C. Lo, S. E. Rensi, W. Torng, and R. B. Altman, "Machine learning in chemoinformatics and drug discovery," *Drug Discovery Today*, vol. 23, no. 8, pp. 1538–1546, 08 2018.

[48] B. Sanchez-Lengeling and A. Aspuru-Guzik, "Inverse molecular design using machine learning: Generative models for matter engineering," *Science*, vol. 361, no. 6400, pp. 360–365, 2018.

[49] Z. Wu, B. Ramsundar, E. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, "Moleculenet: a benchmark for molecular machine learning," *Chemical Science*, vol. 9, pp. 513–530, 2018.

[50] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of Computer-Aided Molecular Design*, vol. 30, no. 8, pp. 595–608, Aug 2016.

[51] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 2224–2232.

[52] M. H. S. Segler, T. Kogej, C. Tyrchan, and M. P. Waller, "Generating focused molecule libraries for drug discovery with recurrent neural networks," *ACS Central Science*, vol. 4, no. 1, pp. 120–131, 2018.

[53] W. Torng and R. B. Altman, "3D deep convolutional neural networks for amino acid environment similarity analysis," *BMC Bioinformatics*, vol. 18, no. 1, p. 302, Jun 2017.

[54] C. R. Collins, G. J. Gordon, O. A. von Lilienfeld, and D. J. Yaron, "Constant size descriptors for accurate machine learning models of molecular properties," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241718, Jun 2018.

[55] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, Contour and Grouping in Computer Vision*. London, UK, UK: Springer-Verlag, 1999, pp. 319–345.

[56] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of Physiology*, vol. 148, no. 3, pp. 574–591, Oct 1959.

[57] D. L. Ringach, "Mapping receptive fields in primary visual cortex," *The Journal of Physiology*, vol. 558, no. Pt 3, pp. 717–728, Aug 2004.

[58] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[59] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, “The loss surfaces of multilayer networks,” in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Lebanon and S. V. N. Vishwanathan, Eds., vol. 38. San Diego, California, USA: PMLR, 09–12 May 2015, pp. 192–204.

[60] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147.

[61] B. Hanin and D. Rolnick, “How to start training: the effect of initialization and architecture,” in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 571–581.

[62] A. Vehbi Olgac and B. Karlik, “Performance analysis of various activation functions in generalized mlp architectures of neural networks,” *International Journal of Artificial Intelligence And Expert Systems*, vol. 1, pp. 111–122, 02 2011.

[63] T. G. Tan, J. Teo, and P. Anthony, “A comparative investigation of non-linear activation functions in neural controllers for search-based game ai engineering,” *Artificial Intelligence Review*, vol. 41, no. 1, pp. 1–25, Jan 2014.

[64] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” *Proceedings of ICML*, vol. 30, p. 3, 2013.

[65] C. Andrieu, N. D. Freitas, A. Doucet, and M. I. Jordan, “An introduction to mcmc for machine learning,” *Machine Learning*, vol. 50, pp. 5–43, 2003.

[66] C. P. Robert, V. Elvira, N. Tawn, and C. Wu, “Accelerating mcmc algorithms,” *WIREs Computational Statistics*, vol. 10, no. 5, p. e1435, 2018.

[67] J. M. R. Byrd, S. A. Jarvis, and A. H. Bhalerao, “Reducing the run-time of mcmc programs by multithreading on smp architectures,” in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, 2008, pp. 1–8.

[68] S. Ahn, B. Shahbaba, and M. Welling, “Distributed stochastic gradient mcmc,” in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. JMLR Workshop and Conference Proceedings, 2014, pp. 1044–1052.

[69] A. Korattikara, Y. Chen, and M. Welling, "Austerity in mcmc land: Cutting the metropolis-hastings budget," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. 181–189.

[70] M. Quiroz, R. Kohn, M. Villani, and M.-N. Tran, "Speeding up mcmc by efficient data subsampling," *Journal of the American Statistical Association*, vol. 114, no. 526, pp. 831–843, 2019.

[71] O. Lamminpää, J. Hobbs, J. Brynjarsdóttir, M. Laine, A. Braverman, H. Lindqvist, and J. Tamminen, "Accelerated mcmc for satellite-based measurements of atmospheric co2," *Remote Sensing*, vol. 11, no. 17, 2019.

[72] S. Livingstone, M. F. Faulkner, and G. O. Roberts, "Kinetic energy choice in Hamiltonian/hybrid Monte Carlo," *Biometrika*, vol. 106, no. 2, pp. 303–319, 04 2019.

[73] C. E. Rasmussen, "Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals," in *Bayesian Statistics 7*, J. M. Bernardo, M. J. Bayarri, J. O. Berger, A. P. Dawid, D. Heckerman, A. F. M. Smith, and M. West, Eds. Oxford University Press, 2003, pp. 651–659.

[74] J. A. Christen and C. Fox, "Markov chain monte carlo using an approximation," *Journal of Computational and Graphical Statistics*, vol. 14, no. 4, pp. 795–810, 2005.

[75] A. Chowdhury and G. Terejanu, "An enhanced metropolis-hastings algorithm based on gaussian processes," in *Model Validation and Uncertainty Quantification, Volume 3*, S. Atamturktur, T. Schoenherr, B. Moaveni, and C. Papadimitriou, Eds. Springer International Publishing, 2016, pp. 227–233.

[76] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'13. Arlington, Virginia, USA: AUAI Press, 2013, pp. 282–290.

[77] H. Haario, E. Saksman, and J. Tamminen, "Adaptive proposal distribution for random walk metropolis algorithm," 1999.

[78] H. Haario, M. Laine, A. Mira, and E. Saksman, "Dram: Efficient adaptive mcmc," *Statistics and Computing*, vol. 16, no. 4, pp. 339–354, Dec 2006.

[79] A. Larjo and H. Lähdesmäki, "Using multi-step proposal distribution for improved mcmc convergence in bayesian network structure learning," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2015, no. 1, pp. 1–14, 2015.

[80] F. Meier, P. Hennig, and S. Schaal, "Incremental local gaussian regression," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 972–980.

[81] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Local gaussian process regression for real time online model learning and control," in *Advances in neural information processing systems 21*, Max-Planck-Gesellschaft. Red Hook, NY, USA: Curran, Jun. 2009, pp. 1193–1200.

[82] C. P. Robert and G. Casella, *The Metropolis-Hastings Algorithm.* New York, NY: Springer New York, 1999, pp. 231–283.

[83] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," Universities of Harvard, Oxford, Toronto, and Google DeepMind, Tech. Rep., 2015.

[84] V. Nguyen, "Bayesian optimization for accelerating hyper-parameter tuning," in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, June 2019, pp. 302–305.

[85] C. E. Rasmussen, "Gaussian processes for machine learning," in *Gaussian processes for machine learning.* MIT Press, 2006.

[86] K. P. Murphy, *Machine Learning a Probabilistic Perspective.* The MIT Press, 2012.

[87] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, pp. 130–141, 1963.

[88] G. J. Székely and M. L. Rizzo, "Energy statistics: A class of statistics based on distances," *Journal of Statistical Planning and Inference*, vol. 143, no. 8, pp. 1249 – 1272, 2013.

[89] K. Krauth, E. V. Bonilla, K. Cutajar, and M. Filippone, "Autogp: Exploring the capabilities and limitations of gaussian process models," in *UAI*, 2016.

[90] J. Djolonga, A. Krause, and V. Cevher, "High-dimensional gaussian process bandits," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 1025–1033.

[91] F. Imani, C. Cheng, R. Chen, and H. Yang, "Nested gaussian process modeling and imputation of high-dimensional incomplete data under uncertainty," *IISE Transactions on Healthcare Systems Engineering*, vol. 9, no. 4, pp. 315–326, 2019.

[92] R. Tripathy, I. Bilionis, and M. Gonzalez, "Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation," *Journal of Computational Physics*, vol. 321, pp. 191 – 223, 2016.

Appendix A

Supporting Information for Prediction of Adsorption Energies

A.1 Energy Data

The energy data are prepared to have the same reference values. For example, the adsorption energy for an intermediate surface species $C_xH_yO_z$ was calculated as:

$$E_{C_xH_yO_z} = E^{DFT}_{C_xH_yO_z} - E^{DFT}_{*} - xE_C - yE_H - zE_O$$

where

$$\begin{aligned} E_C &= E^{DFT}_{CH_4(g)} - 2E^{DFT}_{H_2(g)} \\ E_H &= \frac{1}{2}E^{DFT}_{H_2(g)} \\ E_O &= E^{DFT}_{H_2O(g)} - E^{DFT}_{H_2(g)} \end{aligned}$$

Here, E^{DFT}_{*} is the energy of the free site (clean slab) and E^{DFT}_{X} denotes the adsorption energy of the species X from the DFT calculations. Here, all our data were obtained by running DFT calculations on a Pt(111) surface.

For Chapter 2, the energies for each species on each metal surface are shown in table format in Table A.1.

For Chapter 3, our data set is divided into two groups: 247 4C molecules containing 4 carbon atoms along with a variable number of oxygen and hydrogen atoms; and 29 2C and 3C molecules containing either 2 or 3 carbon atoms along with a variable number of oxygen and hydrogen atoms. All the interpolation predictions were done

Table A.1: All the referenced energies (for each metal and each species) in table format with values in eV.

Species	Pd	Pt	Rh	Re	Ru	Cu	Ag	Ni
CH3CH2COOH	2.971	3.097	2.722	2.591	2.540	2.974	3.149	3.031
CH3CHCOOH	3.586	3.488	3.422	3.306	3.046	3.940	4.268	3.596
CH3CCOOH	4.272	4.049	3.684	3.335	3.310	4.781	5.643	4.061
CH2CHCOOH	3.948	3.821	3.813	3.236	3.295	4.550	4.648	3.869
CHCHCOOH	4.745	4.403	4.166	3.526	3.696	5.255	5.955	4.439
CH3CH2CO	2.860	2.862	2.729	2.768	2.590	3.753	4.133	2.984
CH3CHCO	3.607	3.220	3.270	3.143	3.007	4.593	4.686	3.669
CH3CCO	3.962	3.586	3.644	3.077	3.364	5.001	5.462	3.857
CH2CHCO	4.029	3.864	3.618	3.141	3.355	5.302	5.777	4.114
CHCHCO	4.804	4.337	4.260	3.475	3.641	5.935	6.828	4.656
CHCH	2.671	2.346	2.167	1.309	1.809	3.372	4.789	2.120
CH2C	2.536	2.084	2.089	1.943	1.807	3.686	4.696	2.473
CH2CH	2.273	1.827	2.049	1.671	1.642	3.117	3.664	2.296
CH2CH2	1.472	1.245	1.452	1.210	1.309	2.171	2.302	1.615
CH3CH2COO	3.275	3.308	2.735	2.202	2.286	2.924	3.364	2.721
CH3CHCOO	4.423	4.433	3.738	3.263	3.119	4.460	5.140	4.061
CH3CCOO	5.121	4.873	4.516	4.209	4.140	5.705	6.862	4.895
CH3C	1.568	1.221	1.286	1.309	1.234	3.163	4.492	1.641
CH3CH	1.795	1.523	1.519	1.342	1.284	2.727	3.591	1.678
CH3CH2	1.230	0.986	1.251	1.022	1.140	1.853	2.146	1.362
CH3CH3	0.548	0.558	0.562	0.581	0.575	0.574	0.577	0.574
OH	0.612	0.809	0.189	-0.585	-0.320	0.043	0.572	-0.173
H2O	-0.287	-0.274	-0.368	-0.466	-0.484	-0.210	-0.166	-0.311
CO	1.086	1.272	1.118	1.244	1.162	2.188	2.879	1.177
CO2	2.354	2.362	2.447	2.136	2.253	2.362	2.362	2.407
COOH	2.203	2.043	1.822	1.754	1.659	2.636	3.100	2.075
O	1.151	1.293	0.566	-0.492	0.023	0.852	2.175	-0.637
C	2.134	1.842	1.767	1.658	1.446	4.106	5.624	2.329
H	-0.599	-0.507	-0.543	-0.756	-0.630	-0.267	0.148	-0.553

inside the 4C molecules data and the extrapolation were done by training on 4C and testing on 2C and 3C. The following two tables contain the data for 2C/3C and 4C, respectively. In both tables, the first column contains the chemical formula or some unique id for the corresponding species, the second column contains the SMILES notation for the species and the third column contains the referenced adsorption energy of the species.

Table A.2: Energies with the SMILES notation for all the 2C and 3C species.

Species	SMILES	Energy (eV)
CH2CH	[CH2][CH]	1.548
CH2CH2	[CH2][CH2]	0.917
CH2CHCO	[CH2][CH][C](=O)	3.442
CH2CHCOOH	[CH2][CH]C(=O)O	3.233
CH3CCO	C[C][C](=O)	3.145
CH3CCOO	C[C]C(=O)[O]	4.339
CH3CCOOH	C[C]C(=O)O	3.543
CH3CH	C[CH]	1.250
CH3CH2	C[CH2]	0.655
CH3CH2CO	CC[C](=O)	2.488
CH3CH2COO	CCC(=O)[O]	3.004
CH3CH2COOH	CCC(=O)O	2.713
CH3CH3	CC	0.314
CH3CHCO	C[CH][C](=O)	2.719
CH3CHCOO	C[CH]C(=O)[O]	3.883
CH3CHCOOH	C[CH]C(=O)O	2.904
CHCH	[CH][CH]	2.121
CHCHCO	[CH][CH][C](=O)	3.893
CHCHCOOH	[CH][CH]C(=O)O	3.893
CH3CH2CH2O	CCC[O]	2.377
CH3CH2CH2OH	CCCO	1.275
CH3CH2CHO	CCC(=O)	2.532
CH3CH2CHOH	CC[CH]O	1.524
CH3CH2COH	CC[C]O	1.657
CH3CHCH2O	C[CH]C[O]	2.650
CH3CHCH2OH	C[CH]CO	1.676
CH3CHCHO	C[CH]C(=O)	2.707
CH3CHCHOH	C[CH][CH]O	1.808
CH3CHCOH	C[CH][C]O	2.015

Table A.3: Energy (in eV) with the SMILES notation for all the 4C species

Species	SMILES	Energy (eV)
COOHCH2CH2COOH	OC(=O)CCC(=O)O	4.866

Continued on next page

Table A.3 – *Continued from the previous page*

Species	**SMILES**	**Energy (eV)**
COHCH2CH2CHO	C(=O)CC[C](O)	4.472
CH2OHCCH2COOH	OC[C]CC(=O)O	5.154
CH2OHCH2CCOOH	OCC[C]C(=O)O	5.174
COHCHCHCHO	C(=O)[CH][CH][C](O)	5.263
COHCHCHCOH	O[C][CH][CH][C](O)	4.950
CHOCHCHCHO	C(=O)[CH][CH]C(=O)	5.760
CH2OHCHCHCOOH	OC[CH][CH]C(=O)O	4.773
CH2OHCHCH2CO	[C](=O)C[CH]C(O)	4.188
CHOHCHCH2CHO	C(=O)C[CH][CH](O)	4.663
CHOHCHCH2COH	O[CH][CH]C[C](O)	3.985
COHCH2CH2COH	O[C]CC[C](O)	3.876
CHOHCH2CHCOH	O[CH]C[CH][C](O)	4.238
CHOHCH2CHCHO	C(=O)[CH]C[CH](O)	4.526
CH2OHCH2CHCO	[C](=O)[CH]CC(O)	4.149
COHCHCH2COOH	O[C][CH]CC(=O)O	4.766
CHOHCH2CHCOOH	O[CH]C[CH]C(=O)O	4.470
COHCCCO	[C](=O)[C][C][C](O)	7.214
CHOCCCO	C(=O)[C][C][C](=O)	7.079
CHOHCCHCO	[C](=O)[CH][C][CH](O)	5.382
COHCCHCOH	O[C][C][CH][C](O)	5.421
COHCCHCHO	C(=O)[CH][C][C](O)	5.967
CHOCH2CH2CHO	C(=O)CCC(=O)	5.306
CHOCCHCHO	C(=O)[C][CH]C(=O)	6.269

Continued on next page

Table A.3 – *Continued from the previous page*

Species	**SMILES**	**Energy (eV)**
COHCHCCHO	C(=O)[C][CH][C](O)	5.823
CHOHCHCCO	[C](=O)[C][CH][CH](O)	5.409
CHOHCCCOOH	O[CH][C][C]C(=O)O	6.343
CH2OHCCHCOOH	OC[C][CH]C(=O)O	5.204
CH2OHCHCCOOH	OC[CH][C]C(=O)O	5.204
CH2OHCCH2CO	[C](=O)C[C]C(O)	4.746
CHOHCCH2COH	O[CH][C]C[C](O)	4.540
CHOHCCH2CHO	C(=O)C[C][CH](O)	5.446
CHOHCH2CCOH	O[CH]C[C][C](O)	4.487
CH2OHCH2CH2COOH	OCCCC(=O)O	4.010
CHOHCH2CCHO	C(=O)[C]C[CH](O)	5.070
CH2OHCH2CCO	[C](=O)[C]CC(O)	4.721
CH2OHCHCHCO	[C](=O)[CH][CH]C(O)	4.534
CHOHCHCHCHO	C(=O)[CH][CH][CH](O)	4.934
CHOHCHCHCOH	O[CH][CH][CH][C](O)	4.458
CH2OHCHCH2CHO	C(=O)C[CH]C(O)	4.625
CH2OHCHCH2COH	OC[CH]C[C](O)	3.778
CHOHCH2CHCHOH	O[CH][CH]C[CH](O)	3.750
CH2OHCH2CHCOH	OCC[CH][C](O)	3.730
CH2OHCH2CHCHO	C(=O)[CH]CC(O)	4.483
CH2OHCH2CH2CO	[C](=O)CCC(O)	3.790
CH2OHCHCH2COOH	OC[CH]CC(=O)O	4.423
CH2OHCH2CHCOOH	OCC[CH]C(=O)O	4.490

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
CHOHCCCO	[C](=O)[C][C][CH](O)	6.628
COHCHCCO	[C](=O)[C][CH][C](O)	5.670
COHCCCOH	O[C][C][C][C](O)	7.062
COHCCCHO	C(=O)[C][C][C](O)	6.950
CHOCCCHO	C(=O)[C][C]C(=O)	6.934
CH2OHCCHCO	[C](=O)[CH][C]C(O)	5.158
CHOHCCHCOH	O[CH][C][CH][C](O)	5.048
CHOHCCHCHO	C(=O)[CH][C][CH](O)	5.937
CHOHCH2CH2COH	O[CH]CC[C](O)	3.600
CHOHCHCCOH	O[CH][CH][C][C](O)	5.294
CHOHCHCCHO	C(=O)[C][CH][CH](O)	5.544
CH2OHCHCCO	[C](=O)[C][CH]C(O)	5.326
CH2OHCCCOOH	OC[C][C]C(=O)O	5.760
CH2OHCCH2CHO	C(=O)C[C]C(O)	5.297
CH2OHCCH2COH	OC[C]C[C](O)	4.353
CHOHCCH2CHOH	O[CH][C]C[CH](O)	4.277
CH2OHCH2CCOH	OCC[C][C](O)	4.399
CH2OHCH2CCHO	C(=O)[C]CC(O)	4.979
CH2OHCHCHCOH	OC[CH][CH][C](O)	4.289
CHOHCH2CH2CHO	C(=O)CC[CH](O)	4.403
CH2OHCHCHCHO	C(=O)[CH][CH]C(O)	4.731
CHOHCHCHCHOH	O[CH][CH][CH][CH](O)	3.987
CH2OHCHCH2CHOH	OC[CH]C[CH](O)	3.455

Continued on next page

Table A.3 – *Continued from the previous page*

Species	**SMILES**	**Energy (eV)**
CH2OHCH2CHCHOH	OCC[CH][CH](O)	3.650
CH2OHCHCH2CO	[C](=O)C[CH]C(O)	4.062
CH2OHCH2CHCO	[C](=O)[CH]CC(O)	4.242
CH2OHCCCO	[C](=O)[C][C]C(O)	5.867
CHOHCCCOH	O[CH][C][C][C](O)	6.490
CHOHCCCHO	C(=O)[C][C][CH](O)	6.420
CH2OHCCHCOH	OC[C][CH][C](O)	4.947
CH2OHCH2CH2COH	OCCC[C](O)	3.356
CH2OHCCHCHO	C(=O)[CH][C]C(O)	5.259
CHOHCHCCHOH	O[CH][C][CH][CH](O)	4.677
CH2OHCHCCOH	OC[CH][C][C](O)	4.949
CH2OHCHCCHO	C(=O)[C][CH]C(O)	5.134
CH2OHCCH2CHOH	OC[C]C[CH](O)	4.177
CH2OHCH2CCHOH	OCC[C][CH](O)	4.134
CH2OHCHCHCHOH	OC[CH][CH][CH](O)	3.902
CH2OHCH2CHCH2OH	OC[CH]CC(O)	3.455
CH2OHCCCOH	OC[C][C][C](O)	5.737
CH2OHCCCHO	C(=O)[C][C]C(O)	5.738
CH2OHCH2CH2CHO	C(=O)CCC(O)	4.176
CHOHCCCHOH	O[CH][C][C][CH](O)	5.757
CH2OHCCHCHOH	OC[C][CH][CH](O)	4.551
CH2OHCHCCHOH	OC[CH][C][CH](O)	4.784
CH2OHCH2CCH2OH	OC[C]CC(O)	4.204

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
CH2OHCHCHCH2OH	OC[CH][CH]C(O)	3.907
CH2OHCCCHOH	OC[C][C][CH](O)	5.144
CH2OHCHCCH2OH	OC[C][CH]C(O)	4.198
CH2OHCCCH2OH	OC[C][C]C(O)	4.601
CHOHCH2CH2CHOH	O[CH]CC[CH](O)	3.436
COCH2CH2COOH	[C](=O)CCC(=O)O	4.755
CH2OHCH2CH2CHOH	OCCC[CH](O)	3.271
CH2OHCH2CH2CH2OH	OCCCC(O)	2.866
COOHCHCH2COOH	OC(=O)[CH]CC(=O)O	5.405
COOHCCH2COOH	OC(=O)[C]CC(=O)O	6.266
COOHCHCHCOOH	OC(=O)[CH][CH]C(=O)O	5.806
COCHCH2COOH	[C](=O)[CH]CC(=O)O	5.209
COCH2CHCOOH	[C](=O)C[CH]C(=O)O	5.075
COOHCCHCOOH	OC(=O)[C][CH]C(=O)O	6.222
COCCH2COOH	[C](=O)[C]CC(=O)O	5.766
COCH2CCOOH	[C](=O)C[C]C(=O)O	5.670
COCH2CH2CO	[C](=O)CC[C](=O)	4.602
COCHCHCOOH	[C](=O)[CH][CH]C(=O)O	5.555
COCHCH2CO	[C](=O)[CH]C[C](=O)	4.961
COHCHCH2COOH	O[C][CH]CC(=O)O	4.758
CHOCHCH2COOH	C(=O)[CH]CC(=O)O	5.693
COHCH2CHCOOH	O[C]C[CH]C(=O)O	4.755
CHOCH2CHCOOH	C(=O)C[CH]C(=O)O	5.459

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
COCCHCOOH	[C](=O)[C][CH]C(=O)O	6.436
COCHCCOOH	[C](=O)[CH][C]C(=O)O	6.227
COOHCCCOOH	OC(=O)[C][C]C(=O)O	7.073
COCCH2CO	[C](=O)[C]C[C](=O)	5.335
COHCH2CH2COOH	O[C]CCC(=O)O	4.367
COHCCH2COOH	O[C][C]CC(=O)O	5.435
CHOCCH2COOH	C(=O)[C]CC(=O)O	6.024
COHCH2CCOOH	O[C]C[C]C(=O)O	5.379
CHOCH2CCOOH	C(=O)C[C]C(=O)O	6.328
COCHCHCO	[C](=O)[CH][CH][C](=O)	5.479
COHCHCHCOOH	O[C][CH][CH]C(=O)O	5.277
CHOCHCHCOOH	C(=O)[CH][CH]C(=O)O	5.679
COHCHCH2CO	[C](=O)C[CH][C](O)	4.430
COCH2CH2CO	[C](=O)CC[C](=O)	4.585
CHOCHCH2CO	C(=O)[CH]C[C](=O)	5.106
CHOCH2CH2COOH	C(=O)CCC(=O)O	5.120
COCHCH2COH	[C](=O)[CH]C[C](O)	4.433
COCHCH2CHO	C(=O)C[CH][C](=O)	5.417
CHOHCHCH2COOH	O[CH][CH]CC(=O)O	4.524
CHOHCH2CHCOOH	O[CH]C[CH]C(=O)O	4.470
COCCHCO	[C](=O)[C][CH][C](=O)	6.641
COCCCOOH	[C](=O)[C][C]C(=O)O	7.121
COHCCHCOOH	O[C][C][CH]C(=O)O	6.250

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
CHOCCHCOOH	C(=O)[C][CH]C(=O)O	6.223
COHCHCCOOH	O[C][CH][C]C(=O)O	5.906
CHOCHCCOOH	C(=O)[CH][C]C(=O)O	6.288
COCH2CH2COH	[C](=O)CC[C](O)	4.216
COHCCH2CO	[C](=O)C[C][C](O)	5.000
CHOCCH2CO	C(=O)[C]C[C](=O)	5.607
COCCH2COH	[C](=O)[C]C[C](O)	5.081
COCCH2CHO	C(=O)C[C][C](=O)	5.845
COCH2CHCO	[C](=O)[CH]C[C](=O)	4.943
CHOHCCH2COOH	O[CH][C]CC(=O)O	5.240
CHOHCH2CCOOH	O[CH]C[C]C(=O)O	5.197
COHCHCHCO	[C](=O)[CH][CH][C](O)	5.178
CHOCHCHCO	C(=O)[CH][CH][C](=O)	5.655
COCH2CHCO	[C](=O)[CH]C[C](=O)	4.822
COCH2CH2CHO	C(=O)CC[C](=O)	4.964
CHOHCHCHCOOH	O[CH][CH][CH]C(=O)O	4.876
CHOHCHCH2CO	[C](=O)C[CH][CH](O)	4.265
COHCH2CH2CO	[C](=O)CC[C](O)	4.218
COHCHCH2COH	O[C][CH]C[C](O)	4.086
COHCH2CHCOH	O[C][CH]C[C](O)	4.967
CHOCH2CH2CO	C(=O)CC[C](=O)	4.978
CHOCHCH2COH	C(=O)[CH]C[C](O)	4.777
CHOCHCH2CHO	C(=O)[CH]CC(=O)	5.685

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
CHOHCH2CHCO	[C](=O)[CH]C[CH](O)	4.392
CH2OHCHCH2COOH	OC[CH]CC(=O)O	4.470
CHOHCH2CH2COOH	O[CH]CCC(=O)O	4.192
CHOHCH2CHCOOH	O[CH]C[CH]C(=O)O	4.179
CHOHCCH2COOH	O[CH][C]CC(=O)O	5.298
CHOHCH2CCOOH	O[CH]C[C]C(=O)O	5.196
CH2OHCH2CHCOOH	OCC[CH]C(=O)O	4.397
COCCCO	[C](=O)[C][C][C](=O)	7.482
COHCCHCO	[C](=O)[CH][C][C](O)	5.914
CHOCCHCO	C(=O)[C][CH][C](=O)	6.193
CHOCHCCO	C(=O)[CH][C][C](=O)	6.447
COHCHCCO	[C](=O)[C][CH][C](O)	5.659
COHCCCOOH	O[C][C][C]C(=O)O	6.988
CHOHCH2CH2CO	[C](=O)CC[CH](O)	3.978
CHOCCCOOH	C(=O)[C][C]C(=O)O	6.939
CHOHCCHCOOH	O[CH][C][CH]C(=O)O	5.864
CHOHCHCCOOH	O[CH][CH][C]C(=O)O	5.579
CHOHCCH2CO	[C](=O)C[C][CH](O)	5.586
COHCH2CCOH	O[C][C]C[C](O)	4.783
COHCCH2CHO	C(=O)C[C][C](O)	5.550
COHCH2CCHO	C(=O)[C]C[C](O)	5.306
CHOHCH2CCO	[C](=O)[C]C[CH](O)	4.782
COHCH2CCHO	C(=O)[C]C[C](O)	5.304

Continued on next page

Table A.3 – *Continued from the previous page*

Species	**SMILES**	**Energy (eV)**
CHOCH2CCHO	C(=O)[C]CC(=O)	6.463
C(OH)2CH2CH2COOH	O[C](O)CCC(=O)O	4.645
C(OH)2CH2CH2CO	O[C](O)CC[C](=O)	4.403
C(OH)2CH2CH2C(OH)2	O[C](O)CC[C](O)O	4.310
C(OH)2CH2CH2COH	O[C](O)CC[C]O	4.018
C(OH)2CH2CH2CHO	O[C](O)CCC(=O)	3.971
C(OH)2CH2CH2CHOH	O[C](O)CC[CH]O	3.875
C(OH)2CH2CH2CH2OH	O[C](O)CCCO	3.685
DCX_IM1	[O]C(=O)CCC(=O)	5.785
DCX_IM10	[O]C(=O)C[CH][CH](O)	5.743
DCX_IM11	[O]C(=O)C[CH][C](=O)	6.486
DCX_IM12	[O]C(=O)C[CH][C](O)	5.818
DCX_IM13	[O]C(=O)C[C]C(=O)	6.999
DCX_IM14	[O]C(=O)C[C]C(=O)O	6.682
DCX_IM15	[O]C(=O)C[C]C(O)	6.084
DCX_IM16	[O]C(=O)C[C][CH](O)	6.314
DCX_IM17	[O]C(=O)C[C][C](=O)	6.861
DCX_IM18	[O]C(=O)C[C][C](O)	6.411
DCX_IM19	[O]C(=O)[CH]CC(=O)	6.261
DCX_IM2	[O]C(=O)CCC(=O)O	5.661
DCX_IM20	[O]C(=O)[CH]CC(=O)O	6.235
DCX_IM21	[O]C(=O)[CH]CC(O)	5.288
DCX_IM22	[O]C(=O)[CH]C[CH](O)	5.450

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
DCX_IM23	[O]C(=O)[CH]C[C](=O)	6.109
DCX_IM24	[O]C(=O)[CH]C[C](O)	5.681
DCX_IM25	[O]C(=O)[CH][CH]C(=O)	6.693
DCX_IM26	[O]C(=O)[CH][CH]C(=O)O	6.580
DCX_IM27	[O]C(=O)[CH][CH]C(O)	5.806
DCX_IM28	[O]C(=O)[CH][CH][CH](O)	5.762
DCX_IM29	[O]C(=O)[CH][CH][C](=O)	6.572
DCX_IM3	[O]C(=O)CCC(O)	4.469
DCX_IM30	[O]C(=O)[CH][CH][C](O)	6.200
DCX_IM31	[O]C(=O)[CH][C]C(=O)	7.044
DCX_IM32	[O]C(=O)[CH][C]C(=O)O	7.359
DCX_IM33	[O]C(=O)[CH][C]C(O)	5.981
DCX_IM34	[O]C(=O)[CH][C][CH](O)	6.740
DCX_IM35	[O]C(=O)[CH][C][C](=O)	7.428
DCX_IM36	[O]C(=O)[CH][C][C](O)	6.922
DCX_IM37	[O]C(=O)[C]CC(=O)	6.946
DCX_IM38	[O]C(=O)[C]CC(=O)O	6.900
DCX_IM39	[O]C(=O)[C]CC(O)	5.833
DCX_IM4	[O]C(=O)CC[CH](O)	5.377
DCX_IM40	[O]C(=O)[C]C[CH](O)	5.978
DCX_IM41	[O]C(=O)[C]C[C](=O)	6.495
DCX_IM42	[O]C(=O)[C]C[C](O)	6.046
DCX_IM43	[O]C(=O)[C][CH]C(=O)	7.251

Continued on next page

Table A.3 – *Continued from the previous page*

Species	SMILES	Energy (eV)
DCX_IM44	[O]C(=O)[C][CH]C(=O)O	6.987
DCX_IM45	[O]C(=O)[C][CH]C(O)	6.146
DCX_IM46	[O]C(=O)[C][CH][CH](O)	6.418
DCX_IM47	[O]C(=O)[C][CH][C](=O)	7.219
DCX_IM48	[O]C(=O)[C][CH][C](O)	6.780
DCX_IM49	[O]C(=O)[C][C]C(=O)	7.618
DCX_IM5	[O]C(=O)CC[C](=O)	5.898
DCX_IM50	[O]C(=O)[C][C]C(=O)O	7.730
DCX_IM51	[O]C(=O)[C][C]C(O)	6.452
DCX_IM52	[O]C(=O)[C][C][CH](O)	7.459
DCX_IM53	[O]C(=O)[C][C][C](=O)	8.571
DCX_IM54	[O]C(=O)[C][C][C](O)	7.737
DCX_IM6	[O]C(=O)CC[C](O)	5.448
DCX_IM7	[O]C(=O)C[CH]C(=O)	6.486
DCX_IM8	[O]C(=O)C[CH]C(=O)O	6.463
DCX_IM9	[O]C(=O)C[CH]C(O)	5.446

A.2 Descriptor or Feature data and their Calculations

A.2.1 Coulomb Matrix and Bag-of-Bonds

The diagonal entries in the Coulomb matrix are given by

$$C(i,i) = 0.5 * Z_i^{2.4}$$

where Z represents the atomic number. The off-diagonal (i,j)-th entry of the Coulomb matrix is given by

$$C(i,j) = \frac{Z_i * Z_j}{r}$$

where r represents the distance between the atoms in Angstrom. The sorted eigenvalues of the matrix are then used as the descriptor. The bag-of-bond method works with only the off-diagonal elements of the Coulomb matrix by placing the entries for each pair of atoms inside a bag and thus building a long vector.

There are 18 sorted eigenvalues for the Coulomb matrix obtained for each species. The Coulomb matrix is 18-by-18 since in our dataset (4C or 2C/3C) the maximum numbers of carbon, oxygen and hydrogen atoms in a species are 4, 4 and 10, respectively; which makes maximum possible molecular size for our database 18.

For bag-of-bonds, there are 108 long vectors for each species. Since, BoB deals with the lower (or upper) triangle of the Coulomb matrix (excluding the diagonal entries), for 18-by-18 matrix, there are $\frac{18*(18-1)}{2}$ or 153 entries. But, we do not consider the entries corresponding to hydrogen-hydrogen which accounts for $\frac{10*(10-1)}{2}$ or 45 entries. Subtracting 45 from 153 gives 108.

A.2.2 Extended Connectivity Fingerprints

To generate fingerprints based on extended connectivity, we have used the open source cheminformatics software 'RDKit'. From there, the API for 'Morgan fingerprint' was used with default radius value 2 (see `https://www.rdkit.org/docs/Cookbook.html`) which is roughly equivalent to ECFP4 (see `https://www.rdkit.org/docs/GettingStartedInPython.html`). We tried with different number of bits and found the results to get better as the number of bits was increased. However, after 1000 bits, there was no significant improvement on our dataset, and hence, we have used 1000 bits.

A.2.3 Flat Molecular Fingerprints from SMILES

This is our hand-coded fingerprints based SMILES. The details of the fingerprint is discussed in Figure 3.2.

A.2.4 Atomic Fingerprints from Coordinates

For each atom, we used a 5-length vector as its atomic fingerprint based on the atomic coordinates. First four of these are based on pairwise distance measures (for each atom, the pairwise distance measures to all the carbons, hydrogens, oxygens, and metal atoms make these four values) and the last value comes from the triplet distance measures from the current atom to all other atom pairs. Since, coordinate data obtained from DFT calculations contain coordinates of the metal catalyst surface, we have included the top two layers of metal atom in our calculation of these fingerprints.

The pairwise measure for the i-th atom, P_i is obtained using the following equation:

$$P_i = \sum_j e^{-R_{ij}^2}.f_c(R_{ij})$$

where the summation, in our case, is over all atoms of an atom type; R_{ij} is the distance between atom i and j; and $f_c(R_{ij})$ is given by:

$$f_c(R_{ij}) = \begin{cases} 0.5[\cos\frac{\pi R_{ij}}{R_c} + 1] & for R_{ij} < R_c \\ 0 & for R_{ij} > R_c \end{cases}$$

where R_c is the cut-off radius. We used 4 angstrom for this value.

The triplet distance measure for the i-th atom, T_i is obtained using the following equation:

$$P_i = \sum_{j,k\neq i}^{all} (1 + \cos\theta_{ijk}).e^{-(R_{ij}^2+R_{ik}^2+R_{jk}^2)}.f_c(R_{ij}).f_c(R_{ik}).f_c(R_{jk})$$

where $\cos\theta_{ijk} = \frac{\vec{R_{ij}}.\vec{R_{ik}}}{R_{ij}R_{ik}}$

For each species, there are 18 (since this is the maximum number of possible atoms in a species in our database) 5-long vectors for each of the possible atoms of that species - starting with 4 carbon atoms, then 10 hydrogen and 4 oxygen. The actual ordering of the atoms inside an atom type does not matter as each set of atomic fingerprints will be fed to separate neural networks where these nets will share the weights. If a species does not have all the C, H or O atoms, those spaces are given values of zero. The neural net for that (absent) atom will output zero contribution towards the total energy.

A.2.5 Atomic Fingerprints from SMILES

For each atom, we used an 8-length vector (according to Figure 3 in the main paper) as its atomic fingerprint based on SMILES. Since accounting for all the bonds centered on carbon and oxygen will include all the bonds from hydrogen, we have only used atomic fingerprints centered on carbon and oxygen atoms. Since no coordinate data is used here, no metal surface information is included in these fingerprints. As any species in our database contains at most four carbon and four oxygen atoms, for each atom we have $4 + 4 = 8$ sets of atomic fingerprints.

For each species, there are 8 8-long vector for each of the possible atoms of that species - starting with 4 carbon atoms, then 4 oxygen. The sequence of values inside each atomic fingerprint follows Figure 3.4. The ordering of atoms, and zero values for non-existent atoms (for smaller-than-max molecules) are handled the same way as it was for the coordinate based fingerprints.

A.3 Calculation Procedure for Interpolative Predictions

Calculation process for prediction across metals (results for which are shown in Table 2.2) is presented in Algorithm 4.

Algorithm 4 Calculation of Prediction Across Metals Using Only Metal Descriptors

```
1: for each descriptor combination do
2:     for each metal do
3:         For each species of the metal get the predicted energy using 7 other energies
           for that species for all other metals.
4:         Absolute error for each prediction stored.
5:         Mean absolute error for that metal and for that descriptor combination is
           saved.
6:     end for
7:     Mean and standard deviation of all the absolute errors is calculated for the
       descriptor combination.
8: end for
```

Calculation process for prediction across species and metals (results for which are shown in Table 2.3) is presented in Algorithm 5.

Algorithm 5 Calculation of Prediction Across Species and Metals

```
1: for each set of list (all metal-species pair for a combination) entries in the 'Inputs'
   section do
2:     for each ML algorithm in GP,KRR,SVR,Ridge,Lasso,Elastic do
3:         for 100 times do
4:             Read the numerical entries.
5:             Shuffle data randomly and then split into train and test sets.
6:             Use 5 fold cross-validation on training data to obtain best hyperparam-
               eters for the current algorithm.
7:             Generate predictions for test set using optimum hyperparameters.
8:             Absolute errors for this run are saved temporarily.
9:         end for
10:        Mean and SD of all the absolute errors for the algorithm is calculated and
           stored.
11:    end for
12:    Save results
13: end for
```

A.4 Machine Learning Models and Hyper-parameter Settings

The hyper-parameters for machine learning (ML) models such as ridge regression, LASSO, support vector regression (SVR), kernel ridge regression (KRR), Gaussian process (GP) were tuned using 5-fold cross validation. All of the above models except

GP was run using the Python-based library 'scikit-learn'. GP was run using the library 'GPy'.

All neural network based models were created using the 'TensorFlow' API. Our proposed multi filter based model was run with the following setting: hyperbolic tangent activation function (this worked better than the ReLU activation function), dropout value of 0.9 (10% of the hidden units were randomly dropped), atomic subnetwork weights and filter contribution weights ($W^{(i)}$s in Figure 4 of the main paper) had regularizer with scale 0.001, atomic subnetwork weights randomly initialized with zero mean and 0.0001 standard deviation, learning rate of 0.001, Adam optimizer, session variables saved with tolerance value of 0.001 for improved validation cost, atom-type contribution weights (W_C and W_O in Figure 4 of main text, shared across the filters) were initialized with constant values of 1 (as their default behavior is to just add up linearly), subnetwork structure of 5-by-8-by-1 for coordinate based atomic fingerprints and 8-by-10-by-1 for SMILES based atomic fingerprints. To get an ensemble of the results, the runs were performed 10 times to obtain the mean predicted value for each species in the test set.

A.5 Results

For all of the following tables for interpolation, the results were obtained by dividing the 4C data randomly into training (size 215) and testing (size 32) set. After running an ML model, the mean of the absolute errors on the testing set gives one MAE. Then the whole data is again randomly permuted and divided into training and testing set, which gives another MAE. The process is repeated 100 times. The mean of all these MAEs (which is also the mean of all the absolute errors across all runs) gives the values for the column with header 'Mean of MAEs'. The standard deviation of the MAEs are given in the column with the header 'SD of the MAEs'. The standard deviation of all the absolute errors across the 100 runs is reported in the column 'SD of AEs'. In case of Gaussian process (GP), there are two additional columns. As GP provides an uncertainty measure around each prediction point, we report the mean and standard deviation of those 'std's in the last two columns of the tables where GP is used.

Table A.4: Interpolation prediction errors for Coulomb matrix (CM) and bag-of-bonds (BoB)

Method	**Model**	**Mean of MAEs (eV)**	**SD of MAEs (eV)**	**SD of AEs (eV)**	**Mean of Stds (eV)**	**SD of Stds (eV)**
CM	**svr**	0.236	0.041	0.241	N/A	N/A
CM	**krr**	0.233	0.050	0.254	N/A	N/A
CM	**ridge**	0.327	0.040	0.272	N/A	N/A
CM	**lasso**	0.319	0.045	0.260	N/A	N/A
CM	**elastic**	0.327	0.047	0.277	N/A	N/A
CM	**gp**	0.230	0.036	0.218	0.305	0.181
BoB	**svr**	0.210	0.038	0.214	N/A	N/A
BoB	**krr**	0.139	0.024	0.136	N/A	N/A
BoB	**ridge**	0.219	0.047	0.279	N/A	N/A
BoB	**lasso**	0.218	0.043	0.257	N/A	N/A
BoB	**elastic**	0.218	0.043	0.271	N/A	N/A
BoB	**gp**	0.222	0.051	0.306	0.099	0.124

Table A.5: Interpolation prediction errors for ECFP (Extended Connectivity Fingerprints) and Flat Fingerprints

Fingerprint type	Model	Mean of MAEs (eV)	SD of MAEs (eV)	SD of AEs (eV)	Mean of Stds (eV)	SD of Stds (eV)
ECFP	**svr**	0.165	0.030	0.179	N/A	N/A
ECFP	**krr**	0.180	0.029	0.188	N/A	N/A
ECFP	**ridge**	0.183	0.031	0.184	N/A	N/A
ECFP	**lasso**	0.186	0.031	0.187	N/A	N/A
ECFP	**elastic**	0.177	0.029	0.180	N/A	N/A
ECFP	**gp**	0.183	0.037	0.185	0.014	0.002
FlatFP	**svr**	0.148	0.021	0.129	N/A	N/A
FlatFP	**krr**	0.141	0.022	0.122	N/A	N/A
FlatFP	**ridge**	0.196	0.027	0.166	N/A	N/A
FlatFP	**lasso**	0.189	0.024	0.156	N/A	N/A
FlatFP	**elastic**	0.189	0.026	0.154	N/A	N/A
FlatFP	**gp**	0.150	0.023	0.127	0.080	0.104

Table A.6: Interpolation prediction errors for additive subnetwork with atomic fingerprints from both coordinates and SMILES

Fingerprint type	Filter Count	Mean of MAEs (eV)	SD of MAEs (eV)	SD of AEs (eV)
Coordinate	**1**	0.347	0.027	0.259
Coordinate	**2**	0.335	0.022	0.244
Coordinate	**4**	0.309	0.024	0.231
Coordinate	**6**	0.301	0.022	0.221
Coordinate	**8**	0.299	0.024	0.215
SMILES	**1**	0.190	0.025	0.164
SMILES	**2**	0.154	0.020	0.118
SMILES	**4**	0.154	0.024	0.124
SMILES	**6**	0.142	0.025	0.120
SMILES	**8**	0.142	0.017	0.133

Table A.7: Extrapolation prediction errors for Coulomb matrix (CM) and bag-of-bonds (BoB)

Method	Model	MAEs (eV)	SD of AEs (eV)	Mean of Stds (eV)	SD of Stds (eV)
CM	**svr**	2.392	1.015	N/A	N/A
CM	**krr**	2.401	1.004	N/A	N/A
CM	**ridge**	12.105	3.273	N/A	N/A
CM	**lasso**	12.105	3.273	N/A	N/A
CM	**elastic**	11.579	3.093	N/A	N/A
CM	**gp**	2.588	1.261	0.995	0.009
BoB	**svr**	2.596	0.612	N/A	N/A
BoB	**krr**	2.046	0.422	N/A	N/A
BoB	**ridge**	4.292	0.858	N/A	N/A
BoB	**lasso**	4.292	0.858	N/A	N/A
BoB	**elastic**	4.280	0.880	N/A	N/A
BoB	**gp**	2.785	0.643	0.100	0.054

Table A.8: Extrapolation prediction errors for ECFP (Extended Connectivity Fingerprints) and Flat Fingerprints

Fingerprint type	Model	MAEs (eV)	SD of AEs (eV)	Mean of Stds (eV)	SD of Stds (eV)
ECFP	**svr**	2.961	0.760	N/A	N/A
ECFP	**krr**	2.872	0.760	N/A	N/A
ECFP	**ridge**	2.965	0.784	N/A	N/A
ECFP	**lasso**	2.965	0.784	N/A	N/A
ECFP	**elastic**	2.989	0.758	N/A	N/A
ECFP	**gp**	2.985	0.764	0.015	0.001
FlatFP	**svr**	2.426	0.660	N/A	N/A
FlatFP	**krr**	2.342	0.625	N/A	N/A
FlatFP	**ridge**	2.431	0.619	N/A	N/A
FlatFP	**lasso**	2.431	0.619	N/A	N/A
FlatFP	**elastic**	2.322	0.588	N/A	N/A
FlatFP	**gp**	2.373	0.646	0.102	0.034

Table A.9: Extrapolation prediction errors for additive subnetwork with atomic fingerprints from both coordinates and SMILES

Fingerprint type	**Filter Count**	**Mean of MAEs (eV)**	**SD of MAEs (eV)**	**SD of AEs (eV)**
Coordinate	**1**	0.324	0.017	0.212
Coordinate	**2**	0.283	0.014	0.196
Coordinate	**3**	0.282	0.014	0.192
Coordinate	**4**	0.282	0.013	0.190
Coordinate	**5**	0.289	0.015	0.201
Coordinate	**6**	0.284	0.014	0.198
Coordinate	**7**	0.290	0.015	0.203
Coordinate	**8**	0.289	0.015	0.203
SMILES	**1**	0.434	0.097	0.314
SMILES	**2**	0.313	0.055	0.248
SMILES	**3**	0.275	0.051	0.214
SMILES	**4**	0.261	0.045	0.185
SMILES	**5**	0.227	0.015	0.143
SMILES	**6**	0.233	0.017	0.149
SMILES	**7**	0.235	0.023	0.163
SMILES	**8**	0.236	0.022	0.159
SMILES	**9**	0.236	0.023	0.164
SMILES	**10**	0.228	0.017	0.156