Learning to Optimize the Positioning of Methane on Metal Catalysts Using Gaussian Processes

Darius Russell Kish

Advisor: Jean-Baptiste Tristan, Ph.D.
Department of Computer Science

With special thanks to Daniel Huang, Ph.D.

Honors Thesis Submitted In Partial Fulfillment of
Bachelor of Arts in Computer Science with Honors
in the College of Arts and Science

Boston College
Chestnut Hill, Massachusetts
April 2021

Abstract

Dry Reforming of Methane is a vital industrial reaction to produce hydrogen from methane, which is used in many downstream reactions. The current method for the reforming of methane uses high pressure steam, leading to an undesirable CO_2 producing side reaction that is a major greenhouse gas contributor. Much research has been dedicated to dry reforming on metal catalysts, but current generation catalysts are prone to deactivation by side products. Physical R&D of catalysts takes approx. 2 years, so computational methods are vital for high-throughput exploration of new catalysts. The standard method uses Quantum Mechanical (QM) theory, taking into account subatomic interactions, however QM comes with a large computational cost. Significant time is spent optimizing orientation of the molecule on the catalyst surface, and improvements to the optimization cycle will greatly improve evaluation throughput.

Using a small dataset of precomputed, optimized systems, we propose and implement a method of transfer learning by using the posterior mean of a Gaussian Process (GP) derived from training data as the prior mean for a Gaussian Process that optimizes the geometry of an unseen system. We implement this method in GPyTorch, which also allows us to fit the gradient of the energy, more canonically recognized as the forces on the atoms. We may then find a geometry of minimum energy by iteratively minimizing the posterior mean of the testing GP, evaluating new steps using QM, and adding the data to the GP for the next iteration. Our main contribution is the framework and code to perform this task, though we show a brief proof of concept for its function.

1 Introduction

1.1 Motivation

Catalysis is a technique in chemistry where an additive substance is used to help lower the activation energy of a reaction, thereby increasing the net gain of energy in a reaction. At industrial scale, catalysts make many important reactions possible that otherwise would require too much energy input to be feasible. Due to their integral role in important industrial reactions, they are a well studied area that receives both academic and industry research funding. Importantly, a catalyst is not consumed like the reactants and instead can be reused, lending to their often complex and expensive formulation.

The class of catalysts generally of greatest interest to industry are called heterogeneous catalysts, since the phase of the catalyst (often a solid metal surface) is different than the reactants, which are liquid or gas phase.[1] Heterogeneous catalysis offers the benefit of easy separation of the catalyst from the product, and metal catalysts can generally be produced in rigid, high surface area forms that facilitate increased reaction capacities. A classic textbook example is hydrogenation reactions on a platinum stabilized on carbon catalyst.

1.1.1 Dry Reforming of Methane

Traditionally, hydrogen and carbon monoxide, known industrially as syngas, is produced through a process called steam reforming, where methane is reacted with high pressure steam.[2] Syngas is used downstream in synthetic natural gas, ammonia and methanol production. A side reaction, called the water-gas shift reaction, occurs under these steam conditions, however, where the steam reactant and carbon monoxide product preferably rearrange to form carbon dioxide and additional hydrogen.

$$\mathrm{H_2O + CH_4 \rightleftharpoons 3\,H_2 + CO}$$
$$\mathrm{CO + H_2O \rightleftharpoons CO_2 + H_2}$$

Scheme 1: steam reforming of methane and the water-gas shift reaction

The water-gas shift reaction causes steam reforming to be a greenhouse gas producing process.[3,4] As a result, a specific interest for heterogeneous catalysis is in dry reforming

of methane (DRM).[5,6] Nickel is a commonly proposed catalyst on which the reaction is performed.

$$CO_2 + CH_4 \longrightarrow 2\,H_2 + 2\,CO$$

Scheme 2: dry reforming of methane

Dry reforming, however, is susceptible to the buildup of carbon, known as coke in industry, on the surface of the metal catalyst, eventually deactivating the catalyst. Current research in DRM has been focused on development of coke buildup-resistant catalysts. One major hurdle is the timeline and cost for development and testing of new physical catalysts, which takes approximately 2 years per catalyst and has exorbitant costs for custom production of small batch catalysts and industrial testing environments. Inhibitory physical development times have incentivized computational chemists to approach the problem. The role of computational chemistry is to predict which catalysts will be performant by using physical-chemistry theories of the reaction rates.[1] The role of computational models for DRM is to predict the reaction rate in the presence of the catalyst. Whereas this rate is measured experimentally by monitoring the concentrations of reactants and products over time, computationally we can model the reaction using macro state thermodynamic properties.

$$k_{rate} \propto e^{-\Delta G_{TS-GS}} \tag{1}$$

The reaction rate is proportional to the exponential of the difference in Gibb's free energy of the transition state* and the ground state minimum geometry.[7] It is important that both the transition and ground state minimum geometries are accurate, as all errors end up passing through this exponential term. The Gibb's free energy is a macro state property whose calculation will be discussed in the next section. We must additionally find both the ground state minimum and transition state geometries, i.e., minimum energy position of the methane molecule relative to the surface catalyst. The transition state is found by calculating the ground state geometry of the reactant and the product for the specific *reaction step*, and then extrapolating the saddle point transition state geometry using known algorithms. In the case of DRM, the methane first dissociates the surface of the catalyst in scheme 3, and is the mechanistic step of interest in this study. Finding these geometries requires the use of quantum mechanics, because modeling accurate interatomic interactions requires modeling the electrons. Quantum mechanical calculations are extremely costly, and will also be discussed in the next section.

*The transition state is defined as the geometry of the energetic saddle point between two other states

$$CH_4 \longrightarrow CH_3 + H$$

Scheme 3: Dissociation of methane on catalyst into CH_3 and H

1.2 Computational Chemistry

Computational approaches to chemistry offer a multi-level, physics-based model of chemical systems, albeit with a number of approximations and simplifications to make the models tractable. The overarching goals of most computational chemistry are twofold: to extrapolate macro state properties of the system, and gain otherwise unobservable perspectives on sub-microscopic properties. Macro state properties are often heavily reliant on micro state properties, though not *vice versa*. It is important to note that computational models of chemistry are by no means a *solved* topic in chemistry, i.e., a simplified tool to aid in other research, and that for many applications, significant research must be invested into computational methods in order to best model the systems under study. To clarify, however, much of the theory behind quantum chemistry is solved, what is not is the implementation. Both the Schrödinger and Dirac equations perfectly model chemical systems, however their pure forms result in intractable systems. Much of the work in modern day QC is the ability to quickly find good approximate solutions, which is often system dependent and requires expert knowledge.

There are two main perspectives that computational chemistry can take on chemical systems: atomistic and quantum mechanical (QM). Atomistic models take on a simplified view of the physical world by forgoing modeling subatomic particles and instead modeling solely atoms and bonds. Bonds are a tricky topic in chemistry, as they often exist for convenience of understanding and communication, such is the case in much of organic chemistry, however their true representation is observed as shared electron density between atoms. As such, it is difficult to model systems with non-trivial bonds, such as interactions between organic moieties and metals, aromatic and conjugated systems, and systems involving many ill-defined bonds. Similarly, the concept of covalent reactions[†] are difficult to model as a bond is considered a base attribute of the system, leading to the comparison of two nonequivalent systems.[8] Quantum mechanical models, on the other hand, model the subatomic particles as well. In chemistry this is often limited to protons, neutrons and electrons. Such a perspective clearly allows for the inclusion of electronic properties in the model. Similarly, covalent changes do not affect the net number of electrons in the systems, only changing their distribution in space, allowing for easier comparison of covalent changes. The disadvantage of QM models is

[†]reactions involving making/breaking bonds

the large increase in computational complexity, which will be introduced later. Approximate methods, such as density functional theory, are already cubic with the number of electrons in the system.

In the case of DRM, computational chemistry offers the a potential advantage over physical R&D by eliminating the physical production and testing of proposed catalysts. Instead, computational models may be built to simulate these catalysts, hopefully capturing the desired properties to an acceptable degree of accuracy in significantly less time. The properties of interest for DRM, the Gibb's free energy, and ground and transition state geometries[‡], necessitate quantum mechanical methods to fully realize the impact of electronic contributions to them.

1.2.1 Quantum Chemistry Calculations

This section serves to very briefly introduce the reader to quantum mechanical calculations, specifically the Gibb's free energy and the potential energy of a geometry. For a comprehensive introduction to QM, see Jensen's textbooks on Computational Chemistry.[7]

The overarching goal of all QM calculations is to find an approximate solution to the time independent Schrödinger equation, equation 2.[9] This of course does not have a solution when many electrons are involved, and it is often approximated using Density Functional Theory (DFT).[10] While the specifics will not be discussed here, DFT decomposes the a system of n electrons into n one-electron Schrödinger equations, known as Kohn-Sham equations, which can then be tractably solved. Importantly, DFT requires approximations on the true many-electron from the Schrödinger equation, though it is often sufficiently accurate and widely used in materials and chemistry research. An important, and well-known approximation is the Born-Oppenheimer approximation, which decouples electrons from the nuclei due to the mass difference between the two species.

$$\hat{H}\,|\Psi\rangle = E\,|\Psi\rangle \tag{2}$$

For sufficiently massive atoms, a relativistic version of the Schrödinger equation, known as the Dirac equation, must instead be solved. The Dirac equation results in four different coupled differential equations for $\gamma^\mu = (\gamma^0, \gamma^1, \gamma^2, \gamma^3)$ that describe the wavefuntion ψ containing four components. To simplify things greatly, both the Schrödinger and Dirac equation are complex and difficult to solve for real chemical systems.

[‡]geometry in the context of chemistry refers to the collection of Cartesian coordinates of all atoms in the system. In other words, a point cloud.

$$(i\gamma^{\mu}\delta_{\mu} - m)\psi = 0 \tag{3}$$

Once an approximate solution to the Schrödinger equation is found via numerical methods, a number of micro state properties of the system may be calculated. Most relevant to our study is the potential energy and the sum of all forces acting on each atom nucleus. For brevity, we can abstract all math and physics related to QM into a DFT function, as we did not contribute to DFT in this study and instead used available software packages that behave like the following function.[11] The DFT function takes an ordered collection of atoms, A, which correspond to the periodic elements (assuming ground state electronic structure), and a corresponding ordered collection of Cartesian coordinates, P. The function returns the energy of the system, and additionally, most DFT software packages make the forces and Hessian readily available. Studying a single system, it is clear that A remains constant and only the positions change, so we can curry A into DFT to make it a function solely of the positions for a given system.

$$DFT_A : \mathbb{R}^{3N} \to \mathbb{R} \tag{4}$$

$$DFT_A(P) = E \tag{5}$$

$$F = \boldsymbol{\nabla} DFT_A(P) \tag{6}$$

The potential energy (hereafter referred to as energy) of the a single, constant system is then a function solely of its geometry. This gives rise to the concept of a potential energy surface (PES), which is the high-dimensional surface of the energy with respect to the geometry.

Geometry optimization, or minimization of the energy, clearly arises from the DFT function, as the availability of first and second order derivatives of the energy with respect to the geometry are readily available. This optimization is commonly performed using L-BFGS, however many other methods are available.

We utilize a DFT package called VASP (Vienna Ab Initio Simulation Package)[11], which after some configuration to obtain an equivalent of A, behaves identically to the DFT function. Quantum Mechanical calculations are, however, extremely computationally expensive, especially when the Dirac equation is necessitated. We utilized the BC Computational Cluster in our study, using 96 cores from Infiniband-networked Intel Xeon Platinum 8260 CPUs (2.40GHz). The primary bottleneck of VASP is CPU time spent solving the Schrödinger

equation. A single point calculation[§] requires approximates two hours on our system of Ni or Co catalyst doped with a single transition metal atom, and the methane molecule. A geometry optimization is dependent on the number of steps needed for convergence, but often takes upwards of one week of walltime.

Geometry optimization is needed to find the ground state geometries of CH_4 and the dissociated CH_3 and H on the catalyst, in order to then predict their Gibb's free energies. A separate form of geometry optimization, finding the geometry of the transition state, is often even more expensive than energy minimization and is also required. It is clear that geometry optimization is a major bottleneck in the computational evaluation of new catalysts. Using VASP, a minimum energy optimization can take upwards of 1 week using 96 cores in the above configuration, and a transition state optimization takes more than a month of walltime. A long-term goal of this study is to improve optimization methods to provide quicker convergence, as will be discussed in later sections.

1.2.2 Calculation of Macro State Thermodynamic Properties

Until now, we've considered only quantum mechanic calculations on a single snapshot of a system in time and space. In order to obtain macro state properties, we must include aspects from statistical mechanics. Much of statistical mechanics is grounded on the Boltzmann distribution below, which describes the probability of a molecule being in a state with energy ε at temperature T.[7]

$$P \propto e^{-\varepsilon/kT} \tag{7}$$

The normalizing constant for the Boltzmann distribution arises from the partition function of the system, Q, which is made of molecular contributions, q. For an ideal gas, Q can be approximated by

$$Q = \frac{q^N}{N!} \tag{8}$$

$$Q \approx \int\int e^{-\hat{\mathbf{H}}(\mathbf{r},\mathbf{p})/kT} \mathrm{d}\mathbf{r}\ \mathrm{d}\mathbf{p} \tag{9}$$

Where $\mathbf{r}$ and $\mathbf{p}$ are the geometry and momentum respectively. The kinetic energy component of the Hamiltonian behaves like an ideal gas, and is thus uninteresting from a computational standpoint. The potential energy component, however, requires the potential energy

[§]solving the energy and forces for a single geometry

computed by DFT. We can obtain Gibb's free energy as

$$G = H - TS = kTV\left(\frac{\delta lnQ}{\delta V}\right)_T - kTlnQ \tag{10}$$

where V is the volume of the system and T is the temperature. The Gibb's free energy is a function H and S, which are the enthalpy and entropy of the system. Enthalpy is a slightly abstract thermodynamic concept relating to the total heat content of the system, and entropy is often interpreted as the disorderedness of the system. For the curious reader, I encourage reading the introductory chapters of a thermodynamics textbook.

The calculation of macro state properties relies on the value of Q, which for all but the smallest di and tri-atomic systems is intractable. Luckily, we can approximate the system by partitioning it further into its translational, rotational, vibrational and electronic states. This is called the ideal gas, rigid rotor, harmonic-oscillator approximation.

$$\varepsilon = \varepsilon_{trans} + \varepsilon_{rot} + \varepsilon_{vib} + \varepsilon_{elec} \tag{11}$$

$$q = q_{trans}\ q_{rot}\ q_{vib}\ q_{elec} \tag{12}$$

The translational component is modeled by one of the simple, tractable Schrödinger equation models, a molecule in a box.

$$q_{trans} = \left(\frac{2\pi MkT}{h^2}\right)^{3/2} V \tag{13}$$

Where M is the mass of the system, and V is the volume of 1 mole of ideal gas.

The rotational component is similarly modeled using the rigid rotor approximation, which is another tractable model for the Schrödinger equation.

$$q_{rot} = \frac{\sqrt{\pi}}{\sigma}\left(\frac{8\pi^2 kT}{h^2}\right)^{3/2}\sqrt{I_1 I_2 I_3} \tag{14}$$

where σ is the order of the rotational subgroup in the molecular point subgroup[¶] and I_i are the moments of inertia.

The vibrational component may be modeled as a harmonic oscillator, which again has a tractable solution for the Schrödinger equation. However, as the electronic component necessitates QM calculation, a better model for the vibrational component can be obtain

[¶]this is often determined by the software package

from the resulting Hamiltonian. Below is an approximate model based on information from the Hamiltonian.

$$q_{vib} = \prod_{i=1}^{3N-6} \frac{e^{-h\nu_i/2kT}}{1 - e^{-h\nu_i/kT}} \tag{15}$$

where ν_i is the vibrational frequency obtained from the Hamiltonian. For systems at a transition state, there are often only $3N - 7$ degrees of freedom.

The electronic component is not modeled by a tractable approximation, and instead information from a QM calculation is used.

$$q_{elec}^{reactant} = g_0 \tag{16}$$

$$q_{elec}^{TS} = g_0 \, e^{-\Delta E_{TS-react}/kT} \tag{17}$$

where g_0 is the degeneracy of the ground state wave function, and $\Delta E_{TS-react}$ is the energy difference between the reactant geometry and transition state geometry. The value of g is many systems is 1.

While more considerations about the system under study are needed to fully realize accurate calculation of thermodynamic properties, I won't belabor the reader with further equations as this section exists purely to demonstrate the need of accurate DFT calculations for the calculation of reaction rate. Importantly, we need accurate, optimized geometries calculated at the DFT level for the ground state reactant, product, and transition states of the system. The equations above can be used with a specific system to approximate the Gibb's free energy, and thus the reaction rate constant, k.

To summarize the steps needed from a computational standpoint:

1. The catalyst itself must be optimized using DFT without the presence of methane.

2. Through independent, parallel optimizations, a number of rough seed placement of the reactant on the catalyst surface are tested to help combat local energy minima. These are optimized using DFT, but the catalyst surface is held constant.

3. Similar to the above step, the product is optimized on the surface of the catalyst.

4. In order to find the transition state, a nudged elastic band optimization is used to find the saddle point between the reactant and product. This involves optimizing a

number of intermediate "images", typically 4-6 in parallel. However, this optimization technique is significantly more involved than finding minimum energy geometries.

The steps above form the high-level algorithm for evaluating a catalyst:

Algorithm 1: Evaluation of a catalyst

Data: A catalyst in $\mathbb{R}^{3N}$ with some base metal atoms, b, and doping atoms, d, totaling to N atoms, and one C and four H atoms

Result: $\hat{k}$, the predicted reaction rate constant

Initialization: *catalyst* $\leftarrow$ Optimize atom positions within the catalyst without presence of methane

Input : *ReactantGuess*, *ProductGuess*: initial guess configurations for *catalyst* + methane system, position vectors in $\mathbb{R}^{3N+15}$

Reactant $\leftarrow$ Optimize *ReactantGuess* configuration

Product $\leftarrow$ Optimize *ProductGuess* configuration

$TS \leftarrow$ use NEB with *Start* and *End* to find transition state

$\Delta G_R \leftarrow$ Calculate Gibb's free energy of *Reactant*

$\Delta G_{TS} \leftarrow$ Calculate Gibb's free energy of TS

return $e^{\Delta G_{TS} - \Delta G_R}$

2 Machine Learning for Quantum Chemistry

As described in the previous section, DFT calculations are a major bottleneck in the evaluation of catalysts. Two areas of improvement are apparent. One may develop optimizations and approximations to DFT methodology by lowering the time spent calculating the energies and forces, or instead, may focus on hastening convergence of geometry optimizations. This section is split into two parts. First, I give an overview of current research towards using machine learning for PES prediction at QM-level accuracy. Second, I describe the reasoning and theory behind our contribution and the specifics of our dataset and task.

2.1 Literature Review

Machine learning for QM covers a vast range of topics in quantum chemistry (QC), many of which are not directly relevant to the purposes of ground-state geometry optimization, so I will restrict the discussion only to relevant methodologies. Another important aspect of the quantum chemistry field is its inherent reliance on approximate methods to this day. Very few systems can be exactly solved for the Schrödinger equation, so nearly all *ab initio*

methods, such as classic Hartree-Fock, the previously discussed DFT family, and the post-Hartree-Fock methods all rely on small approximations to be tractable. Furthermore, there are a widely used class of semi-empirical methods that already incorporate very basic machine learning methods to interpolate empirical data into calculations. From the atomistic side, known as molecular mechanics, the atomistic approximation already necessitates a physics-based, albeit contrived, force equation to approximate the QM observed forces as classical forces. Below is the AMBER-family force field, which incorporates spring, harmonic, and coulombic potentials.[12] The parameters in this equation, r_{eq}, θ_{eq}, A, B, etc. are generally all hand-tuned values designed to approximate common biological molecules, like proteins and small organic molecules.

$$\begin{aligned} E_{\text{total}} = &\sum_{\text{bonds}} K_r(r - r_{eq})^2 + \sum_{\text{angles}} K_\theta(\theta - \theta_{eq})^2 + \sum_{\text{dihedrals}} \frac{V_n}{2}[1 + \cos(n\phi - \gamma)] \\ &+ \sum_{i<j} \left[\frac{A_{ij}}{R_{ij}^{12}} - \frac{B_{ij}}{R_{ij}^{6}} + \frac{q_i q_j}{\epsilon R_{ij}} \right] \end{aligned} \tag{18}$$

Importantly, at all levels of QC theory approximations are needed. Though many are as physics-based as possible, specifically semi-empirical methods come closest to abandoning physical backing for their regression components. The longstanding inclusion of approximations and alternative representations in the field has paved way for a number of interesting machine learning methods, along with relatively good reception. Architectures that have some level of physical backing are sought after by the field, though other architectures have been explored.

There exist a number of molecule representations that aim to preserve some translation, rotation and permutation equivariance of the energy and forces.[13] Some are specific to the architecture, such as ANI, and will be discussed alongside that methodology, though others are more generic.[14] A popular alternative representation for molecular systems are internal coordinates. Instead of specifying a point cloud of Cartesian coordinates, a seed atom is chosen for the first position, and all other atoms are subsequently defined as functions of distances and angles from the seed atom. This representation is certainly invariant to translation and rotation of the system, though again due to the lack of total order, the system is not permutation-invariant. Because of this, it has had only mild adoption in literature, and the search for architectures operating on Cartesian coordinates has been given more weight. Another popular method has been pairwise atom distance matrices, or pairwise Coulomb matrices, which incorporate both distance and charge information. Again, they suffer from

difficulties with permutation equivariance. At a more abstract level, a molecule can be viewed as a graph, with the vertices being atoms and the edges being bonds. Here, it is trivial to describe atom properties and bond lengths as vertex and edge attributes, though there is no methodology to efficiently include 3D positioning of the atoms in space. Even more abstract are string representations of molecules, the most popular being SMILES, which form a crude language to build an unweighted molecular graph. Both of these methods are not robust for geometry optimization, and are typically used to predict other ground-state properties without knowledge of the ground-state geometry.

2.1.1 Considerations for QM Sysems

Quantum Mechanical systems, or physical systems in general, have a number of interesting systemic properties that differ from other studied systems in ML. Most prominent is the equivariance of the system. Molecules are often defined as a point cloud of atoms, which is a collection of Cartesian coordinates. This representation is easily reconcilable with physics models and computer representation of the system. However, the system as a whole is equivariant to the coordinate system representation. If we translate or rotate the molecule, the energy is invariant, and the internal forces are equivariant to that change. Similarly, the properties are equivariant to atom order permutation, as there is no inherent total order on atoms in a system. This is known as $SE(3)$ equivariance.[13] This poses a significant challenge to many conventional ML architectures, which are sensitive to these changes in the system orientation. Much of the work on ML for QM has been put into designing architectures that incorporate these equivariances, and largely this has come from the theoretical physics community. Similarly, molecules have varying numbers of atoms, so architectures that can handle variable length data are often also needed. Finally, when predicting the energy of the molecule, it is often desirable to not only fit the energy, but also the gradient of the energy, which are the forces obtained from the QM calculation. Gradient fitting is a further constraint on architecture that very few current systems can handle. Beyond gradient fitting, additional constraints are desirable, like the conservation of energy and a differentiable PES.

Additionally, he major dataset against which most implementations are benchmarked is called the QM9 dataset.[15,16] Consisting of 134,000 small, organic molecules made of C, O, N, H, and F, QM9 reports minimum energy geometries and various other electronic and thermodynamic properties. It is considered to sufficiently cover chemical space of interest for small molecule design. This dataset, however, has a number of drawbacks. These molecules are presented as monomers, meaning no intermolecular interaction, and they consist of very

basic elements, bonding and electronic configurations. There are no transition metals, which are often difficult to model and sometimes require solving the more complex Dirac equation. Being so, methodologies that are performant on QM9 may not be appropriate for DRM, which requires intermolecular interaction, metals, and non-trivial bonding. Furthermore, the average QM9 molecule is on the order of 20 atoms, whereas our DRM task requires solving systems of approximately 130 atoms, so scaling should also be considered. One unique aspect of the DRM system, however, is that the number of atoms in the system stays consistent, eliminating the need to handle variable length data.

2.1.2 ANI and SchNet

One of the earlier successes in using Neural Networks (NN) for QC came from ANI, which builds upon a theory of molecule representations proposed by Behler and Parinello.[14,17] They define a symmetry function, where the energy is defined as a sum of atomic contributions. Each contribution is computed from a summed radial function of its surrounding atoms, which is akin to the external field potential used for the single-electron DFT decomposition. This treatment ensures the energy is invariant to the atom ordering and position in space, however it does not provide a robust method for fitting forces acting on an atom, which are equivariant to its position. It can perform geometry optimization through its own gradients, which may not correspond to those of the physical system. Through a number of implementation upgrades, the authors report good performance on a number of benchmarks, including the reproduction of potential energy surfaces.

In 2017, SchNet was published as a competing architecture to ANI.[18,19] Similar to ANI's symmetry function layers, SchNet utilizes a continuous-filter convolution to impose equivariance. Another major difference is that Schnet is able to learn the representation of the system by learning these filters, whereas ANI's representation is set and only the layer parameters are learned. Schnet is additionally built on a graph neural network architecture. By construction, SchNet is also energy conserving, meaning that the total energy of the system is constant and composed of the potential energy of the molecule, and the energy from the forces acting on it. They fit QM-calculated forces in their model, and show that the forces are also equivariant to rotation and translation. Additionally, the authors claim the potential energy surface is continuous, making it viable for geometry optimization.

2.1.3 Symmetry Group Methods

From a slightly different perspective, a number of researchers are working on supporting symmetry group equivariances.[13] Notably, *e*3*nn* comes from a more physics background, but is designed to support $SE(3)$ equivariance.[20,21] This is done through spherical harmonic decomposition of the system, where the point cloud is approximated through a fixed number of spherical harmonic functions and those are then operated on as features. Another more general option is *LieConv*, whose architecture can support the equivariance of arbitrary Lie groups.[22] *LieConv* supports the conservation of energy in systems much like SchNet, however *e*3*nn* does not. Both architectures do not scale directly with the number of atoms, but instead with the detailed-ness of the featurization. For example, in *e*3*nn* the user can specify how many of each class of spherical harmonic function will be used to represent the data. This is beneficial for large systems, however it also poses issues where large systems likely need more detailed representation, thus implicitly scaling the method with the system size. *LieConv* is appealing in its ability to explicitly respect $SE(3)$ equivariance and conserve energy, however its current implementation does not scale to large systems.

2.2 Gaussian Processes for Energy Prediction

Another machine learning method is Gaussian Process Regression (GPR) for the prediction of the energy.[23–25] Before the rise of auto-differentiation and GPU computational methods that allowed NN approaches to become extensible, some work had been done using GPs to predict QC properties. Many methods resulted in intractable problems of data representation, mainly $SE(3)$ equivariance, and relied on alternative representations discussed previously such as symmetry function vectors and pairwise matrices. These bloated representations limited the size of the system that could be used with the GP. Much of the work went towards pairing and designing various Kernel functions for a representation method.

2.3 Gaussian Processes

Gaussian processes (GP) are a class of non-parametric machine learning methods. A Gaussian process attempts to model $P_{X,Y}$ for some training inputs X and corresponding outputs Y, in this case geometries and energies. We assume the energies are drawn from a multivariate Gaussian Distribution. $\mathbf{Y} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$. The covariance matrix, Σ is the key to the GP's power, and can be defined by any valid kernel function. In our case, we use the radial basis function (RBF), which takes the form

$$\mathbf{\Sigma}_{ij} = \tau e^{\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{\sigma^2}} \tag{19}$$

where τ and σ are learnable hyperparameters. $\mathbf{\Sigma}$ can be decomposed into

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix} \tag{20}$$

where $\mathbf{K}$ is the training-training kernel-matrix, $\mathbf{K}_*$ is the training-testing kernel-matrix, and $\mathbf{K}_{**}$ is the testing-testing kernel-matrix, which for a single prediction is just the variance defined by the kernel function for that diagonal element. We can then find the latent function, f as

$$f_*|(\mathbf{Y}, \mathbf{X}) \sim \mathcal{N}(\mathbf{K}_*^T \mathbf{K}^{-1} y, \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*) \tag{21}$$

We can optimize the hyperparameters, θ, by maximizing the marginal likelihood $P(y|\mathbf{X}, \theta)$.

2.4 Gaussian Process with Derivatives

As we also have force data available, we can add additional constraints to the GP by forcing it to also fit the forces, which are the gradient of the energy. The following section has been graciously derived by Dr. Dan Huang, which shows how the forces may also be fit.

The *derivative property* of GPs states that if $f \sim GP(m, k)$ then there are $D + 1$ GPs

$$\begin{aligned}
f &\sim GP(\tilde{m}_1, \tilde{k}_1) && \text{(Original function)} \\
\pi_1 \circ \left(\frac{\partial}{\partial \mathbf{1}} f \right) &\sim GP(\tilde{m}_{1+1}, \tilde{k}_{1+1,1+1}) && \text{(1st component of derivative)} \\
&\vdots \qquad\qquad \vdots \\
\pi_D \circ \left(\frac{\partial}{\partial \mathbf{1}} f \right) &\sim GP(\tilde{m}_{1+D}, \tilde{k}_{1+D,1+D}) && \text{(Dth component of derivative)}
\end{aligned}$$

where

$$\tilde{m}(\mathbf{x}) = \begin{pmatrix} m(\mathbf{x}) \\ \mathbf{0} \end{pmatrix}^{(1+D)\times 1}$$

and

$$\tilde{k}(\mathbf{x},\mathbf{y}) = \begin{pmatrix} (k(\mathbf{x},\mathbf{y}))^{1\times 1} & \left(\frac{\partial k(\mathbf{x},\mathbf{y})}{\partial \mathbf{y}^T}\right)^{1\times D} \\ \left(\frac{\partial k(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}\right)^{D\times 1} & \left(\frac{\partial^2 k(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}\partial \mathbf{y}}\right)^{D\times D} \end{pmatrix}^{(1+D)\times(1+D)}.^{\|}$$

The conditional distribution of the random vector $[(\overline{f}(\mathbf{x}_*), \overline{\frac{\partial}{\partial \mathbf{1}} f}(\mathbf{x}_*))|(f(\mathbf{x}), \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}})]$ is

$$\begin{pmatrix} \overline{f}(\mathbf{x}_*) \\ \left(\overline{\frac{\partial}{\partial \mathbf{1}} f}\right)(\mathbf{x}_*) \end{pmatrix} \sim \mathcal{N}(\overline{\tilde{m}}(\mathbf{x}_*), \overline{\tilde{k}}(\mathbf{x}_*, \mathbf{x}_*))$$

where

$$\overline{\tilde{m}}(\mathbf{x}_*) = \tilde{m}(\mathbf{x}_*) + \tilde{k}(\mathbf{x}_*, \mathbf{x})\tilde{k}(\mathbf{x},\mathbf{x})^{-1}((f(\mathbf{x})\ \frac{\partial}{\partial \mathbf{1}^T} f(\mathbf{x}))^T - \tilde{m}(\mathbf{x}))$$

and

$$\overline{\tilde{k}}(\mathbf{x}_*, \mathbf{y}_*) = \tilde{k}(\mathbf{x}_*, \mathbf{y}_*) - \tilde{k}(\mathbf{x}_*, \mathbf{x})\tilde{k}(\mathbf{x},\mathbf{x})^{-1}\tilde{k}(\mathbf{x}, \mathbf{y}_*)\,.$$

Several observations can be made about the conditional distribution of the random vector $[(\overline{f}(\mathbf{x}_*)\ \overline{\frac{\partial}{\partial \mathbf{1}} f}(\mathbf{x}_*))^T|(f(\mathbf{x}), \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}})]$.

1. The random function

$$\overline{f}_{\mathbf{x}}(\mathbf{x}_*) = [\overline{f}(\mathbf{x}_*)|f(\mathbf{x}), \frac{\partial}{\partial \mathbf{1}} f(\mathbf{x})]$$

$^{\|}$The derivation of the derivative property is somewhat involved. First, we show that $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$ has a multivariate normal distribution with covariance

$$cov(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}, \frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}) = \frac{\partial k(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}\partial \mathbf{y}}$$

by using the limiting definition of the derivative of a random variable. We should use the fact that $(f(\mathbf{x})\ f(\mathbf{x}+\epsilon))^T$ has a multivariate Gaussian distribution that can be decomposed by the i.i.d. Gaussian decomposition property with the appropriate covariance. Second, we show that the covariances

$$cov(f(\mathbf{x}), \frac{\partial f(\mathbf{y})}{\partial \mathbf{y}}) = \frac{\partial k(\mathbf{x},\mathbf{y})}{\partial \mathbf{y}^T}$$

and

$$cov(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}, f(\mathbf{y})) = \frac{\partial k(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}$$

by another application of the limiting definition of a random variable. Finally, we observe that

$$\begin{pmatrix} f(\mathbf{x}) \\ \left(\frac{\partial}{\partial \mathbf{1}} f\right)(\mathbf{x})_1 \\ \vdots \\ \left(\frac{\partial}{\partial \mathbf{1}} f\right)(\mathbf{x})_D \end{pmatrix} \sim \mathcal{N}(\tilde{m}(\mathbf{x}), \tilde{k}(\mathbf{x},\mathbf{x}))$$

so that we can construct f and each $\pi_i \circ \left(\frac{\partial}{\partial \mathbf{1}} f\right)$ as a GP using Kolmogorov's extension theorem and the marginalization property of Gaussian processes.

is a draw from a Gaussian process as

$$\overline{f}_{\mathbf{x}} \sim GP(\overline{\tilde{m}}_1, \overline{\tilde{\tilde{k}}}_{1,1}).$$

Similarly the random functions

$$\left(\overline{\frac{\partial}{\partial \mathbf{1}} f}_{\mathbf{x}}\right)_i (\mathbf{x}_*) = [\overline{\frac{\partial}{\partial \mathbf{1}} f}_{\mathbf{x}}(\mathbf{x}_*) | f(\mathbf{x}), \frac{\partial}{\partial \mathbf{1}} f(\mathbf{x})]$$

are draws from the respective Gaussian processes as

$$\pi_i \circ \left(\overline{\frac{\partial}{\partial \mathbf{1}} f}_{\mathbf{x}}\right) \sim GP(\overline{\tilde{m}}_{1+i}, \overline{\tilde{\tilde{k}}}_{1+i,1+i}) \,.$$

2. The posterior mean obtained from conditioning on the values and derivatives is

$$\begin{aligned}
\overline{\tilde{m}}(\mathbf{x}_*) &= \tilde{m}(\mathbf{x}_*) + \tilde{k}(\mathbf{x}_*, \mathbf{x}) \begin{pmatrix} \alpha \\ \boldsymbol{\beta} \end{pmatrix} \\
&= \tilde{m}(\mathbf{x}_*) + k(\mathbf{x}_*, \mathbf{x})\alpha + \sum_{d=1}^{D} \left(\frac{\partial k(\mathbf{x}_*, \mathbf{x})}{\partial \mathbf{x}}\right)_d \boldsymbol{\beta}_d
\end{aligned}$$

where

$$\begin{pmatrix} \alpha \\ \boldsymbol{\beta} \end{pmatrix} = \tilde{k}(\mathbf{x}, \mathbf{x})^{-1}((f(\mathbf{x}) \ \frac{\partial}{\partial \mathbf{1}^T} f(\mathbf{x}))^T - \tilde{m}(\mathbf{x}))$$

is the solution to a linear system of equations. Note that the α here obtained from conditioning on values and derivatives is different than the values of $\boldsymbol{\alpha}$ one would obtain from conditioning on just the values.**

3. $\frac{\partial}{\partial \mathbf{1}} \overline{\tilde{m}}(\mathbf{x}_*)_1 = (\overline{\tilde{m}}(\mathbf{x}_*)_{1+1}, \ldots, \overline{\tilde{m}}(\mathbf{x}_*)_{1+D})$ so that the derivative of the posterior mean is equivalent to evaluating the posterior mean to obtain the derivatives. Note that $\frac{\partial}{\partial \mathbf{1}} \overline{\tilde{m}}(\mathbf{x}_*)_1 \neq \frac{\partial}{\partial \mathbf{1}} \overline{m}(\mathbf{x}_*)$ so that the derivative of the posterior mean obtained from conditioning on values and derivatives is not equivalent to the derivative of the posterior mean obtained from conditioning on just values.

**The intuition for this is that the weights obtained from constant approximations is different from the weights obtained from linear approximations.

3 Implementation

While Dr. Huang's derivation of of GPs that also fit derivatives shows it is possible from a theoretical standpoint, from an implementation perspective this is not trivial. There are many popular GP libraries for python, primarily Scikit-Learn and GPy, however they do not offer implementations that also fit gradients. Furthermore, it would be beneficial to have a library that performs Bayesian optimization on the GP, though currently no library supports this natively with derivative fitting. GPyTorch, built on a PyTorch backend, does support derivative fitting and ended up being the library we chose to work with. This library unfortunately lacks support for Bayesian optimization, and its C-level implementation also relies heavily on the inclusion of a homoscedastic noise parameter. Our data is known to be incredibly low-noise, as it comes DFT-level theory, so the need to incorporate artificial noise into the model is undesirable. Additionally, we have no mechanism in GPyTorch to implement the conservation of energy, which takes the form of an additional constraint on the gradient of the energy. While the shortcomings of GPyTorch are unideal, it is the currently the only library that supports derivative fitting and can handle the scale of our system. Work beyond the scope of this study is being done towards developing energy conserving, derivative fitting GPs that scale, though it is currently in its infancy.

We developed a method to potentially aid in the convergence of geometry optimization by reducing the number of calls to the DFT function needed. In its basal form, a GP is fit using the RBF kernel to the initial (seed) configuration of the system, along with its energy from a DFT call. The GP becomes a model for the PES, and we can perform a much less costly optimization on the GP to obtain a new candidate geometry. This geometry is evaluated using DFT, and the resulting ground truth energy is added to the GP model. This continues until the desired number of steps or a convergence criteria is met. A number of details a missing from the basal form. First, we use a mechanism described in section 3.2 to perform transfer learning from prior PES optimization data, which helps form the shape of the test GP's PES. Second, we need to choose an optimization scheme for the GP. That may be performed using Bayesian Optimization, which accounts also for the variance, allowing for different sampling schemes. We, however, chose to implement simple gradient descent of the GP posterior mean starting from the previous proposed minimum geometry. This was done using the PyTorch backend and the Adam optimizer.

3.1 Our DRM Dataset

The remainder of the implementation relies heavily on the structure of our dataset. Our dataset for DRM luckily allows us to sidestep the ugliness of $SE(3)$ invariance and even variable length data. Our dataset consists of three geometry optimization runs per molecule CH_4, $CH_3 + H$, CH_3 and H on both Ni and Co substrates, doped with one of 24 possible transition metal doping atoms. There are approximately 2000 geometry and energy observations per molecule for a given substrate. One advantage of the dataset is that the systems are highly homogeneous. The catalyst remains in constant orientation across all systems, and the molecules are defined relative to its top surface. Similarly, the atoms are kept in constant order for all systems, which eliminates the need for permutation invariance. Lastly, constraining the model to only optimize geometries for *one* molecule, we can eliminate the need to handle variable length data. This vastly simplifies the methodology we need to implement, as we can now comfortably work on a $\mathbb{R}^{3N}$ vector of the Cartesian coordinates, instead of a more complex representation.

An additional task is finding an appropriate reference energy. Because the doping atom identity changes with every potential catalyst, there are a differing number of electrons in each system, leading to different absolute energies of the system. Absolute energies in chemistry are generally not meaningful, as we care more about comparing two states of a *single* system. In our case, we are interested in the change in potential energy of interatomic electronic interactions with changes in conformation, as this is the only meaningful difference between systems. It is trivial to compare two geometries of the same system because a simple difference of energies can be used. When comparing two different systems, however, separating the energy contribution of the geometry and the energy contribution from differing numbers of electrons is less trivial. Generally, the systems are brought into a reference view, often comparing against the known global minimum energies for each system. This clearly removes the contribution from differing numbers of electrons. In our case, we are searching for the global minimum, so this is not an option. Another option, however, is to compute the *atomization energy* which the summed energy contributions of each atom alone. This is equivalent to measuring the energy of a system whose geometry consists of each atom infinitely separated from the other atoms in the system, which is a known and valid reference state for any system. This led us to choose atomization energy as our reference energy.

3.2 Transfer Learning through the Posterior Mean

There are a number of ways to transfer learn for Gaussian processes. Most trivially is to transfer learn through the hyperparameters of the kernel, which for the RBF kernel assumes a similar shape and smoothness of the modeled function. Another option is to transfer data between experiments by simply incorporating that previous experiments' data into the GP's training values. Lastly, we can train a GP on data from previous experiments, and then use it as the prior mean of a training GP. The advantage of transferring other experiments' data through the posterior, rather than incorporating it directly into the test GP, is that the test GP is not forced to fit both the current *and* previous experiments' data, rather it is merely biased to trend towards previous experiments' energies in regions of uncertainty. We also perform transfer learning of the hyperparameters of the training GP to the test GP, since we expect the PES to generally share the same shape and smoothness regardless of the doping atom identity.

3.3 Summary of Components

Parser: We developed a parser for VASP output files, called OUTCAR files, which obtain the geometries and corresponding energies and forces in the file. Additionally, the experiment type (CH_4, CH_3, H, etc) is determined from the input file, called the POSCAR file, if available. For atoms included in this dataset, the parser can output both absolute energies and energies relative to the atomiziation energy.

Dataset: Using the parser, we parse all optimization runs provided to us into a dictionary. We also provide wrapper functions to this dictionary that, given a doping atom and substrate atom, gather the data necessary to perform a leave-one-out validation experiment.

Gaussian Process: We developed wrappers for GPyTorch's Gaussian Process with derivatives class, which configures the GP appropriately with an RBF kernel, the appropriate derivatives and the likelihood function through which we obtain the posterior distribution. Optionally, the user can provide another GP as the prior mean, instead of the default zero mean. Additionally in this package are hyperparameter optimizers and helper functions that transfer parameters between GP models via the model state dictionary.

Optimization: We provide a wrapper function for gradient descent optimization,

both of the geometry and hyperparameters using Adam. This can be configured to stop based on gradient convergence, or number of steps taken.

DFT Evaluation: We programmatically write a valid VASP input and configurably call VASP. Once VASP has finished, we parse the resulting OUTCAR file and provide the new energy and forces.

Each of these components are abstracted through their APIs, which down the line will help aid in replacing the GPyTorch implementation with a better, custom implementation. We also provide a closed optimization loop that demonstrates that the framework functions, albeit not optimally.

4 Results and Discussion

To preface this section, the primary goal of this work was to develop a framework that can *perform* optimization of the geometry using a Gaussian process. Actually tuning such a system to make it performant can be incredibly tricky, and was not part of the scope of this work. Similarly, a number of components in this module optimization framework could be significantly improved, either for computational efficiency or better optimization performance. We provide a proof of concept that the scheme works, though we admit in its current form it is far from optimal.

Three major improvements stand out. First, the gradient descent optimization of the geometry on the GP does not account for variance information, and a better Bayesian scheme may be more appropriate. Second, the data utilization in the training GP, whose posterior mean is transferred to the testing GP, is done naively through random sampling. For better performance, both in terms of the objective and efficiency, choosing only critical data points from the dataset may yield better performance by better defining the PES. Similarly, we currently pool all experiments together, but another method may be to fit a training GP to a full DFT optimization of a *single* experiment, as it will provide a clean and homogeneous GP to transfer.[††] Finally, a better GP backend would be beneficial, specifically one that obeys the conservation of energy. Dr. Huang reports from analogous experiments on small

[††]Unfortunately, even with our improved method, testing new frameworks can take up to of 2 weeks. An optimization takes approximately 4 days, and we run multiple to ensure randomized starting conditions are averaged out. These often run sequentially on the cluster. Many of these ideas may be trivial to implement, but aren't included in the results due to the time constraints of the semester.

organic molecules that non energy-conserving GPs very poorly reproduce the PES.[‡‡]. Lastly, in order to provide an advantage over the traditional L-BFGS DFT optimization, we need to develop stopping criteria. This likely will take a form similar to gradient convergence in BFGS methods, as we do obtain the gradient information from the DFT call. This is trivial to implement, though is not currently utilized as we don't have a convergence guarantee from our method.

To preface the below figures, we remind the reader that the task was to develop a framework that supports geometry optimization via GPs fitted with gradients and transfer learning the posterior mean. The figures below reflect a proof that this framework functions, however it is neither optimal nor competitive against VASP at this stage. Much work is needed both on the GP backend and the scheme for transfer learning still.

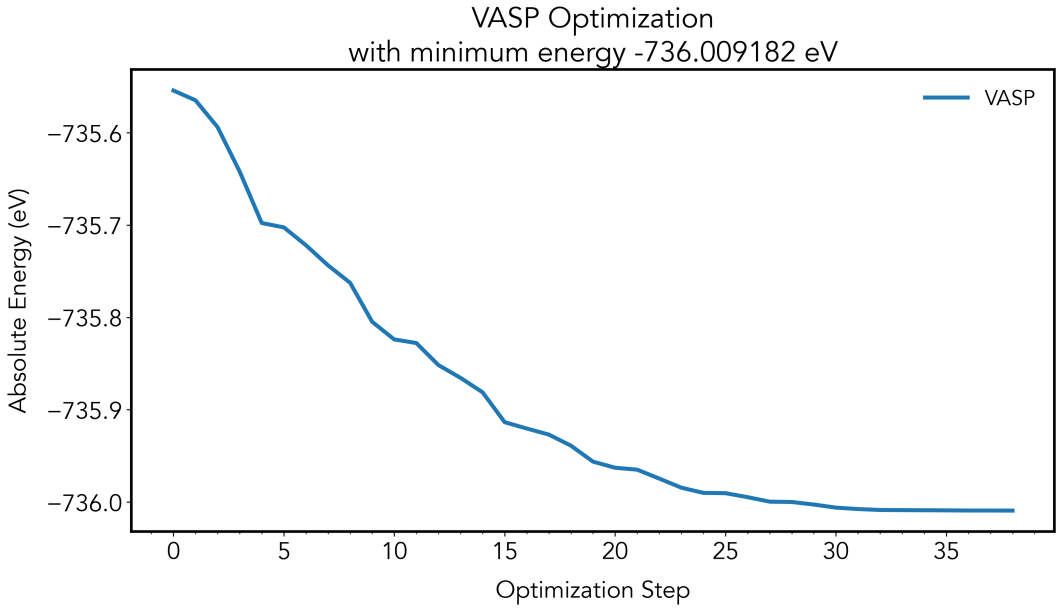

Figure 1: The VASP optimization run takes 38 steps to converge to an energy of -736.009 eV. It demonstrates a consistent decrease in energy along the course of optimization.

[‡‡]Results not published

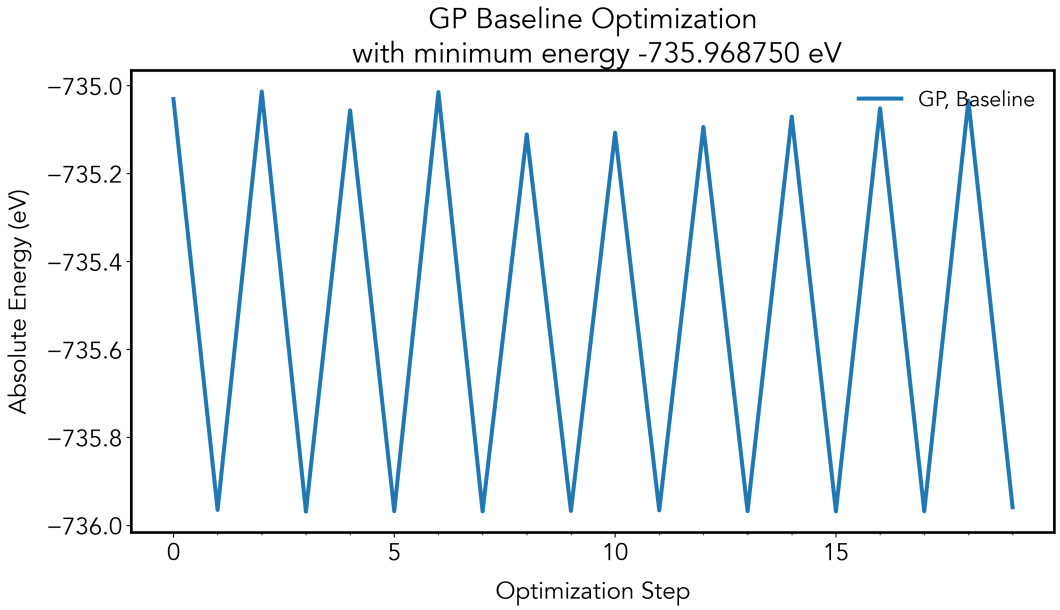

Figure 2: The baseline GP utilizing a 0-mean shows oscillatory behavior during the course of optimization, and has minimum energy of -735.968 eV.

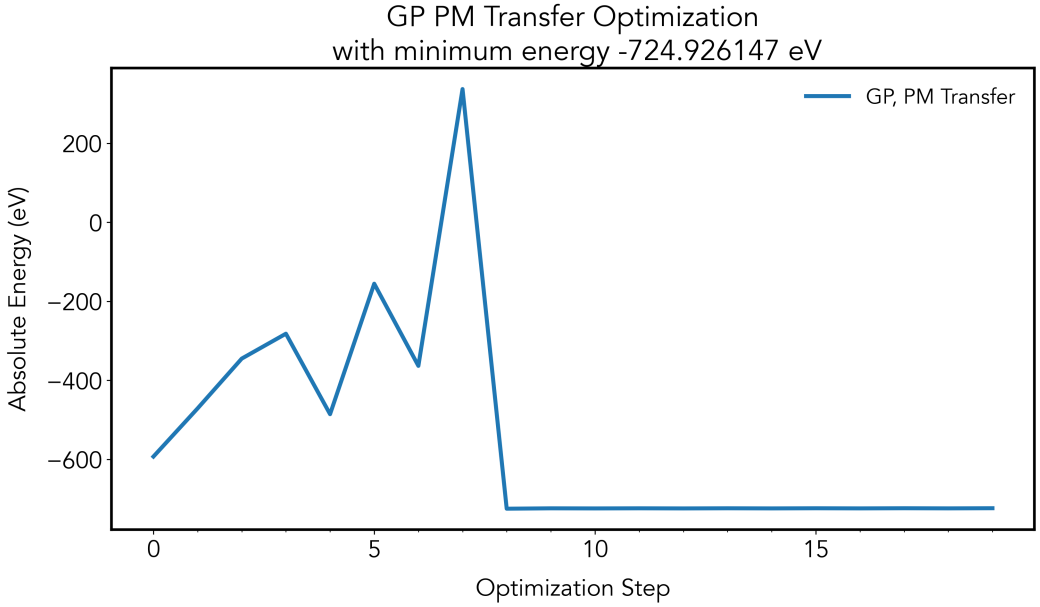

Figure 3: The GP utilizing a transferred GP posterior mean as its prior mean has a minimum energy of -724.926 eV.

Figure 1 demonstrates the reference VASP optimization for this system. It shows clearly a consistently decreasing energy over the steps of the optimization until reaching its gradient

convergence criteria. The baseline GP optimizer in Figure 2 shows oscillatory behavior and no clear convergence within 20 steps. Its steps stay within 1 eV of each other, and its minimum energy geometry is within 0.04 eV of the VASP optimizer. The source of the oscillation is difficult to diagnose since the PES is a $\mathbb{R}^{3N}$ dimensional surface, though we can reason a few things about the GP's PES model. The baseline GP is fit with a 0-mean, so the seed point on the PES will be located below the mean, making a concave region near it. The minimum of the GP posterior mean will be near this point, though this may have no relation to the actual PES' minimum. We believe the GP is rocking in its PES minimum, updating it slightly as it rocks along the $\mathbb{R}^{3N}$ concavity. The transfer learned GP that utilizes a GP posterior mean as its prior mean in Figure 3 currently has the worst performance. It is very difficult to reason about this, as the PES is influenced by the posterior mean of the training GP. It appears, however, that the PES is being globally pulled up, which explains the high energy configurations that are the GP PES minimum at that step. It may be better for future work to focus on reasonable PES dimensionalities that will aid in diagnosing the issues. Similarly, a different GP backend that does not rely so heavily on homoscedastic noise will eliminate issues of hyperparameter tuning from the diagnosis. GPyTorch is unstable with very low noise parameters, but we know our VASP energies are low noise, so the inclusion of noise parameters leads to undesirable behavior. An unexpected outcome of this work is the realization that there are, as of today, no good libraries in Python to fit GPs with gradients, and especially none that conserve energy.

References

1. in. *Fundamental Concepts in Heterogeneous Catalysis* 1–5 (John Wiley & Sons, Ltd, 2014). ISBN: 9781118892114. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118892114.ch1. https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118892114.ch1.

2. Iulianelli, A., Liguori, S., Wilcox, J. & Basile, A. Advances on methane steam reforming to produce hydrogen through membrane reactors technology: A review. *Catalysis Reviews* **58,** 1–35. eprint: https://doi.org/10.1080/01614940.2015.1099882. https://doi.org/10.1080/01614940.2015.1099882 (2016).

3. Jang, W.-J., Shim, J.-O., Kim, H.-M., Yoo, S.-Y. & Roh, H.-S. A review on dry reforming of methane in aspect of catalytic properties. *Catalysis Today* **324.** SI: Green Catal-

ysis, 15–26. ISSN: 0920-5861. https://www.sciencedirect.com/science/article/pii/S092058611830083X (2019).

4. Lavoie, J.-M. Review on dry reforming of methane, a potentially more environmentally-friendly approach to the increasing natural gas exploitation. *Frontiers in Chemistry* **2,** 81. ISSN: 2296-2646. https://www.frontiersin.org/article/10.3389/fchem.2014.00081 (2014).

5. Khairudin, N. F., Sukri, M. F. F., Khavarian, M. & Mohamed, A. R. Understanding the performance and mechanism of Mg-containing oxides as support catalysts in the thermal dry reforming of methane. eng. *Beilstein journal of nanotechnology* **9.** PMC5905271[pmcid], 1162–1183. ISSN: 2190-4286. https://doi.org/10.3762/bjnano.9.108 (Apr. 2018).

6. Akri, M. *et al.* Atomically dispersed nickel as coke-resistant active sites for methane dry reforming. *Nature Communications* **10,** 5181. ISSN: 2041-1723. https://doi.org/10.1038/s41467-019-12843-w (Nov. 2019).

7. Jensen, F. *Introduction to Computational Chemistry* ISBN: 0470011874 (John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006).

8. Bartók, A. P., Kondor, R. & Csányi, G. On representing chemical environments. *Physical Review B* **87.** ISSN: 1550-235X. http://dx.doi.org/10.1103/PhysRevB.87.184115 (May 2013).

9. Galler, A., Canfield, J. & Freericks, J. K. *Schrodinger's original quantum-mechanical solution for hydrogen* 2020. arXiv: 2007.14798 [quant-ph].

10. Kohn, W., Becke, A. D. & Parr, R. G. Density Functional Theory of Electronic Structure. *The Journal of Physical Chemistry* **100,** 12974–12980. eprint: https://doi.org/10.1021/jp9606691. https://doi.org/10.1021/jp9606691 (1996).

11. Kresse, G. & Furthmüller, J. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B* **54,** 11169–11186. https://link.aps.org/doi/10.1103/PhysRevB.54.11169 (16 Oct. 1996).

12. Tian, C. *et al.* ff19SB: Amino-Acid-Specific Protein Backbone Parameters Trained against Quantum Mechanics Energy Surfaces in Solution. *Journal of Chemical Theory and Computation* **16.** PMID: 31714766, 528–552. eprint: https://doi.org/10.1021/acs.jctc.9b00591. https://doi.org/10.1021/acs.jctc.9b00591 (2020).

13. Smidt, T. Euclidean Symmetry and Equivariance in Machine Learning. https://chemrxiv.org/articles/preprint/Euclidean_Symmetry_and_Equivariance_in_Machine_Learning/12935198 (Sept. 2020).

14. Smith, J. S., Isayev, O. & Roitberg, A. E. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chem. Sci.* **8,** 3192–3203. http://dx.doi.org/10.1039/C6SC05720A (4 2017).

15. Ruddigkeit, L., van Deursen, R., Blum, L. C. & Reymond, J.-L. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling* **52.** PMID: 23088335, 2864–2875. eprint: https://doi.org/10.1021/ci300415d. https://doi.org/10.1021/ci300415d (2012).

16. Ramakrishnan, R., Dral, P. O., Rupp, M. & von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data* **1,** 140022. ISSN: 2052-4463. https://doi.org/10.1038/sdata.2014.22 (Aug. 2014).

17. Behler, J. & Parrinello, M. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.* **98,** 146401. https://link.aps.org/doi/10.1103/PhysRevLett.98.146401 (14 Apr. 2007).

18. Westermayr, J., Gastegger, M. & Marquetand, P. Combining SchNet and SHARC: The SchNarc Machine Learning Approach for Excited-State Dynamics. *The Journal of Physical Chemistry Letters* **11.** PMID: 32311258, 3828–3834. eprint: https://doi.org/10.1021/acs.jpclett.0c00527. https://doi.org/10.1021/acs.jpclett.0c00527 (2020).

19. Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A. & Müller, K.-R. SchNet – A deep learning architecture for molecules and materials. *The Journal of Chemical Physics* **148,** 241722. eprint: https://doi.org/10.1063/1.5019779. https://doi.org/10.1063/1.5019779 (2018).

20. Batzner, S. *et al. SE(3)-Equivariant Graph Neural Networks for Data-Efficient and Accurate Interatomic Potentials* 2021. arXiv: 2101.03164 [physics.comp-ph].

21. Fuchs, F. B., Worrall, D. E., Fischer, V. & Welling, M. *SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks* 2020. arXiv: 2006.10503 [cs.LG].

22. Finzi, M., Stanton, S., Izmailov, P. & Wilson, A. G. *Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data* 2020. arXiv: 2002.12880 [stat.ML].

23. Denzel, A. & Kästner, J. Gaussian process regression for geometry optimization. *The Journal of Chemical Physics* **148,** 094114. eprint: https://doi.org/10.1063/1.5017103. https://doi.org/10.1063/1.5017103 (2018).

24. Kamath, A., Vargas-Hernández, R. A., Krems, R. V., Carrington, T. & Manzhos, S. Neural networks vs Gaussian process regression for representing potential energy surfaces: A comparative study of fit quality and vibrational spectrum accuracy. *The Journal of Chemical Physics* **148,** 241702. eprint: https://doi.org/10.1063/1.5003074. https://doi.org/10.1063/1.5003074 (2018).

25. Panosetti, C., Engelmann, A., Nemec, L., Reuter, K. & Margraf, J. T. Learning to Use the Force: Fitting Repulsive Potentials in Density-Functional Tight-Binding with Gaussian Process Regression. *Journal of Chemical Theory and Computation* **16.** PMID: 32155065, 2181–2191. eprint: https://doi.org/10.1021/acs.jctc.9b00975. https://doi.org/10.1021/acs.jctc.9b00975 (2020).