


Local Bayesian optimizer for atomic structures

Estefanía Garijo del Río , Jens Jørgen Mortensen , and Karsten Wedel Jacobsen
CAMD, Department of Physics, Technical University of Denmark, 2800 Kongens Lyngby, Denmark

 (Received 11 July 2019; published 5 September 2019)

A local optimization method based on Bayesian Gaussian processes is developed and applied to atomic structures. The method is applied to a variety of systems including molecules, clusters, bulk materials, and molecules at surfaces. The approach is seen to compare favorably to standard optimization algorithms like the conjugate gradient or Broyden-Fletcher-Goldfarb-Shanno in all cases. The method relies on prediction of surrogate potential energy surfaces, which are fast to optimize, and which are gradually improved as the calculation proceeds. The method includes a few hyperparameters, the optimization of which may lead to further improvements of the computational speed.

DOI: [10.1103/PhysRevB.100.104103](https://doi.org/10.1103/PhysRevB.100.104103)

I. INTRODUCTION

One of the great successes of density functional theory (DFT) [1,2] is its ability to predict ground-state atomic structures. By minimizing the total energy, the atomic positions in solids or molecules at low temperatures can be obtained. However, the optimization of atomic structures with density functional theory or higher-level quantum chemistry methods require substantial computer resources. It is therefore important to develop new methods to perform the optimization efficiently.

It is of key interest here that for a given atomic structure a DFT calculation provides not only the total electronic energy but also, at almost no additional computational cost, the forces on the atoms, i.e., the derivatives of the energy with respect to the atomic coordinates. This means that for a system with N atoms in a particular configuration only a single energy value is obtained while $3N$ derivatives are also calculated. It is therefore essential to include the gradient information in an efficient optimization.

A number of well-known function optimizers exploring gradient information exist [3] and several are implemented in standard libraries like the SciPy library [4] for use in Python. Two much-used examples are the conjugate gradient (CG) method and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm. Both of these rely on line minimizations and perform particularly well for a nearly harmonic potential energy surface (PES). In the CG method, a series of conjugated search directions are calculated, while the BFGS method gradually builds up information about the Hessian, i.e., the second derivatives of the energy, to find appropriate search directions.

The Gaussian process (GP) method that we are going to present has the benefit that it produces smooth surrogate potential energy surfaces (SPESs) even in regions of space where the potential is nonharmonic. This leads to a generally improved convergence. The number of algebraic operations that has to be carried out in order to move from one atomic structure to the next is much higher for the GP method than for the CG or BFGS methods; however, this is not of concern for

optimizing atomic structures with DFT, because the electronic structure calculations themselves are so time consuming. For more general optimization problems where the function evaluations are fast, the situation may be different.

Machine learning for PES modeling has recently attracted the attention of the materials modeling community [5–18]. In particular, several methods have focused on fitting the energies of electronic structure calculations to expressions of the form

$$E(\rho) = \sum_{i=1}^n \alpha_i k(\rho^{(i)}, \rho). \quad (1)$$

Here, $\{\rho^{(i)}\}_{i=1}^n$ are some descriptors of the n atomic configurations sampled, $k(\rho^{(i)}, \rho)$ is known as a kernel function, and $\{\alpha_i\}_{i=1}^n$ are the coefficients to be determined in the fit. Since there are n coefficients and n free parameters, the SPES determined by this expression has the values of the calculations at the configurations on the training set.

Here we note that expression (1) can easily be extended to

$$E(\rho) = \sum_{i=1}^n \alpha_i k(\rho^{(i)}, \rho) + \sum_{i=1}^n \sum_{j=1}^{3N} \beta_{ij} \frac{\partial k(\rho^{(i)}, \rho)}{\partial r_j^{(i)}}, \quad (2)$$

where $\{r_j^{(i)}\}_{j=1}^{3N}$ represent the coordinates of the N atoms in the i th configuration. The new set of parameters β_{ij} together with α_i can be adjusted so that not only the right energy of a given configuration $\rho^{(i)}$ is predicted, but also the right forces. This approach has two advantages with respect to the previous one: (i) the information included in the model scales with the dimensionality; (ii) the new model is smooth and has the right gradients.

In the case of systems with many identical atoms or similar local atomic structures it becomes advantageous to construct SPESs based on descriptors or fingerprints characterizing the local environment [5–11]. The descriptors can then be constructed to obey basic principles as rotational and translational symmetries and invariance under exchange of identical atoms. Here we shall develop an approach based on Gaussian processes which works directly with the atomic

coordinates and effectively produces a surrogate PES of the type Eq. (2) aimed at relaxing atomic structures. We note that Gaussian processes with derivatives for PES modeling are a field that is developing fast, with recent applications in local optimization [19] and path determination in elastic band calculations [13,20,21].

II. GAUSSIAN PROCESS REGRESSION

We use Gaussian process regression with derivative information to produce a combined model for the energy E and the forces \mathbf{f} of a configuration with atomic positions $\mathbf{x} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N)$:

$$\mathbf{U}(\mathbf{x}) = (E(\mathbf{x}), -\mathbf{f}(\mathbf{x})) \sim \mathcal{GP}(\mathbf{U}_p(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')), \quad (3)$$

where $\mathbf{U}_p(\mathbf{x}) = (E_p(\mathbf{x}), \nabla E_p(\mathbf{x}))$ is a vector-valued function which constitutes the prior model for the PES and $K(\mathbf{x}, \mathbf{x}')$ is a matrix-valued kernel function that models the correlation between pairs of energy and force values as a function of the configuration space.

In this work, we choose the constant function $\mathbf{U}_p(\mathbf{x}) = (E_p, \mathbf{0})$ as the prior function. For the kernel, we use the squared-exponential covariance function to model the correlation between the energy of different configurations:

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 e^{-\|\mathbf{x} - \mathbf{x}'\|^2/2l^2}, \quad (4)$$

where l is a typical scale of the problem and σ_f is a parameter describing the prior variance at any configuration \mathbf{x} . The full kernel K can be obtained by noting that [22,23]

$$\text{cov}(E(\mathbf{x}), E(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}'), \quad (5)$$

$$\text{cov}\left(E(\mathbf{x}), \frac{\partial E(\mathbf{x}')}{\partial x'_i}\right) = \frac{\partial k(\mathbf{x}, \mathbf{x}')}{\partial x'_i} \equiv J_i(\mathbf{x}, \mathbf{x}'), \quad (6)$$

$$\text{cov}\left(\frac{\partial E(\mathbf{x})}{\partial x_i}, \frac{\partial E(\mathbf{x}')}{\partial x'_j}\right) = \frac{\partial^2 k(\mathbf{x}, \mathbf{x}')}{\partial x_i \partial x'_j} \equiv H_{ij}(\mathbf{x}, \mathbf{x}'), \quad (7)$$

and assembling these covariance functions in a matrix form:

$$K(\mathbf{x}, \mathbf{x}') = \begin{pmatrix} k(\mathbf{x}, \mathbf{x}') & \mathbf{J}(\mathbf{x}, \mathbf{x}') \\ \mathbf{J}(\mathbf{x}', \mathbf{x})^T & H(\mathbf{x}, \mathbf{x}') \end{pmatrix}. \quad (8)$$

The expressions for the mean and the variance for the posterior distribution follow the usual definitions incorporating the additional matrix structure. Let $X = \{\mathbf{x}^{(i)}\}_{i=1}^n$ denote the matrix containing n training inputs and let $Y = \{\mathbf{y}^{(i)}\}_{i=1}^n = \{(E(\mathbf{x}^{(i)}), -\mathbf{f}(\mathbf{x}^{(i)}))\}_{i=1}^n$ be the matrix containing the corresponding training targets. By defining

$$K(\mathbf{x}, X) = (K(\mathbf{x}, \mathbf{x}^{(1)}), K(\mathbf{x}, \mathbf{x}^{(2)}), \dots, K(\mathbf{x}, \mathbf{x}^{(n)})) \quad (9)$$

and

$$(K(X, X))_{ij} = K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}), \quad (10)$$

we get the following expressions for the mean,

$$\begin{aligned} \bar{\mathbf{U}}(\mathbf{x}) &= (\bar{E}(\mathbf{x}), -\bar{\mathbf{f}}(\mathbf{x})) \\ &= \mathbf{U}_p(\mathbf{x}) + K(\mathbf{x}, X) \mathbb{K}_X^{-1} (Y - \mathbf{U}_p(X)), \end{aligned} \quad (11)$$

and the variance,

$$\sigma^2(\mathbf{x}) = K(\mathbf{x}, \mathbf{x}) - K(\mathbf{x}, X) \mathbb{K}_X^{-1} K(X, \mathbf{x}), \quad (12)$$

of the prediction, where $\mathbb{K}_X = K(X, X) + \Sigma_n^2$. Here, we have assumed an additive Gaussian noise term with covariance matrix Σ_n [22]. This term corrects only for the self-covariance of the points in the training set, and thus, it is a diagonal matrix that models the self-correlation of forces with a hyperparameter σ_n^2 and the self-correlation of energies with $\sigma_n^2 \times l^2$. We note that even for computational frameworks where the energy and forces can be computed with very limited numerical noise, small nonzero values of σ_n are advantageous since they prevent the inversion of the covariance matrix $K(X, X)$ from being numerically ill conditioned [13].

In the following, we will refer to $\bar{E}(\mathbf{x})$ as defined in Eq. (11) as the surrogate potential energy surface (SPES) and distinguish it from the first-principles PES, $E(\mathbf{x})$.

III. GAUSSIAN PROCESS MINIMIZER: GPMin

The GP framework can be used to build an optimization algorithm. In this section, we introduce the main ideas behind the proposed Gaussian process minimizer (denoted GPMin from hereon). A more detailed description of the algorithm can be found in the Appendix in the form of a pseudocode.

The GP regression provides a SPES that can be minimized using a gradient-based local optimizer. For this purpose, we have used the L-BFGS-B algorithm as implemented in SciPy [24]. The prior value for the energy is initially set as the energy of the initial configuration and then the expression (11) is used to produce a SPES from that data point alone. This model is then minimized, and the evaluation at the new local minimum generates new data that is then fed into the model to produce a new SPES that will have a different local minimum. Before generating each new SPES the prior value for the energy is updated to the maximum value of the energies previously sampled. This step is important because it makes the algorithm more stable. If a high-energy configuration is sampled, the forces may be very large leading to a too large new step. The increase of the prior value tends to dampen this by effectively reducing the step size. The whole process is then iterated until convergence is reached.

It is illustrative to consider in more detail the first step of the algorithm. It is straightforward to show using Eqs. (4)–(11) that if only a single data point $\mathbf{x}^{(1)}$ is known the SPES is given by

$$\bar{E}(\mathbf{x}) = E^{(1)} - \mathbf{f}^{(1)} \cdot (\mathbf{x} - \mathbf{x}^{(1)}) e^{-\|\mathbf{x} - \mathbf{x}^{(1)}\|^2/2l^2}, \quad (13)$$

where $E^{(1)}$ and $\mathbf{f}^{(1)}$ are the energy and forces of the SPES at the point $\mathbf{x}^{(1)}$, respectively. We have here used that the prior energy is set to the energy of the first configuration $E^{(1)}$. One can confirm that this is the prior energy by noting that points far away from $\mathbf{x}^{(1)}$, where no information is available, take on this value for the energy. It is seen that the initial force $\mathbf{f}^{(1)}$ gives rise to a Gaussian depletion of the SPES. The first step of the GPMin algorithm minimizes the SPES leading to a new configuration,

$$\mathbf{x} = \mathbf{x}^{(1)} + l \frac{\mathbf{f}^{(1)}}{\|\mathbf{f}^{(1)}\|}. \quad (14)$$

The first step is thus in the direction of the force with a step length of l . Considering the information available this is a very natural choice.

GPMIn depends on a number of parameters: the length scale l , the prior value of the energy E_p , the energy width σ_f , and the noise or regularization parameter σ_n . It can be seen from expressions (4) and (11) that the prediction of the SPES depends only on the ratio of σ_f and σ_n and not their individual values.

The prior energy E_p is, as explained above, taken initially as the energy of the first configuration and then updated if larger energies are encountered. It is important that the prior value is not too low to avoid large steps, since the prior energy is the value of the SPES for all configurations far away (on the scale of l) from previously investigated structures.

The scale l is very important as it sets the distance over which the SPES relaxes back to the prior value E_p when moving away from the region of already explored configurations. It therefore also effectively determines a step length in the algorithm.

One interesting advantage of the Bayesian approach is that it allows for update of parameters (usually termed hyperparameters) based on existing data. We investigate this option by allowing the value of the length scale l to change. Since the update procedure also depends on the width parameter σ_f , we update this as well. The updated hyperparameters, $\theta = (l, \sigma_f)$, are determined by maximizing the marginal likelihood:

$$\theta = \arg \max_{\theta} P(Y|X, \theta). \quad (15)$$

The optimization may fail, for example if there is not enough evidence and the marginal likelihood is very flat, and if that happens, the previous scale is kept. The update procedure allows the algorithm to find its own scale as it collects more information, producing a model that self-adapts to the problem at hand. In Sec. VI we shall consider in more depth the adequate choices for the values of the hyperparameters and the different strategies for the update of hyperparameters when the optimizers are applied to DFT calculations.

IV. COMPUTATIONAL DETAILS

We illustrate and test the method on a variety of different systems using two different calculation methods: An interatomic effective medium theory potential (EMT) [25,26] as implemented in ASE [27,28] and DFT. The DFT tests have been performed using GPAW [29] with the local density approximation (LDA) exchange-correlation functional and a plane wave basis set with an energy cutoff at 340 eV. The Brillouin zone has been sampled using the Monkhorst-Pack scheme with a k -point density of $2.0/(\text{\AA}^{-1})$ in all three directions. The PAW setup with one valence electron has been used for the sodium cluster for simplicity. In addition to the default convergence criteria for GPAW, we specify that the maximum change in magnitude of the difference in force for each atom should be smaller than $10^{-4} \text{ eV \AA}^{-1}$ for the self-consistent field iteration to terminate. This improves the convergence of the forces. All systems have been relaxed until the maximum force of the atoms was below 0.01 eV \AA^{-1} .

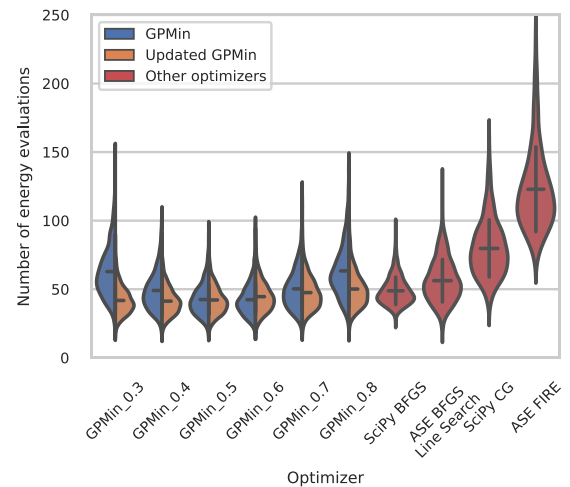


FIG. 1. Statistics of the number of energy evaluations for 1000 relaxations of a 10-atom gold cluster. The initial conditions have been randomly generated. The left-hand side of the plot shows the distribution of the number of energy evaluations for GPMIn in its two variants for scales ranging from 0.3 to 0.8 Å: keeping the scale fixed or allowing it to be updated. The right-hand side shows the performance of other widely used optimizers, which have been sorted according to the average number of function evaluations.

V. EXAMPLE: GOLD CLUSTERS DESCRIBED IN EFFECTIVE MEDIUM THEORY

In the first example GPMIn is used to find the structure of 10-atom gold clusters as described by the EMT potential, and the efficiency is compared with other common optimizers. For this purpose, we generate 1000 random configurations of a 10-atom gold cluster. The configurations are constructed by sequentially applying three uniform displacements for each atom in a cubic box with side length 4.8 Å and only keeping those that lie farther than 1.7 times the atomic radius of gold away from any of the other atoms already present in the cluster. Each configuration is then optimized with different choices of parameters for GPMIn, and, for comparison, the same structures are optimized with the ASE implementations of FIRE [30] and BFGS Line Search, and the SciPy implementations of BFGS and the CG.

For the gold clusters, we have investigated the effect of updating σ_f and l for six different initial scales between 0.3 and 0.8 Å and initial $\sigma_f = 1.0 \text{ eV}$. Since the EMT potential has very small numerical noise, we choose a small value of $\sigma_n/\sigma_f = 5 \times 10^{-4} \text{ eV \AA}^{-1}$ for the regularization. In the update version of the optimizer, we update the scale every fifth iteration.

The statistics of the number of energy evaluations are shown in Fig. 1. The GP optimizers are seen to be the fastest on average, with the appropriate choice of the hyperparameters. For the initial scale of 0.5 Å, for example, the updated version of GPMIn had relaxed the clusters after 42.1 ± 0.3 energy evaluations and the nonupdated one after 42.5 ± 0.3 , as compared to 48.8 ± 0.3 and 56.2 ± 0.5 for the BFGS implementations in SciPy and ASE, respectively. CG exhibits an average number of steps of 79.7 ± 0.7 , and FIRE, 122.9 ± 1.0 .

Figure 1 shows the trend in the performance as the scale is varied. For this system, $l = 0.5 \text{ \AA}$ has the lowest average and variance for GPMIn. The performance depends rather sensitively on the scale parameter: reducing the scale results in a more conservative algorithm where more but smaller steps are needed. Increasing the scale leads to a more explorative algorithm with longer steps that may fail to reduce the energy. In the algorithm with updates, the scale is automatically modified to compensate for a nonoptimal initial scale. The update is particularly efficient for small scales where the local environment is sufficiently explored. For larger scales the sampling is less informative and it takes longer for the algorithm to reduce the scale.

We note that under the appropriate choice of scale, both GPMIn with and without update are among the fastest for the best-case scenario, with 18 evaluations for the regular GPMIn optimizer and 19 for the updated version with scale $l = 0.5 \text{ \AA}$, compared to 19 for ASE BFGS, 27 and 34 for the SciPy implementations of BFGS and CG, respectively, and 70 for FIRE. We further note that the updated version has by far the best worst-case performance.

Of the total of 18 000 relaxations, only 17 failed to find a local minimum. These 17 relaxations were all run with the GPMIn optimizer with $l = 0.8 \text{ \AA}$ without the updates. An optimizer with a too long scale fails to build a successful SPES: the minimum of the SPES often has a higher energy than the previously evaluated point. Thus, we consider that the optimization has failed if after 30 such catastrophic attempts, the optimizer has still not been able to identify a point that reduces the energy or if SciPy's BFGS cannot successfully optimize the predicted SPES.

VI. DETERMINATION OF THE HYPERPARAMETERS

We now continue by considering the use of the GP optimizers more generally for systems with PESs described by DFT. Default values of the hyperparameters should be chosen such that the algorithm performs well for a variety of atomic systems. For this purpose, we have chosen a training set consisting of two different structures: (i) a 10-atom sodium cluster with random atomic positions and (ii) a carbon dioxide molecule on a (111) surface with two layers of gold and a 2×2 unit cell. We have generated 10 slightly different initial configurations for each of the training systems by adding random numbers generated from a Gaussian distribution with standard deviation 0.1 \AA . The training configurations are then relaxed using DFT energies and forces.

For each pair of the hyperparameters $(l, \sigma_n/\sigma_f)$, we relax the training systems and average over the number of DFT evaluations the optimizer needs to find a local minimum. The results are shown in Fig. 2. The plot shows that the metallic cluster benefits from relatively large scales, while the CO on gold system with a tight CO bond requires a shorter scale. A too long scale might even imply that the optimizer does not converge. The set of hyperparameters $l = 0.4 \text{ \AA}$, $\sigma_n = 1 \text{ meV \AA}^{-1}$, and $\sigma_f = 1 \text{ eV}$ seems to be a good compromise between the two cases and these are the default values we shall use in the following.

A similar procedure has been used to determine the default values of the hyperparameters and their initial values in the

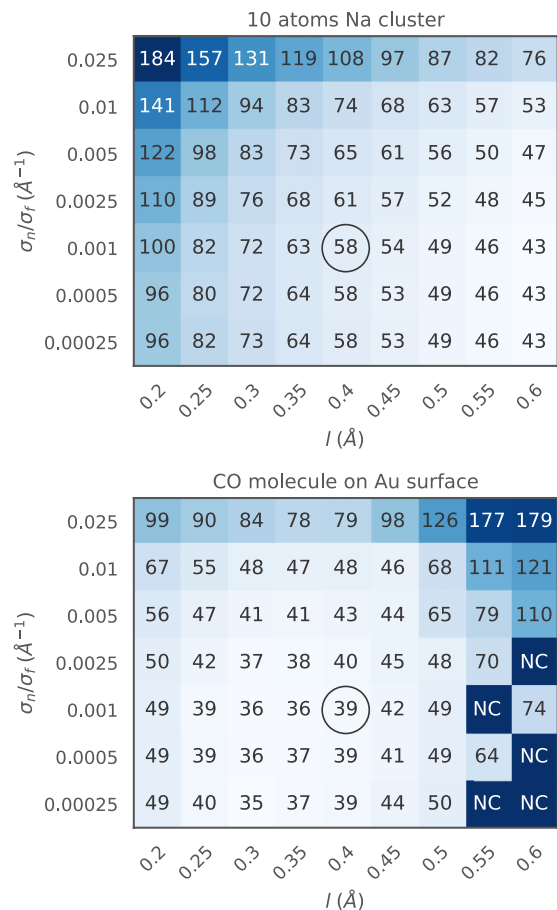


FIG. 2. Average number of potential energy evaluations needed to relax 10 atomic structures as a function of the two hyperparameters: the length scale l , and the regularization parameter σ_n . The label NC (not converged) indicates that at least one of the relaxations did not converge. The default choices for the hyperparameters are indicated by circles.

updated versions of GPMIn. Here, the hyperparameter σ_n/σ_f is kept fixed during the optimization, whereas l and σ_f are determined using expression (15). The value of σ_n/σ_f and the initial values of the other hyperparameters are then determined from the analysis of the performance of the optimizer on the two systems in the training set. The evolution of the hyperparameters depends on the details of the optimization of the marginal likelihood together with the frequency at which the hyperparameters are optimized. Here, we explore three different strategies: Unconstrained maximization of the marginal log-likelihood every 5 energy evaluations (“GPMIn-5”), and two constrained optimization strategies, where the outcome of the optimization is constrained to vary in the range $\pm 10\%$ and $\pm 20\%$ of the value of the hyperparameter in the previous step (“GPMIn-10%” and “GPMIn-20%,” respectively). In the latter two cases we let the optimization take place whenever new information is added to the sample. The algorithm used to maximize the marginal log-likelihood is L-BFGS-B [24] for all strategies.

We have relaxed the same 10 slightly different copies of the two training set systems described before using these three strategies for three different initial values of the scale (0.2,

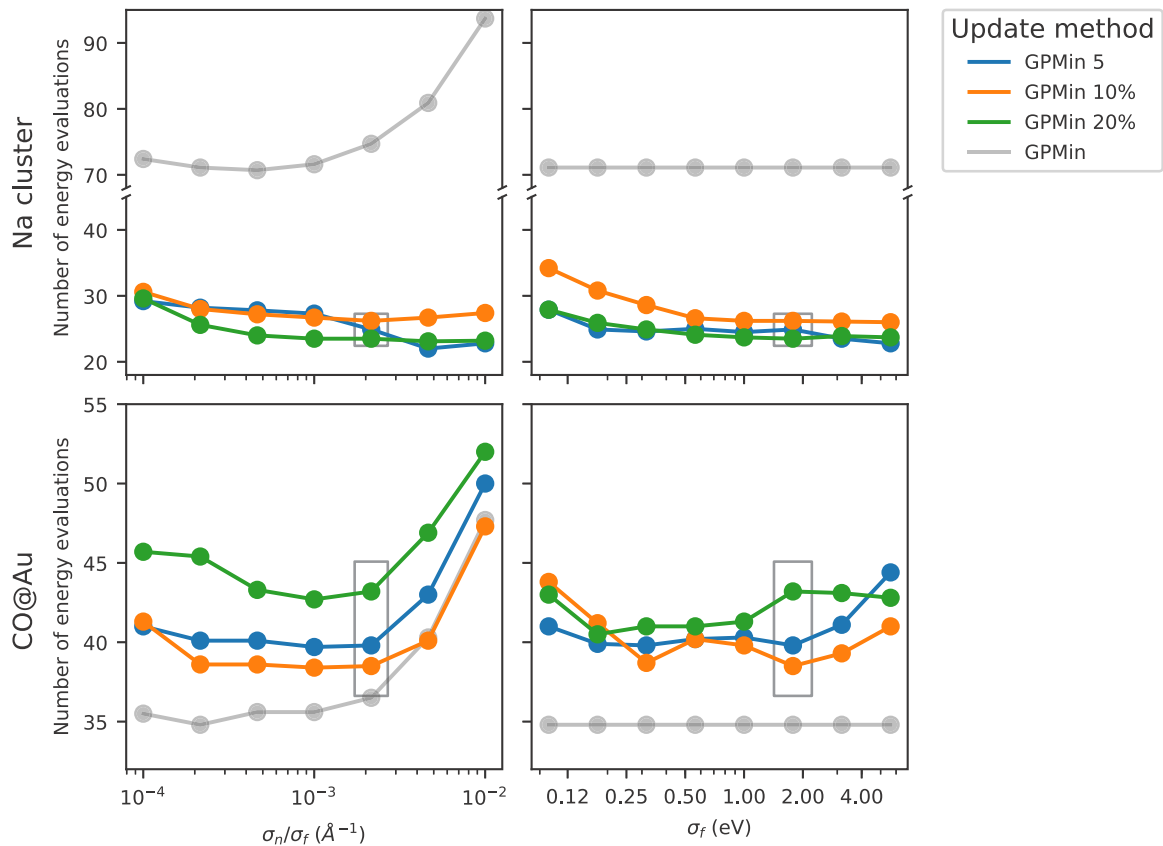


FIG. 3. Average number of energy evaluations needed to relax the two training set systems as a function of the hyperparameter σ_n/σ_f or of the initial value of σ_f , while the other one is kept fixed. The results are shown for the three different updating strategies and compared with the result of running GPMIn without update with the same choice of hyperparameters. The rectangles show the values of the hyperparameter that have been chosen as default values. The value of σ_f chosen in the right panels has been used in the relaxations shown in the left panels, and similarly, the value of σ_n/σ_f that has been found optimal in the left panel is the one that has been used in the relaxations in the right panel.

0.3, and 0.4 Å), eight different initial values of σ_f , and seven different values of the regularization parameter σ_n/σ_f . An overview of the full results can be found in the Supplemental Material [31].

The average numbers of energy evaluations needed to relax the training set for the different strategies and hyperparameters are shown in Fig. 3. The initial value of the scale is chosen as 0.3 Å. The plot shows the variation of the average number of energy evaluations with σ_n/σ_f when the initial value of $\sigma_f = 1.8$ eV and the variation with σ_f when the value of $\sigma_n/\sigma_f = 2 \times 10^{-3}$ Å⁻¹. The performance of the optimizers is seen to depend rather weakly on the parameter values in particular for the sodium cluster. We shall therefore in the following use the values $\sigma_f = 1.8$ eV and $\sigma_n/\sigma_f = 2 \times 10^{-3}$ Å⁻¹.

From the figure it can also be seen that the versions of the optimizer with updates perform considerably better than GPMIn without updates for the sodium cluster, while for the CO molecule on gold, the version without update works slightly better than the three optimizers with updates.

To understand this behavior further we consider in Fig. 4 the evolution of the length scale l as it is being updated. The scale is initially set at three different values $l = 0.2, 0.3, 0.4$ Å. For the sodium cluster the update procedure quickly leads to a much longer length scale around 1.5 Å.

For GPMIn-5 the length scale is raised dramatically already at the first update after five energy evaluations, while for GPMIn-10% and GPMIn-20% the length scale increases gradually because of the constraint built into the methods. The advantage of a longer length scale is in agreement with the results above for the gold cluster described with the EMT interatomic interactions, where a long length scale also led to faster convergence. The situation is different for the CO/Au system, where the update leads first to a significant decrease in the scale and later to an increase saturating at a value around 0.3 Å. This result was to be expected from the one shown in Fig. 2 for the performance of GPMIn without the hyperparameter update. We interpret the variation of the scale for the CO/Au system as being due to the different length scales present in the system, where the CO bond is short and strong while the metallic bonds are much longer. In the first part of the optimization the CO configuration is modified requiring a short scale, while the later stages involve the CO-metal and metal-metal distances. Overall the update of the scale does not provide an advantage over the GPMIn without updates where the scale is kept fixed at $l = 0.4$ Å. It can be seen that the final scales obtained, for example in the case of the sodium cluster optimized with GPMIn-10%, vary by about 30%, where the variation depends on the particular system being optimized and not on the initial value for the length scale.

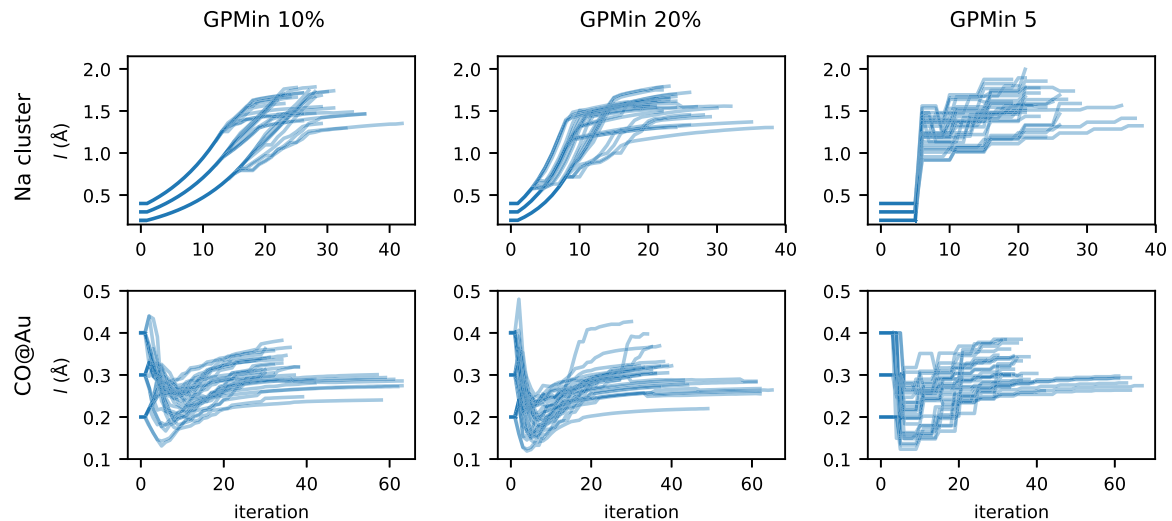


FIG. 4. Evolution of the length scale l with iteration for the three optimizers with update GPMIn-5, GPMIn-10%, and GPMIn-20%. The upper panel in each case shows the results for the sodium cluster, while the lower panel shows the evolution for the CO/Au system. In all cases three different values $l = 0.2, 0.3, 0.4 \text{ \AA}$ for the initial scale have been considered. For the sodium cluster the length scale is seen to increase significantly, while in the case of the CO/Au system, the length scale first decreases and then subsequently increases. The final length scale varies by about 30% dependent on the particular initial structure of the systems.

In the following we shall use $l = 0.3 \text{ \AA}$ as the initial scale for the optimizers with updates. As shown in Figs. S1, S2, and S3 in the Supplemental Material [31], the results do not depend very much on the initial scale in the range 0.2–0.4 \AA . Furthermore, the results for the EMT gold cluster indicate that long length scales should be avoided: it is easier for the algorithm to increase the length scale than to decrease it.

To summarize, we select the following default (initial) values of the hyperparameters for the updated versions of GPMIn: $l = 0.3 \text{ \AA}$, $\sigma_f = 2.0 \text{ eV}$, and $\sigma_n = 0.004 \text{ eV \AA}^{-1}$ ($\sigma_n/\sigma_f = 0.002 \text{ \AA}^{-1}$). These values are used in the rest of this paper.

VII. RESULTS

To test the Bayesian optimizers we have investigated their performance for seven different systems with DFT: a CO molecule on a Ag(111) surface, a C adsorbate on a Cu(100) surface, a distorted Cu(111) surface, bulk copper with random displacements of the atoms with Gaussian distribution and width 0.1 \AA , an aluminum cluster with 13 atoms in a configuration close to fcc, the H_2 molecule, and the pentane molecule. All surfaces are represented by two-layer slabs with a 2×2 unit cell and periodic boundary conditions along the slab. The bulk structure is represented by a $2 \times 2 \times 2$ supercell with periodic boundary conditions along the three unit cell vectors. For each of the systems we have generated ten slightly different initial configurations by rattling the atoms by 0.1 \AA . The resulting configurations are then relaxed using the ASE and SciPy optimizers, together with the different GPMIn optimizers.

It should be noted that in a few cases an optimizer fails to find a local minimum: an atomic configuration is suggested for which GPAW raises an error when it attempts to compute the energy, because two atoms are too close. This happens for

SciPy’s BFGS for one of the CO/Ag configurations and for SciPy’s conjugate gradient method for one of the hydrogen molecule configurations.

The results are collected in Fig. 5. For the sake of clarity, ASE FIRE has been excluded from the plot, since it takes about a factor of three more steps than the fastest optimizer for all systems. The average number of DFT evaluations for the relaxation of the systems in the test set with the implementation of FIRE in ASE is 122 ± 4 for CO/Ag, 91 ± 5 for the pentane molecule, 58 ± 4 for C/Cu, 85 ± 3 for the aluminum cluster, 62 ± 2 for the Cu slab, 53 ± 1 for Cu bulk, and 30 ± 3 for the H_2 molecule.

The GP optimizers are seen to compare favorably or on par with the best one of the other optimizers in all cases. GPMIn without update is on average faster than the other optimizers for 6 of the 7 systems. For the bulk Cu system, it is only slightly slower than the ASE-BFGS algorithm. The updated GP optimizers perform even better with one exception: GPMIn-5 is clearly worse than the other GP optimizers and ASE BFGS for the copper bulk system. Since the atomic displacements from the perfect crystal structure are quite small ($\sim 0.1 \text{ \AA}$), this system is probably within the harmonic regime and requires only a few (~ 10) iterations to converge. The ASE BFGS can therefore be expected to perform well, which is also what is observed in Fig. 5. GPMIn-5 does not update the scale for the first 5 iterations, and when it does so, the new scale does not lead to immediate convergence. The plain GPMIn and the two other optimizers with updates perform on par with ASE BFGS.

Generally, the updated optimizers perform better than GPMIn without updates, and both GPMIn-10% and GPMIn-20% with constrained update perform consistently very well. The updated optimizers are clearly better than the plain GPMIn for the Al cluster, similarly to the behavior for the Na cluster used in the determination of hyperparameters. For the other training system, the CO/Au system, GPMIn was seen to perform

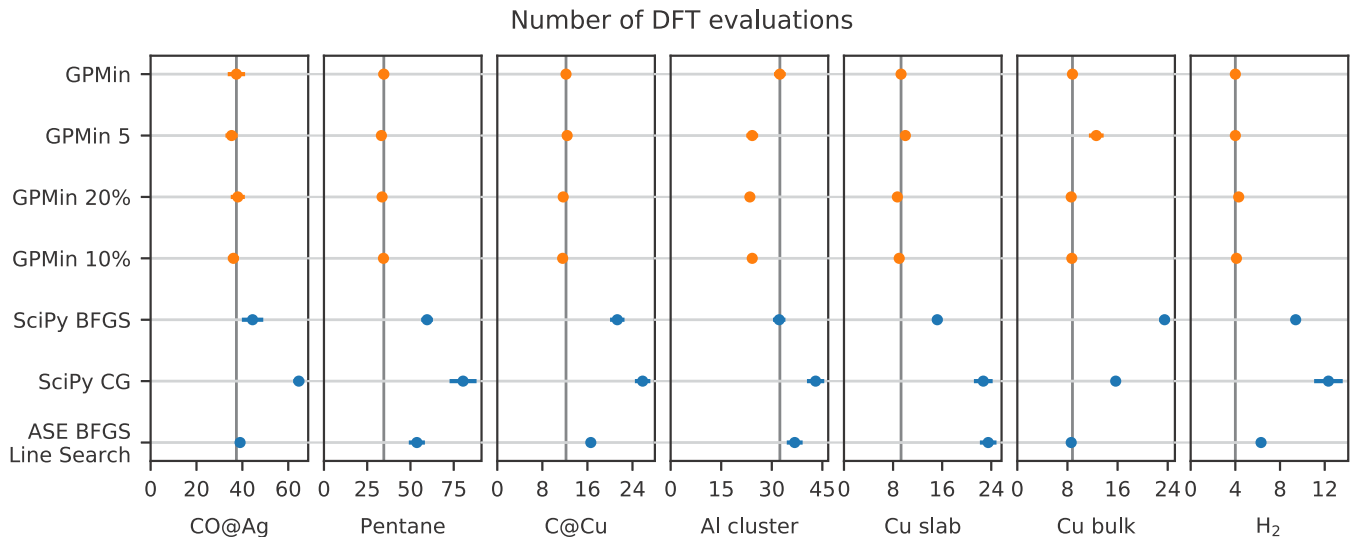


FIG. 5. Number of DFT evaluations required to optimize a given structure. For each structure 10 different initial configurations are generated and optimized. The vertical line represents the average number of steps of GPMIn without parameter updates. The error bar represents the error on the average. A different color has been used to highlight the optimizers of the GPMIn family.

better than all the updated optimizers. However, in Fig. 3 the scale was chosen to be $l = 0.3 \text{ \AA}$, which is superior for that particular system. This behavior does not appear for any of the test systems including the CO/Ag system, which otherwise could be expected to be somewhat similar.

VIII. DISCUSSION

We ascribe the overall good performance of the GP optimizers to their ability to predict smooth potential energy surfaces covering both harmonic and anharmonic regions of the energy landscape. Since the Gaussian functions applied in the construction of the SPES all have the scale l , the SPES will be harmonic at scales much smaller than this around the minimum configuration. If the initial configuration is in this regime the performance of the optimizer can be expected to be comparable to BFGS, which is optimal for a harmonic PES, and this is what is for example observed for the Cu bulk system. We believe that the relatively worse performance of the SciPy implementation of BFGS can be attributed to an initial guess of the Hessian that is too far from the correct one.

Given the performance on both the training and test sets, GPMIn-10% seems to be a good choice. It should be noted that updating the hyperparameters require iteration over the marginal log-likelihood leading to an increased computational cost. However, this is not a problem at least for systems comparable in size to the ones considered here.

The current version of the algorithm still has room for improvement. For example, different strategies for the update of hyperparameters may be introduced. Another, maybe even more interesting, possibility is to use more advanced prior models of the PES than just a constant. The prior model to the PES could for example be obtained from fast lower-quality methods. Somewhat along these lines there have been recent attempts to use previously known semiempirical potentials

for preconditioning more traditional gradient-based optimizers [32,33]. This approach might be combined with the GP framework suggested here.

We also note that the choice of the Gaussian kernel, even though encouraged by the characteristics of the resulting potential [22] and its previously reported success for similar problems [13], is to some extent arbitrary. It would be worthwhile to test its performance against other kernel functions, for example the Matérn kernel, which has been reported to achieve better performance in different contexts [19,34,35]. The kernels used in the work here are also limited to considering only one length scale. More flexible kernels allowing for different length scales for different types of bonds would be interesting to explore.

The probabilistic aspect, including the uncertainty as expressed in Eq. (12), is presently used only in the update of the hyperparameters. It could potentially lead to a further reduction of the number of function evaluations [13]. The uncertainty provides a measure of how much a region of configuration space has been explored and can thereby guide the search also in global optimization problems [16,34,36].

Finally, a note on the limitations of the present version of the optimizer. The construction of the SPES involves the inversion of a matrix [Eq. (11)] which is a square matrix, where the number of columns is equal to $n = N_c(3N + 1)$, where N is the number of atoms in the system and N_c the number of previously visited configurations. This is not a problem for moderately sized systems, but for large systems, where the optimization also requires many steps, the matrix inversion can be very computationally time consuming, and the current version of the method will only be efficient if this time is still short compared to the time to perform the DFT calculations. In addition, this can also result in a memory issue for large systems where the relaxation takes many steps. These issues may be addressed by considering only a subset of the data points or other sparsification techniques. Recently, Wang

et al. [37] showed that by using the black-box matrix-matrix multiplication algorithm it is possible to reduce the cost of training from $O(n^3)$ to $O(n^2)$ and then by using distributed memory and 8 GPUs they were able to train a Gaussian process of $n \sim 4 \times 10^4$ (this would correspond to about 100 steps for 150 atoms with no constraints) in 50 seconds. This time is negligible compared to the time for DFT calculations of systems of this size.

The GPMIn optimizers are implemented in Python and available in ASE [27].

ACKNOWLEDGMENTS

We appreciate fruitful conversations with P. Bjørn Jørgensen. This work was supported by Grant No. 9455 from VILLUM FONDEN.

APPENDIX

The optimization algorithm can be represented in pseudocode as follows:

Input:

Initial structure: $\mathbf{x}^{(0)} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_N)$

Hyperparameters: $l, \sigma_n,$

Tolerance: f_{\max}

$E^{(0)}, \mathbf{f}^{(0)} \leftarrow \text{CALCULATOR}(\mathbf{x}^{(0)})$

$E_p \leftarrow E^{(0)}$

while $\max_i |\mathbf{f}_i^{(0)}| > f_{\max}$ **do**

$X, Y \leftarrow \text{UPDATE}(\mathbf{x}^{(0)}, E^{(0)}, \mathbf{f}^{(0)})$

$E_p \leftarrow \max Y_E$

$\mathbf{x}^{(1)} \leftarrow \text{L-BFGS-B}(\text{GP}(X, Y), \text{start_from} = \mathbf{x}^{(0)})$

$E^{(1)}, \mathbf{f}^{(1)} \leftarrow \text{CALCULATOR}(\mathbf{x}^{(1)})$

while $E^{(1)} > E^{(0)}$ **do**

$X, Y \leftarrow \text{UPDATE}(\mathbf{x}^{(1)}, E^{(1)}, \mathbf{f}^{(1)})$

$E_p \leftarrow \max Y_E$

$\mathbf{x}^{(1)} \leftarrow \text{L-BFGS-B}(\text{GP}(X, Y), \text{start_from} = \mathbf{x}^{(0)})$

$E^{(1)}, \mathbf{f}^{(1)} \leftarrow \text{CALCULATOR}(\mathbf{x}^{(1)})$

if $\max_i |\mathbf{f}_i^{(1)}| > f_{\max}$ **then break**

end if

end while

$\mathbf{x}^{(0)}, E^{(0)}, \mathbf{f}^{(0)} \leftarrow \mathbf{x}^{(1)}, E^{(1)}, \mathbf{f}^{(1)}$

end while

Output: $\mathbf{x}^{(0)}, E^{(0)}$

-
- [1] P. Hohenberg and W. Kohn, *Phys. Rev.* **136**, B864 (1964).
- [2] W. Kohn and L. J. Sham, *Phys. Rev.* **140**, A1133 (1965).
- [3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. (Cambridge University Press, New York, 2007).
- [4] E. Jones, T. Oliphant, and P. Peterson, SciPy: Open source scientific tools for Python, <http://www.scipy.org>.
- [5] A. P. Bartók, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csányi, and M. Ceriotti, *Sci. Adv.* **3**, e1701816 (2017).
- [6] B. Huang and O. A. von Lilienfeld, [arXiv:1707.04146](https://arxiv.org/abs/1707.04146).
- [7] A. Glielmo, C. Zeni, and A. De Vita, *Phys. Rev. B* **97**, 184307 (2018).
- [8] J. Behler and M. Parrinello, *Phys. Rev. Lett.* **98**, 146401 (2007).
- [9] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, *Phys. Rev. Lett.* **108**, 058301 (2012).
- [10] G. Csányi, T. Albaret, M. C. Payne, and A. De Vita, *Phys. Rev. Lett.* **93**, 175503 (2004).
- [11] A. Khorshidi and A. A. Peterson, *Comput. Phys. Commun.* **207**, 310 (2016).
- [12] T. Yamashita, N. Sato, H. Kino, T. Miyake, K. Tsuda, and T. Oguchi, *Phys. Rev. Mater.* **2**, 013803 (2018).
- [13] O.-P. Koistinen, F. B. Dagbjartsdóttir, V. Ásgeirsson, A. Vehtari, and H. Jónsson, *J. Chem. Phys.* **147**, 152720 (2017).
- [14] T. L. Jacobsen, M. S. Jørgensen, and B. Hammer, *Phys. Rev. Lett.* **120**, 026102 (2018).
- [15] M. Todorović, M. U. Gutmann, J. Corander, and P. Rinke, *npj Comput. Mater.* **5**, 35 (2019).
- [16] M. S. Jørgensen, U. F. Larsen, K. W. Jacobsen, and B. Hammer, *J. Phys. Chem. A* **122**, 1504 (2018).
- [17] E. V. Podryabinkin and A. V. Shapeev, *Comput. Mater. Sci.* **140**, 171 (2017).
- [18] K. Gubaev, E. V. Podryabinkin, G. L. Hart, and A. V. Shapeev, *Comput. Mater. Sci.* **156**, 148 (2019).
- [19] A. Denzel and J. Kästner, *J. Chem. Phys.* **148**, 094114 (2018).
- [20] O.-P. Koistinen, V. Ásgeirsson, A. Vehtari, and H. Jónsson, [ChemRxiv \(2019\)](https://arxiv.org/abs/10.26434/chemrxiv.8850440.v1), doi:10.26434/chemrxiv.8850440.v1.
- [21] J. A. Garrido Torres, P. C. Jennings, M. H. Hansen, J. R. Boes, and T. Bligaard, *Phys. Rev. Lett.* **122**, 156001 (2019).
- [22] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning* (MIT, Cambridge, Massachusetts, 2006).
- [23] J. Wu, M. Poloczek, A. G. Wilson, and P. Frazier, in *Advances in Neural Information Processing Systems 30*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017), pp. 5267–5278.
- [24] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *SIAM J. Sci. Comput.* **16**, 1190 (1995).
- [25] K. W. Jacobsen, J. K. Nørskov, and M. J. Puska, *Phys. Rev. B* **35**, 7423 (1987).
- [26] K. W. Jacobsen, P. Stoltze, and J. K. Nørskov, *Surf. Sci.* **366**, 394 (1996).
- [27] Atomic Simulation Environment (ASE), <https://wiki.fysik.dtu.dk/ase>.
- [28] A. H. Larsen, J. J. Mortensen *et al.*, *J. Phys.: Condens. Matter* **29**, 273002 (2017).
- [29] J. Enkovaara, C. Rostgaard, J. J. Mortensen *et al.*, *J. Phys.: Condens. Matter* **22**, 253202 (2010).
- [30] E. Bitzek, P. Koskinen, F. Gähler, M. Moseler, and P. Gumbsch, *Phys. Rev. Lett.* **97**, 170201 (2006).
- [31] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/PhysRevB.100.104103> for a full overview of the average number of DFT calculations needed to relax the structures in the training set with different hyperparameters and strategies for GPMIn with updates.

- [32] J. O. B. Tempkin, B. Qi, M. G. Saunders, B. Roux, A. R. Dinner, and J. Weare, *J. Chem. Phys.* **140**, 184114 (2014).
- [33] L. Mones, C. Ortner, and G. Csányi, *Sci. Rep.* **8**, 13991 (2018).
- [34] D. J. Lizotte, Practical Bayesian optimization, Ph.D. thesis, University of Alberta, 2008.
- [35] G. Schmitz and O. Christiansen, *J. Chem. Phys.* **148**, 241704 (2018).
- [36] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, *Proc. IEEE* **104**, 148 (2016).
- [37] K. A. Wang, G. Pleiss, J. R. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson, [arXiv:1903.08114](https://arxiv.org/abs/1903.08114).