

Gradient-Guided Furthest Point Sampling for Robust Training Set Selection

Morris Trestman^{1,2}, Stefan Gugler^{1,2}, Felix A. Faber⁸, O. A. von Lilienfeld^{1,2,3,4,5,6,7*}

¹*Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany*

²*Berlin Institute for the Foundations of Learning and Data, 10587 Berlin, Germany*

³*Chemical Physics Theory Group, Department of Chemistry,
University of Toronto, St. George Campus, Toronto, ON, Canada*

⁴*Department of Materials Science and Engineering,
University of Toronto, St. George Campus, Toronto, ON, Canada*

⁵*Vector Institute for Artificial Intelligence, Toronto, ON, Canada*

⁶*Department of Physics, University of Toronto, St. George Campus, Toronto, ON, Canada*

⁷*Acceleration Consortium, University of Toronto, Toronto, ON, Canada and*

⁸*Data Science and Modelling, Pharmaceutical Sciences R&D, AstraZeneca, Gothenburg, Sweden*

Smart training set selections procedures enable the reduction of data needs and improves predictive robustness in machine learning problems relevant to chemistry. We introduce Gradient Guided Furthest Point Sampling (GGFPS), a simple extension of Furthest Point Sampling (FPS) that leverages molecular force norms to guide efficient sampling of configurational spaces of molecules. Numerical evidence is presented for a toy-system (Styblinski-Tang function) as well as for molecular dynamics trajectories from the MD17 dataset. Compared to FPS and uniform sampling, our numerical results indicate superior data efficiency and robustness when using GGFPS. Distribution analysis of the MD17 data suggests that FPS systematically under-samples equilibrium geometries, resulting in large test errors for relaxed structures. GGFPS cures this artifact and (i) enables up to two fold reductions in training cost without sacrificing predictive accuracy compared to FPS in the 2-dimensional Styblinski-Tang system, (ii) systematically lowers prediction errors for equilibrium as well as strained structures in MD17, and (iii) systematically decreases prediction error variances across all of the MD17 configuration spaces. These results suggest that gradient-aware sampling methods hold great promise as effective training set selection tools, and that naive use of FPS may result in imbalanced training and inconsistent prediction outcomes.

I. INTRODUCTION

Chemical space is vast, encompassing an immense variety of potential chemical structures and reactions. Even confined to a single potential energy surface (PES), the cost of adequate sampling increases exponentially with the size of the molecular system. Similarity-exploiting machine learning (ML) methods,[1–4] such as kernel ridge regression (KRR) lower this cost by predicting molecular configurations outside of the sampled dataset.[5–15] KRR is particularly suitable for small to intermediate-sized datasets,[16–18] typically in the hundreds of data points range. However, it still requires a number of highly accurate yet expensive reference calculations to train the ML model.[19–26]

Reducing training costs by determining the most information dense training data among the available labeled data remains a challenge. Figure 1 (left) shows a toy example for a diatomic potential: a large set of labeled data (grey) is needed to obtain the ground truth, the Lennard-Jones (LJ) potential (black), but the goal

is to have a smaller, information-dense training set (orange). Datasets used to train such single-PES models, such as MD17[27] and ISO17[28–30], are typically generated via molecular dynamics simulations. As shown in Figure 1 (middle), molecular configurations are sampled according to the Boltzmann distribution (blue). However, training sets selected from Boltzmann distributed data often contain redundancies and under-sample higher energy molecular configurations, reducing model robustness (increasing the incidence of outlier test errors), e.g. when predicting transition state and reactive structures.[31]

A more robust solution to PES prediction is to generate a large, diverse initial dataset, and then select the most informative subset as training points for higher-level reference calculations.[32–35] This concept has given rise to diverse sampling techniques, which aim to generate compact training sets with even coverage of the relevant configuration space, while retaining the predictive power of the larger dataset as possible.[36–42] These sampling techniques stand apart from the common practice of uniform random sampling (URS),[43–45], and can be broadly broken down into unsupervised and supervised methods.

Unsupervised sampling methods do not require labeled data, operating only on molecular descriptors. A well-known method is furthest point sampling (FPS),

* Correspondence email addresses: morris.trestman@tu-berlin.de, stefan.gugler@tu-berlin.de, faber.felix1@gmail.com, anatole.vonlilienfeld@gmail.com

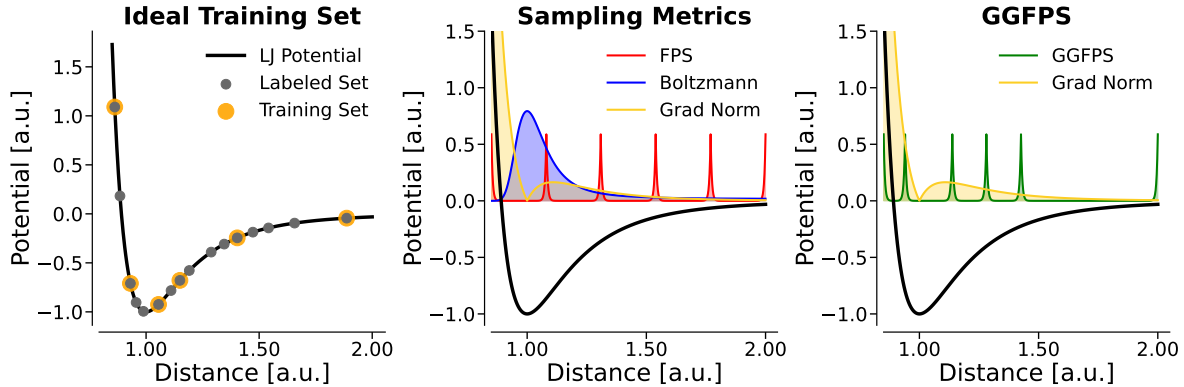


Figure 1. Left: A subset (orange) of labeled data (grey) from the Lennard-Jones potential, which represent a well performing training set from the labeled reaction data. Center: Furthest point (red), Boltzmann (dark blue), and gradient norm (yellow) sampling metrics applied to the Lennard-Jones potential. Right: Gradient norm sampling (yellow), and an instance of the GGFPS sampling metric (green).

which selects new configurations that maximize a distance metric in descriptor space from previously selected configurations.[46, 47] FPS is shown schematically in Figure 1 (center, red). By enforcing even sampling over all of the original dataset, FPS often increases model robustness, usually at the cost of an increase in the overall interpolative test error due to undersampling densely packed dataset regions.[45, 48–51] Other unsupervised dataset sampling methods include FPS variants applied to local atomic environments[52], entropy-based optimization[53, 54] and stratified sampling[38], all aiming to reduce data redundancy and achieve good accuracy-transferability trade-offs.

When training data regression targets (labels) are available, supervised training set sampling methods incorporate them into the sampling strategy. One type of supervised sampling is domain expertise, where FPS training sets are augmented to contain additional strained and transition state structures, in order to bias the data set towards a specific design principle.[49, 55, 56] But this approach may introduce human bias, results in mixed model robustness, and become impracticable for larger datasets and more complex models.[53, 54, 57, 58] Other supervised sampling methods include iteratively adding configurations when predicted energies differ significantly from reference data,[59] and combining distances in feature and target-property spaces.[33, 34] Alternatively, dataset generation procedures such as mindless molecules have been proposed to mitigate human intervention.[60, 61]

In this work, we introduce a supervised sampling method that combines FPS with a Euclidean force norm bias, which we call Gradient Guided Furthest Point Sampling (GGFPS). Forces are the negative gradient of the energy with respect to the atomic positions, are often inherently available in a reference energy calcula-

tions, and have long been incorporated into the kernels and loss functions of ML models.[18, 62–65]. We use the L^2 norm of the gradient as shown in Figure 1 (center, yellow)). Additionally, force and Hessian information have been used successfully in dataset binning for training force fields and for active learning.[66, 67]

Because molecular force norms indirectly describe the variance of molecular energy labels, using force norms as a sampling metric provides a way to cover variance not just in descriptor space, but also in label space. GG FPS contains a gradient bias hyperparameter which tunes the sampling to the variance in property distributions of different labeled datasets. The GG FPS is conceptually illustrated in Figure 1 (right, green). Gradient normals are shown for reference (right, yellow).

Our results demonstrate that while FPS results in lower test errors than URS in low dimensional uniformly distributed datasets, FPS often leaves large sections of non-uniformly distributed datasets like molecular trajectories un-sampled, resulting in worse than random performance. We show that GG FPS training sets span the entire dataset across all tested systems and outperform both URS and FPS.

This paper is structured as follows: In Section II we introduce the GG FPS algorithm, and in Section III we compare and discuss GG FPS to FPS and URS across a toy function and the molecular dynamics trajectories of the MD17 dataset.

II. METHODS

A. Furthest Point Sampling (FPS)

Let $\mathbf{X} = [\mathbf{x}_1^\top, \mathbf{x}_2^\top, \dots, \mathbf{x}_{N_{\text{tot}}}^\top]^\top \in \mathbb{R}^{N_{\text{tot}} \times d}$ be the matrix of all molecular descriptors (each row is a $\mathbf{x}_i \in \mathbb{R}^d$).

Define the labeled dataset as $\mathcal{L} = \{1, 2, \dots, N_{\text{tot}}\}$ i.e., a set of indices into $\mathbf{X}$. The FPS algorithm constructs a training subset $\mathcal{T} \subset \mathcal{L}$ of size $N = |\mathcal{T}|$ through the following procedure: First, the training set is initialized as $\mathcal{T} = \emptyset$ and the unsampled set as $\mathcal{A} = \mathcal{L}$. The algorithm then select initial point $c \in \mathcal{A}$ randomly and updates the training subset $\mathcal{T} \leftarrow \mathcal{T} \cup \{c\}$ and the unsampled set $\mathcal{A} \leftarrow \mathcal{A} \setminus \{c\}$. As long as the size of the training subset is smaller than N_{tot} , more points are selected. So $\mathcal{T} \cup \mathcal{A} = \mathcal{L}$. For this, the minimum Euclidean distances between all points $j \in \mathcal{A}$ and $j \in \mathcal{T}$ are calculated for $\mathbf{d}$ with element

$$d_j = \min_{i \in \mathcal{T}} \mathbf{D}_{ij}$$

where $\mathbf{D}_{ij} = \|\mathbf{x}_j - \mathbf{x}_i\|_2 \in \mathbb{R}^{N \times N}$ is the pairwise distance matrix. We select a new point to add to the training set that is furthest away among all these minimal distances, i.e. select

$$j^* = \arg \max_{j \in \mathcal{A}} d_j. \quad (1)$$

Now the training subset, $\mathcal{T}$, and unsampled set $\mathcal{A}$, are updated as before.

B. Gradient Guided Furthest Point Sampling (GGFPS)

GGFPS extends FPS by including gradient norm information, $\mathbf{g} = \{\|\mathbf{F}_i\|_2\}$, where $\mathbf{F}_i \in \mathbb{R}^{M \times 3}$ are forces in three spatial directions for configuration i , where M is the number of atoms of the system. The algorithm balances geometric spread with importance sampling in high-gradient regions.

The algorithm proceeds as follows: The gradient norms $\mathbf{g}$ and distance matrix $\mathbf{D}_{ij}$ are computed. Training set $\mathcal{T}$ and unsampled set $\mathcal{A}$ are initialized as in FPS and the minimum distances are set to an arbitrarily large number. Among all points $j \in \mathcal{L}$, we sample an initial point c with the probability

$$p_j = \frac{g_j}{\sum_{j \in \mathcal{L}} g_j}. \quad (2)$$

and $\mathcal{T}$, $\mathcal{A}$, $\mathbf{d}$ with $\mathbf{D}_{i,c}$ are updated accordingly. An alternative is to choose as the initial point the configuration that maximizes the gradient norm.

To guide each subsequent selection, GGFPS computes a weighted score for every candidate $j \in \mathcal{A}$. We introduce a gradient norm biasing hyperparameter β that is an exponent over the gradient norms. To prevent overfitting to a specific gradient norm value, the GGFPS algorithm interpolates between $-\beta$ and β .

Rather than fixing a single exponent β' , a hyperparameter $\beta > 0$ defines a range $[-\beta, \beta]$ that is

partitioned into N points, in an alternating fashion: $\{\beta^{(N)}, -\beta^{(1)}, \beta^{(N-1)}, -\beta^{(2)}, \dots\}$ which we re-index as $\{\beta_k\}_{k=1}^N$. β_k flips index signs during interpolation in order to avoid a path dependency, which is problematic because GGFPS, like FPS, first generates a sparse training set that is filled in as data points are added. If GGFPS starts with only large β_k values, then the sparse training set will be comprised of only high gradient norm data points. The β_k index sign flipping ensures that the sparse training set contains both low and high gradient norm data points.

At iteration k , the score

$$s_j = (g_j)^{\beta_k} d_j \quad (3)$$

is used, where d_j is the current minimum distance of point j to the existing set $\mathcal{T}$. When β_k is positive, high-gradient configurations are favored; when β_k is negative, low-gradient configurations are favored; when $\beta_k = 0$, standard FPS is recovered.

At each iteration, analogously to Eq. 1, the candidate point with the highest score

$$j^* = \arg \max_{j \in \mathcal{A}} s_j \quad (4)$$

is selected, and $\mathcal{T}$, $\mathcal{A}$, and $\mathbf{d}$, $\mathbf{D}_{i,j^*}$ are updated. These steps repeat until the desired number of training points N is reached or until N_{tot} is exhausted. The GGFPS algorithm pseudo-code is provided in the SI (Algorithm 1).

C. Kernel Ridge Regression

We use Kernel Ridge Regression (KRR) as the predictive model to evaluate the performance of our GGFPS method. KRR establishes a nonlinear mapping between molecular descriptors from a descriptor space, $\mathbf{x} \in \mathbb{R}^d$ and target properties $y \in \mathbb{R}$ through the kernel trick, which implicitly projects inputs into a reproducing kernel Hilbert space.[2, 4, 68, 69]

For a query molecule with descriptor $\mathbf{x}_q$, the predicted property $\hat{y}_q$ is expressed as

$$\hat{y}_q = \sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}_q) \quad (5)$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a positive-definite kernel function, and $\alpha \in \mathbb{R}^N$ are the dual coefficients learned during training. We use a different kernel function for each of our systems, as described in Section IID.

The matrix formulation for predictions on a test set $\{\mathbf{x}_q\}_{q=1}^{N_{\text{test}}}$ becomes

$$\hat{\mathbf{y}}_{\text{test}} = \mathbf{K}_{\text{test}} \alpha \quad (6)$$

with the kernel between train and test points, $\mathbf{K}_{\text{test}} = k(\mathbf{x}_i, \mathbf{x}_q) \in \mathbb{R}^{N \times N_{\text{test}}}$.

The dual coefficients are obtained through regularized least-squares minimization,

$$\boldsymbol{\alpha} = (\mathbf{K}_{\text{train}} + \lambda \mathbf{I})^{-1} \mathbf{y}_{\text{train}} \quad (7)$$

where $\mathbf{K}_{\text{train}} = k(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}^{N \times N}$ is the training kernel matrix and $\lambda > 0$ is the regularization parameter, while $\mathbf{y}_{\text{train}}$ contains quantum chemical reference values.

D. Representations and kernel functions

We tested our algorithm on the 2D Styblinski-Tang function and the aspirin MD17 trajectory. In this section, we discuss their form and representation.

1. The Styblinski-Tang Function

The Styblinski-Tang (ST) function is a multi-modal, d -dimensional benchmark function used to test optimization algorithms.[70, 71] It contains a mixture of wells surrounded by steep walls and is defined for $d = 2$ as

$$f(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^2 (x_i^4 - 16x_i^2 + 5x_i) \quad (8)$$

where $\mathbf{x} = [x_1, x_2]^T \in \mathbb{R}^2$. The global minimum of the function is located at $\mathbf{x}^* = [-2.903534, -2.903534]$, with a function value of $f(\mathbf{x}^*) = -78.33198$. The configurations are uniformly sampled from the domain $[-4, 4]$ on each coordinate. The Cartesian coordinates are used directly as input descriptors $\mathbf{x}$.

We use a Gaussian kernel function as the similarity metric,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \quad (9)$$

where σ is the bandwidth hyperparameter of the kernel function.

2. The MD17 Trajectories

We represent the configurations of the MD17 aspirin, toluene, malonaldehyde, naphthalene, paracetamol, and uracil trajectories[27] with FCHL19 [72], a faster albeit slightly less accurate version of the original Faber-Christensen-Huang-Lilienfeld (FCHL) representation.[73] FCHL19 is a smooth local representation that contains radial and angular distribu-

tions of atoms across a given molecule’s atomic environments. We use a local Gaussian kernel to compute similarities between configurations, where each kernel element represents the pairwise summation over kernel similarities between the atomic environments of the configurations[74],

$$k(\mathbf{x}_A, \mathbf{x}_B) = \sum_{i \in A} \sum_{j \in B} \delta_{Z_i Z_j} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\right) \quad (10)$$

for atoms i and j in molecules A and B which are the FCHL representations, Z_i and Z_j are their respective nuclear charges, and δ is the Kronecker delta.

III. RESULTS AND DISCUSSION

A. Sampling the Styblinski-Tang function

Here we apply FPS, GGFPS, and URS to the ST function in 2 dimensions. The ST function values and corresponding L^2 gradient norms, $\mathbf{g}$, are shown in Figure 2 (left and center-left, respectively). In both plots, lighter colors denote higher values.

To illustrate the result of the GGFPS sampling, a heatmap of 50 ST function GGFPS training sets, each with 100 data points, is shown in Figure 2 (center-right), where lighter colors indicate a higher sampling density. With $\beta = 0.5$, GGFPS aligns sampling density with the ST function gradient norm surface (center-left), emphasizing high-gradient regions while ensuring surface coverage. Figure 2 (right) shows that for 1,000 samples, URS (black dashed) centers around gradient norms of 20, while FPS (red dash-dotted) peaks at norms of 20 and 60.

Figure 2 (right) also shows the β (solid colored lines) and β' (dotted colored lines) GGFPS distributions. While both β and β' range from 0 to 2, β describes an interpolated range between $[-\beta, \beta]$ (see Section II B), while β' is a constant value. For the ST function, the β and β' KDE distributions are similar for low β values, but diverge as β increases.

With the β range set, we generate ST function data and train models on training sets generated by URS, FPS, and GGFPS. Labeled sets of sizes $\{50, 100, 250, 500, 1,000\}$ are selected via URS. GGFPS and FPS training sets are sub-sampled from these labeled sets, with the remainder used for testing. All sets are bootstrapped 100 times. The KRR hyperparameters are generated using 5-fold grid-search cross-validation (CV). For GGFPS, β is optimized via grid search for each training set fold. This is because the ideal sampling hyperparameter differs for each training set. The β grid contains 20 linearly spaced values from

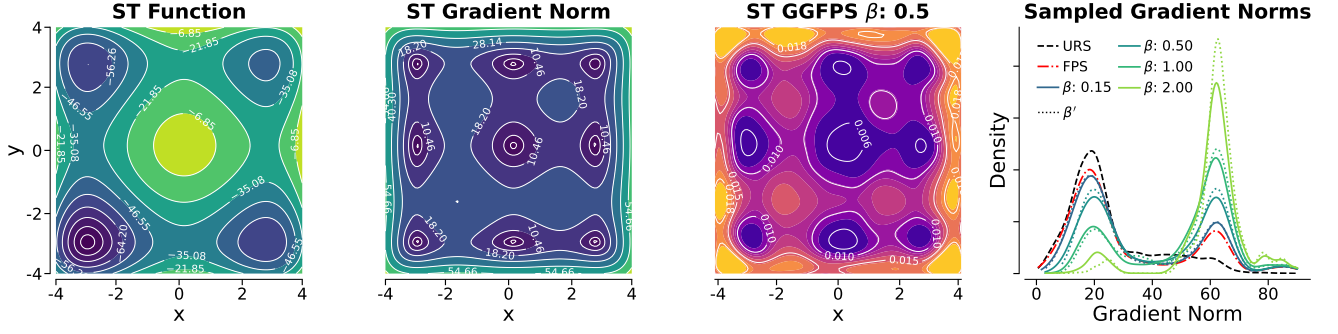


Figure 2. Left: A contour plot of the ST function surface in two dimensions. Center-left: The ST function gradient norm surface. Center-right: a heatmap of 50 ST function training sets generated by GGFPS, each with 100 data points, with a β value of 0.5. In each of the three, a lighter color denotes a higher value. Right: ST function gradient norm distributions of training sets selected via URS (black dashed line), FPS (red dash-dotted line), GGFPS over increasing β values (solid color-coded lines), and the constant β' versions of GGFPS (dotted lines). Training sets were sampled from 50 uniform randomly labeled sets of 1000 data points each.

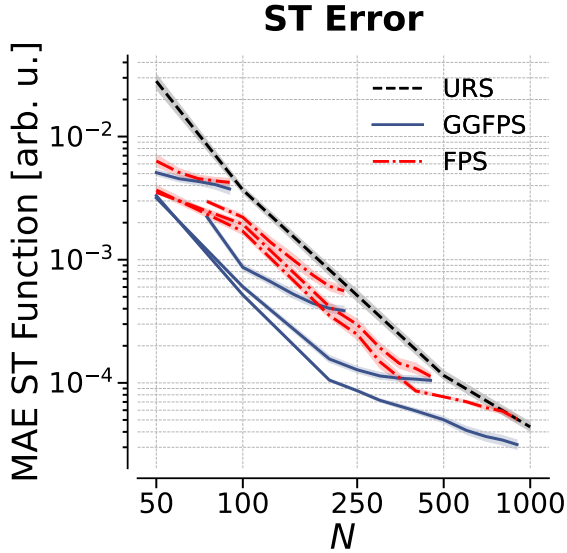


Figure 3. The MAE predictions of the 2D ST function surface with respect to training set size for URS, FPS, and GGFPS (black dashed, red dash-dotted and blue solid lines). The GGFPS β values are optimized per training set fold via grid search. The FPS and GGFPS learning curves are read backwards. E.g. from a labeled set size of 1,000 data points (whose error is shown via URS), FPS and GGFPS sub-select from 950 to 50 training points. Note, at $N = 950$ the GGFPS learning curve is lower than the URS learning curve at $N = 1,000$. This is an artifact of using the MAE metric, and disappears when RMSE is used instead (See SI Figure 9).

0 to 2, and RMSE is used as the cost function.

The ST function learning curves for URS (dashed), GGFPS (solid), and FPS (dash-dotted) are shown in log-log scale in Figure 3, where the y-axis denotes the

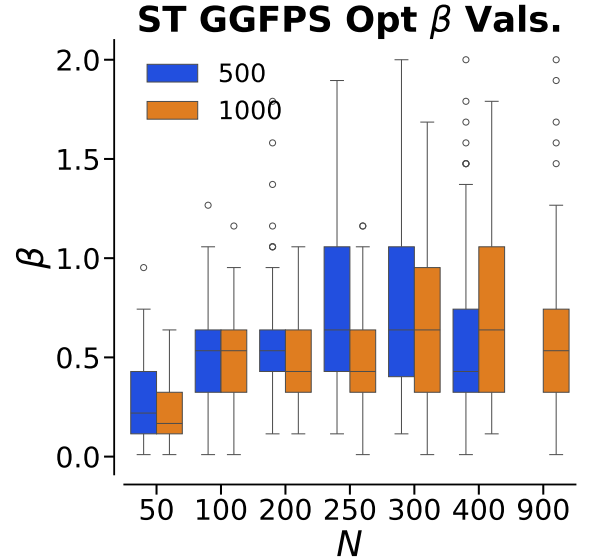


Figure 4. Box plots of the β values corresponding to the lowest GGFPS CV errors with respect to training set size. Blue and orange box plots correspond to training sets drawn from labeled sets with 500 and 1,000 data points, respectively.

test mean absolute error (MAE) and the x-axis shows the training set size, N . The FPS and GGFPS training sets sampled from the labeled set of 1,000 data points start at an N of 950. These 950 data points are the “best” performing out of the initial 1,000. We continue this sampling process until we reach a training set size of 50. The corresponding learning curves are therefore read “backwards” in the plot. The same is true for initial labeled set sizes of 500, 250, and 100. The URS learning curves, by virtue of uniform sampling, are invariant to

the initial labeled set size.

Across all labeled set sizes, FPS requires on average 1.5 times fewer training points than URS to achieve the same predictive accuracy. The performance gap between FPS and URS increases with lower training set sizes for all labeled set sizes.

GGFPS improves upon the predictive performance of FPS by up to a factor of 3 for the same number of training points and the same labeled set size. It also improves predictive efficiency by up to a factor of 2 compared to FPS, achieving the same predictive accuracy with half the number of training points. Compared to URS, GGFPS training sets match the MAE of the entire labeled set with on average one half the total training set size across all labeled set sizes.

We note that smaller initial labeled sets produce GGFPS training sets with lower relative predictive accuracy compared to FPS, for a given N . For a labeled set of 100 data points, the GGFPS MAE matches FPS.

The MAE of the GGFPS training sets comprised of 950 data points is lower than the MAE of the original labeled set of 1,000 data points from which the GGFPS data is selected. The MAE difference disappears when the RMSE is plotted instead of the MAE (see SI Figure 9 (left)), indicating the presence of test error outliers.

The optimal GGFPS β values selected in the KRR CV process are shown in Figure 4 for set sizes of 500 and 1,000 (blue and orange). For both, the spread increases with training set size, implying a flatter loss surface as the training points saturate more of the labeled data space. These distributions are function specific, and reveal a degree of gradient bias for each function surface.

Additionally, the optimal cross-validated GGFPS kernel widths, shown in SI Figure 9 (right), are consistently between 1.5 and 2 times smaller than their URS and FPS counterparts. Smaller kernel widths correspond to training points that are either closer together, or are located in highly varying regions of the function space.

Figure 5 compares the FPS and GGFPS absolute test errors. In the left column, corresponding to 50 training points out of 1,000 labeled points, the median absolute error is of similar magnitude, mirroring the results from Figure 3. However, the FPS models are less robust, with sharper error transitions in the high variance regions of the ST function. GGFPS models maintain robustness across the center and sides because, as shown in Figure 2 (center right), they allocate more training points to the higher variance ST function regions. However, the GGFPS models do accumulate (relatively) more error in the low-gradient corner regions. The MAE values seen in the learning curves can mask the predictive differences between the sampling methods, which is further discussed in Section III B.

The right column shows the error surfaces for training

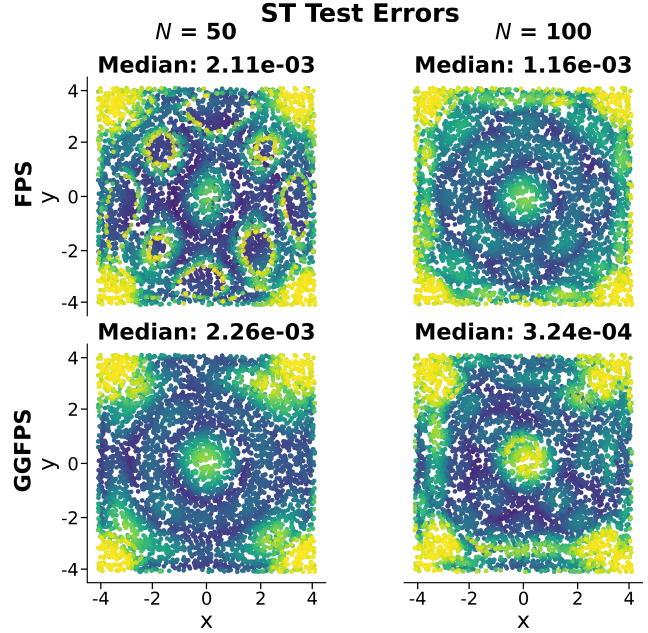


Figure 5. Scatter plots of the 2D ST function absolute test errors corresponding to models trained on FPS (top row) and GGFPS (bottom row), with training set sizes, $N = 50$ (left column) and $N = 100$ (right column). All models start with an initial labeled set of 1,000 points. Median absolute test error values are shown above each scatter plot.

set sizes of 100 data points. Here, the median GGFPS error is five times lower than FPS, and even the highest GGFPS errors remain low relative to FPS. The FPS errors behave more smoothly than at 50 training points because FPS has better saturated the higher variance ST regions, with the exception of the ST function edges and corners. There, sparse FPS sampling results in the highest test errors. GGFPS reduces edge errors due to its high-gradient sampling bias, but performs relatively worse (although still better than FPS) in low-gradient regions near the flat ST function center and next to the function corners. The ST test error surfaces across all training set sizes are shown in SI Figure 10.

Despite performing worse than GGFPS, FPS still provides an attractive sampling method for low dimensional, uniformly distributed systems. However, because FPS does not use label information, one can ‘break’ FPS by constructing an adversarial function where the variance of the dataset labels is localized to a sub-region (e.g. the toy function shown in the SI Figure 11). In this case, FPS will perform no better than URS, because most of the descriptor samples that FPS draws are far from the high label variance region. In contrast, GGFPS dramatically outperforms FPS and URS in this toy example by focusing on the high variance sub-region of the toy function, while still sparsely sampling the rest

of the function surface.

B. Sampling the MD17 Trajectories

While samples from the ST function are uniformly distributed in their descriptor, samples from the chemical potential energy surface are approximately Boltzmann distributed. This is due to the non-physicality of uniformly sampling molecular coordinate spaces, and the curse of dimensionality, which notes that the volume occupied by a bounded space grows exponentially with its dimension, and that most of the volume of the space becomes concentrated at its edges.[75, 76]. Boltzmann distributed datasets introduce the additional difficulty in automated training set selection of adapting not only to the variance of the dataset labels, but also the variance of the dataset density.

Here we apply FPS, GGFPS, and URS to the MD17 aspirin, toluene, malonaldehyde, naphthalene, paracetamol, and uracil trajectories.[27]. Like with the ST function, training set sizes, N , range from 50 to 1,000. Labeled sets consist of $N_{\text{tot}} = 25,000$ configurations. The labeled sets were generated with the sGDML package[17] in order to mirror the property distributions of the entire trajectories, and were bootstrapped 50 times. Like with the ST function, the β sweep ranges were selected through grid-search cross validation, with the sweep bounds ranging from 0 to 2.

The top row of Figure 6 show the energy distributions of the labeled MD17 data sets (gray histogram), and the URS (black dashed line) FPS (red dash-dot line) and GGFPS sampling methods (blue to yellow solid lines), including the constant β' versions (dotted lines), as kernel density estimations (KDEs). Each KDE distribution is comprised of 25 training sets, each with 100 configurations.

Notably, the FPS distributions bias towards both high energy and high force norm configurations across all molecules, under-sampling low to medium energy and force norm structures. Rather than simply under-sample the densest regions of the MD17 configuration space, FPS consistently ‘shifts’ right, implying that the configurations which are ‘furthest’ from each other in representation space correspond to strained structures.

Figure 6 shows that such biases can be corrected with GGFPS, whose β values range from 0 to 2. β values of 0 recover FPS sampling, and for visual clarity the $\beta = 0$ GGFPS distributions are not shown. In contrast to the ST function, here the β' GGFPS distributions differ strongly from their β sweep counterparts, as they consistently overfit to specific regions of configuration space.

For force norms, the GGFPS swept β training sets form an inverse Boltzmann distribution as the β values increase, over-representing equilibrium and

strained structures and under-representing the bulk of the datasets. This can be seen as sampling the regions of configuration space that have the highest variation in density.

The GGFPS training set energy distributions more closely match their FPS counterparts, except that they have better coverage of low energy configurations. Again, the peaks of the GGFPS training set energy distributions more or less match the energy values at which the sampling density of the labeled datasets vary the most.

Figure 7 shows the MAE learning curves (top row) and MAE variances (bottom row) corresponding to the URS, FPS, and GGFPS sampling methods. The results give credence to the observation that FPS poorly samples MD17 trajectories, as both the FPS mean predictive error and predictive variance are consistently higher than their URS counterparts. While the FPS mean errors all broadly converge with their corresponding URS errors at higher training set sizes, their variances do not. Additionally, at lower training set sizes, FPS predictive variances are up to an order of magnitude greater than predictive variances.

In contrast, the GGFPS learning curves show lower predictive errors than URS across all training set sizes for every MD17 molecule. The GGFPS predictive variances are on average 7 times smaller than URS predictive variances for aspirin across all training set sizes, and 5, 2, 5, 3, and 2 times smaller for paracetamol, malonaldehyde, naphthalene, toluene, and uracil, respectively. The optimized GGFPS β values are shown in SI Figure 18.

It is intuitive that models trained on GGFPS training sets have lower predictive variances than their URS counterparts. Because the GGFPS training sets include more uncommon configurations, they therefore have fewer catastrophic errors. Less intuitive is the fact that GGFPS training sets result in lower mean prediction errors than URS training sets. The medium force norm configurations in the middle of the MD17 Boltzmann distributions comprise the vast majority of both the labeled and unlabeled (test) data. GGFPS drastically under-samples these configurations compared to URS, and (invoking the bias/variance tradeoff) one would expect a concurrent increase in GGFPS mean test error, even if the GGFPS test errors for rare configurations are, on average, lower.

We now show how these test errors are distributed across the configuration space in Figure 8 by fixing the training set size to 100 configurations (other training set sizes are shown in the SI Figures 13 to 17), binning the predicted test configurations with respect to their force norm values, and plotting the mean absolute errors and variances of the bins against their force norms. Each bin contains up to 30 configurations, with on average (50 bootstraps $\times$

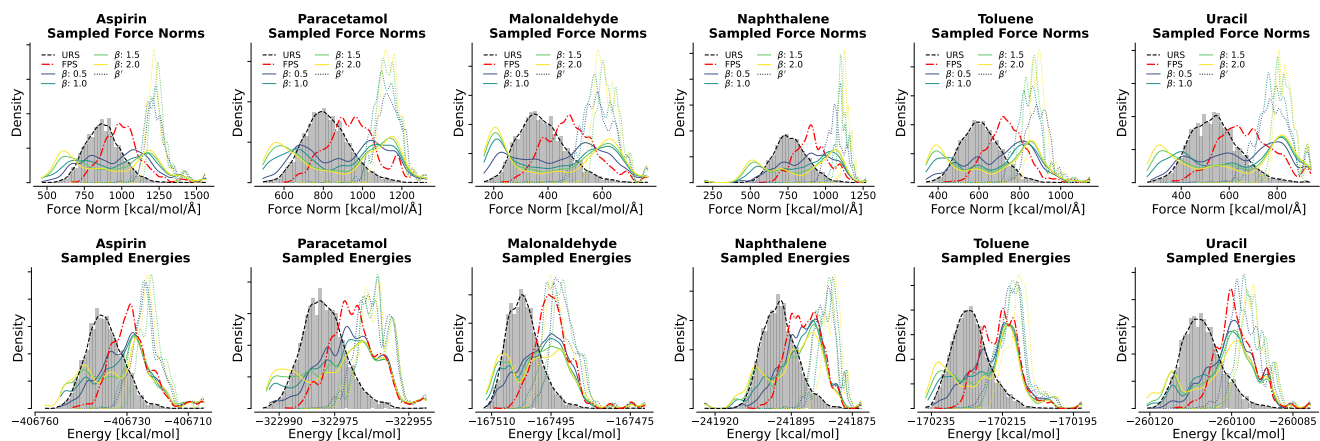


Figure 6. MD17 FCHL19 force norm (top row) and energy (bottom row) distributions of training sets, each with 100 configurations, selected from a labeled set of 20,000 configurations (grey histogram) via URS (black dashed outline), FPS (red dash-dot line), and GGFPS over increasing β values (solid lines from blue to yellow), and the constant β' versions of GGFPS (dotted lines).

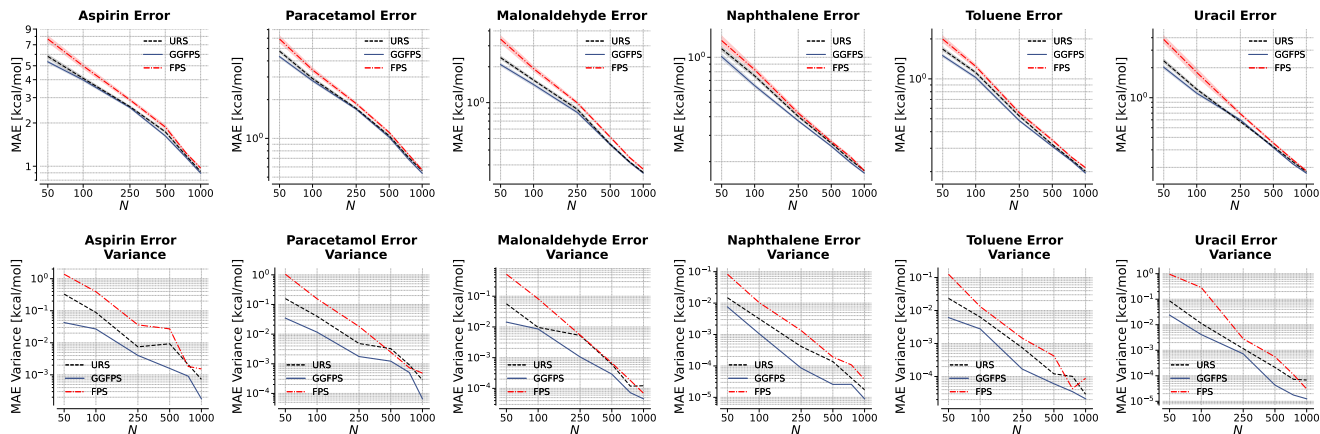


Figure 7. Top row: The MAE predictions of the MD17 trajectories with respect to training set size, N , for URS, FPS, and GGFPS (black dashed, red dot-dashed, and blue solid lines, respectively). Bottom row: The MAE prediction variances of the MD17 trajectories with respect to training set size, N , for URS, FPS, and GGFPS (black dashed, red dot-dashed, and blue solid lines).

50,000 test configurations/30 configurations per bin) 8,333 bins per MD17 molecule. Importantly, the erratic behavior of all three sampling methods at the upper end of the force norms comes from the sparsity of available samples in that regime (see Figure 6, top right), meaning some spikes in error, despite the 50 bootstraps, correspond to only a couple dozen test error instances.

The URS predictive errors and variances are highest at force norm values for which there is limited labeled data. URS errors consistently increase for high force norm configurations, for every MD17 molecule. The errors dip lower for the medium force norm configurations that form the bulk of the Boltzmann distribution. Naphthalene, whose trajectory includes many

relaxed structures far from the bulk of the trajectory dataset, sees a large increase in test errors at low force norms. Conversely, uracil and toluene have low test errors and variances at low force norms, consistent with the paucity of low force norm structures far from the bulk datasets. The other molecules see a slight increase in test errors at low force norms.

FPS training sets result in dramatically worse predictive performance for low force norm configurations across most of MD17, with mean test errors and predictive variances up to twice as high as URS. The two exceptions to this behavior are naphthalene and toluene, which show a slightly higher variance, but the same mean error. One explanation is that naphthalene and toluene are more rigid and less functionalized than the

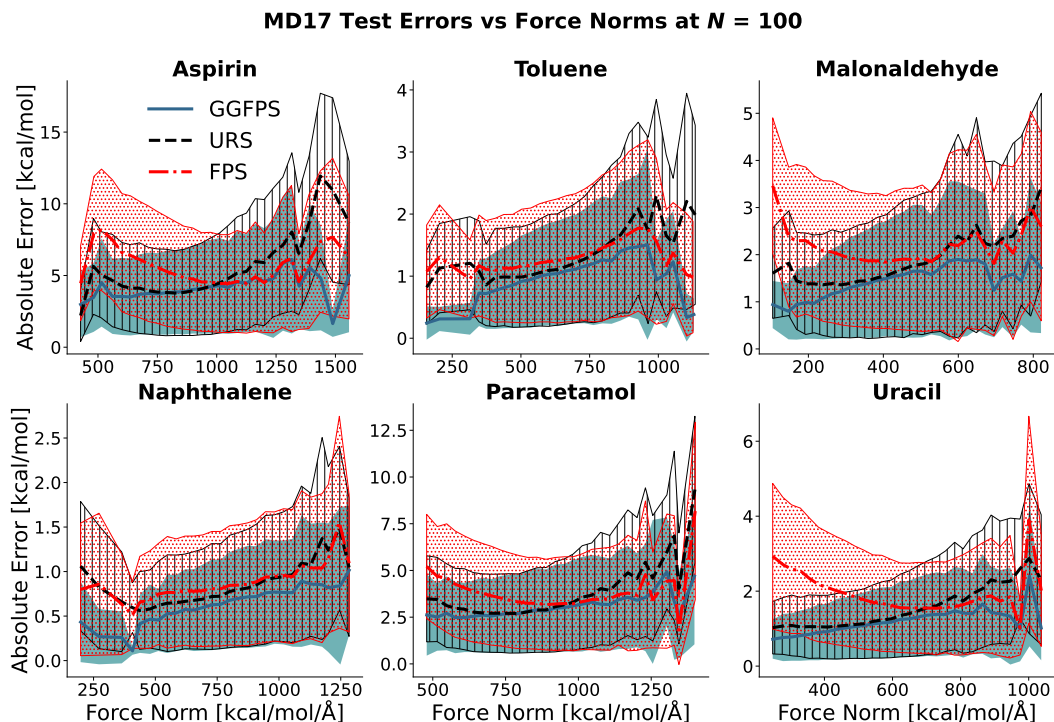


Figure 8. MD17 trajectory absolute test errors versus force norms for FPS, GGFPS, and URS training sets (blue, orange, green), with a training set size (N) of 100 configurations.

other molecules, resulting in a higher number of similar low force norm configurations.

FPS and URS test errors and variances tend to converge for all systems at force norm values corresponding to the FPS training set distribution peaks in Figure 6. Following the increased FPS coverage of high force norm configurations, FPS test error and variance are in general better than URS at higher force norms, with the exceptions of malonaldehyde and naphthalene.

GGFPS outperforms URS and FPS across all force norm values. The most strained configuration energies are predicted on average twice as accurately with GGFPS training sets compared to URS. Also, GGFPS predictive variances for strained structures are on average 4 times smaller than URS variances, and twice as small as FPS variances.

The mean absolute GGFPS errors and error variances are slightly lower than their URS and FPS counterparts for medium force norm configurations, despite the paucity of medium force norm configurations sampled by GGFPS. For toluene, malonaldehyde, and naphthalene, equilibrium and low force norm configurations are predicted on average twice as accurately and with half the variance with GGFPS training sets. GGFPS outperforms URS by an average factor of 1.5 for the remaining molecules.

GGFPS consistently outperforms URS and FPS

across training set sizes, however the improvement is most distinct in the low data regime, before the configuration space becomes saturated. As shown in SI Figure 13, at 50 training configurations GGFPS dramatically outperforms URS and FPS, with up to a four time decrease in test error for low and high force norm configurations compared to URS. GGFPS training sets still show a marked improvement in test error and variance over URS and FPS for high and low force norm configurations at 750 training points, with the exception of malonaldehyde, as seen in Figure 16. By 1,000 training points, while the GGFPS test errors broadly converge with the URS and FPS test errors, the GGFPS variances remain up to twice as small as URS variances for high force norm configurations.

FPS performs poorly on MD17 not just because it under-samples dense regions of the configuration spaces, but also because it systematically under-samples low force norm regions. In contrast, GGFPS also under-samples dense configuration space regions, but through proper coverage over all of the PES, it significantly outperforms FPS across all training set sizes and MD17 molecules.

IV. CONCLUSIONS

We have introduced Gradient Guided Furthest Point Sampling (GGFPS), a supervised sampling method that combines gradient norms with furthest point sampling (FPS) to balance predictive accuracy and robustness. Applied to the 2D Styblinski-Tang (ST) function and the MD17 molecular trajectories, GGFPS demonstrates superior performance over FPS and uniform random sampling (URS). For the ST function, GGFPS achieves the MAE of the full dataset with 50 % fewer training points, and has on average half the test error of FPS for the same number of training points.

On the MD17 trajectories, GGFPS reduces MAE by factors of up to 2 (vs. FPS) and 3 (vs. URS) in high-force norm regions, critical for capturing transition states and strained configurations. GGFPS achieves up to 3 times the predictive accuracy of FPS, and twice the predictive accuracy of URS, for relaxed and equilibrium structures at a given training set size. GGFPS training sets of 50 configurations predict high force norm configurations with equivalent accuracy to URS and FPS training sets of up to 500 configurations. They also predict low force norm configurations with equivalent accuracy to URS training sets of up to 100 configurations, and FPS training sets of up to 250 configura-

tions. The performance increase is most pronounced in the low-data regime, suitable for building fast models. However, significant benefits over FPS are observed up to a training set size of 750 configurations.

GGFPS lowers predictive variance by up to 50 % compared to URS and up to an order of magnitude compared to FPS, ensuring robustness across Boltzmann-distributed and high-dimensional data. The tunable β parameter allows adaptive biasing towards sparse regions of the potential energy surface, addressing FPS’s undersampling of equilibrium structures.

While we have successfully used GGFPS to interpolative learning relevant to molecular dynamics applications where the train and test data are sampled from the same PES, a further area of research is its extension to extrapolative learning, for example across chemical compound space. Another use of GGFPS could be to assist with the selection of diverse systems for generating synthetic novel data which would enable the training of more robust surrogate models of more complex properties.

ACKNOWLEDGEMENTS

S.G. was supported by the Postdoc.Mobility fellowship by the Swiss National Science Foundation (project no. 225476).

-
- [1] C.-J. Simon-Gabriel and B. Schölkopf, *Journal of Machine Learning Research* **19**, 1 (2018).
 - [2] T. Hofmann, B. Schölkopf, and A. J. Smola, *The Annals of Statistics* **36**, 1171 (2008).
 - [3] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms* (The MIT Press, 2001).
 - [4] V. Vapnik, *The nature of statistical learning theory* (Springer science & business media, 2013).
 - [5] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, *Phys. Rev. Lett.* **108**, 058301 (2012).
 - [6] J. Behler and M. Parrinello, *Phys. Rev. Lett.* **98**, 146401 (2007).
 - [7] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, *Phys. Rev. Lett.* **104**, 136403 (2010).
 - [8] O. A. von Lilienfeld, *Angew. Chem. Int. Ed Engl.* **57**, 4164 (2018).
 - [9] K. Hansen, G. Montavon, F. Biegler, S. Fazli, M. Rupp, M. Scheffler, O. A. von Lilienfeld, A. Tkatchenko, and K.-R. Müller, *J. Chem. Theory Comput.* **9**, 3404 (2013), <http://pubs.acs.org/doi/pdf/10.1021/ct400195d>.
 - [10] K. Hansen, F. Biegler, O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, *J. Phys. Chem. Lett.* **6**, 2326 (2015).
 - [11] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, G. E. Dahl, O. Vinyals, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, *J. Chem. Theory Comput.* **13**, 5255 (2017).
 - [12] G. N. Simm and M. Reiher, *J. Chem. Theory Comput.* **14**, 5238 (2018).
 - [13] J. Proppe, S. Gugler, and M. Reiher, *J. Chem. Theory Comput.* **15**, 6046 (2019), 1906.09342.
 - [14] S. Gugler and M. Reiher, *J. Chem. Theory Comput.* **18**, 6670 (2022).
 - [15] S. Gugler and M. Reiher, “Molecular similarity in machine learning of energies in chemical reaction networks,” (2025).
 - [16] Y. Zhang and C. Ling, *npj Comput Mater* **4**, 25 (2018).
 - [17] S. Chmiela, H. E. Sauceda, I. Poltavsky, K.-R. Müller, and A. Tkatchenko, *Comput. Phys. Commun.* **240**, 38 (2019).
 - [18] A. S. Christensen, L. A. Bratholm, F. A. Faber, and O. Anatole von Lilienfeld, *J. Chem. Phys.* **152**, 044107 (2020).
 - [19] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, *Nature* **559**, 547 (2018).
 - [20] L. Himanen, A. Geurts, A. S. Foster, and P. Rinke, *Adv. Sci.* **6**, 1900808 (2019).
 - [21] R. Batra, L. Song, and R. Ramprasad, *Nat Rev Mater* **6**, 655 (2021).
 - [22] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, *npj Comput Mater* **3**, 54 (2017).

- [23] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, *npj Comput Mater* **5**, 1 (2019).
- [24] G. R. Schleider, A. C. M. Padilha, C. M. Acosta, M. Costa, and A. Fazzio, *J. Phys. Mater.* **2**, 032001 (2019).
- [25] P. O. Dral, *J. Phys. Chem. Lett.* **11**, 2336 (2020).
- [26] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, *Rev. Mod. Phys.* **91**, 045002 (2019), 1903.10563.
- [27] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, *Science Advances* **3**, e1603015 (2017).
- [28] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, in *Advances in Neural Information Processing Systems*, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017).
- [29] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, *Nature Communications* **8** (2017), 10.1038/ncomms13890.
- [30] R. Ramakrishnan, P. Dral, M. Rupp, and O. A. von Lilienfeld, *Scientific Data* **1**, 140022 (2014).
- [31] C. van der Oord, M. Sachs, D. P. Kovács, C. Ortner, and G. Csányi, *Npj Comput. Mater.* **9** (2023).
- [32] D. Dragoni, T. D. Daff, G. Csányi, and N. Marzari, *Physical Review Materials* **2** (2018), 10.1103/physrevmaterials.2.013808.
- [33] R. K. Cersonsky, B. A. Helfrecht, E. A. Engel, S. Klavinek, and M. Ceriotti, *Machine Learning: Science and Technology* **2**, 035038 (2021).
- [34] N. Bernstein, G. Csányi, and V. L. Deringer, *npj Computational Materials* **5** (2019), 10.1038/s41524-019-0236-6.
- [35] S. Wengert, G. Csányi, K. Reuter, and J. T. Margraf, *Chemical Science* **12**, 4536–4546 (2021).
- [36] F. Célerse, M. D. Wodrich, S. Vela, S. Gallarati, R. Fabregat, V. Juraskova, and C. Corminboeuf, *Journal of Chemical Information and Modeling* **64**, 1201–1212 (2024).
- [37] T. T. Nguyen, E. Székely, G. Imbalzano, J. Behler, G. Csányi, M. Ceriotti, A. W. Götz, and F. Paesani, *The Journal of Chemical Physics* **148** (2018), 10.1063/1.5024577.
- [38] J. Qi, T. W. Ko, B. C. Wood, T. A. Pham, and S. P. Ong, *npj Computational Materials* **10** (2024), 10.1038/s41524-024-01227-4.
- [39] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi, *Chemical Reviews* **121**, 10073–10141 (2021).
- [40] N. Riquelme-Granada, K. A. Nguyen, and Z. Luo, in *Communications in Computer and Information Science*, Communications in computer and information science (Springer International Publishing, Cham, 2021) pp. 195–222.
- [41] Y. Wen, Z. Li, Y. Xiang, and D. Reker, *Digit. Discov.* (2023).
- [42] H. He and E. A. Garcia, *IEEE Trans. Knowl. Data Eng.* **21**, 1263 (2009).
- [43] A. Mannodi-Kanakkithodi, G. Pilania, T. D. Huan, T. Lookman, and R. Ramprasad, *Scientific reports* **6**, 1 (2016).
- [44] V. Botu, R. Batra, J. Chapman, and R. Ramprasad, *The Journal of Physical Chemistry C* **121**, 511–522 (2016).
- [45] A. P. Bartók, S. De, C. Poelking, N. Bernstein, J. R. Kermode, G. Csányi, and M. Ceriotti, *Science Advances* **3** (2017), 10.1126/sciadv.1701816.
- [46] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II, *SIAM J. Comput.* **6**, 563 (1977).
- [47] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi, in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2 - Conference B: Computer Vision & Image Processing. (Cat. No.94CH3440-5)* (1994) pp. 93–97 vol.3.
- [48] G. Imbalzano, A. Anelli, D. Giofré, S. Klees, J. Behler, and M. Ceriotti, *The Journal of Chemical Physics* **148** (2018), 10.1063/1.5024611.
- [49] P. Rowe, V. L. Deringer, P. Gasparotto, G. Csányi, and A. Michaelides, *The Journal of Chemical Physics* **153** (2020), 10.1063/5.0005084.
- [50] S. Wengert, G. Csányi, K. Reuter, and J. T. Margraf, *Journal of Chemical Theory and Computation* **18**, 4586–4593 (2022).
- [51] N. Boulangeot, F. Brix, F. Sur, and E. Gaudry, *Journal of Chemical Theory and Computation* (2024), 10.1021/acs.jctc.4c00367.
- [52] R. Li, C. Zhou, A. Singh, Y. Pei, G. Henkelman, and L. Li, *The Journal of Chemical Physics* **160** (2024), 10.1063/5.0187892.
- [53] D. Montes de Oca Zapiain, M. A. Wood, N. Lubbers, C. Z. Pereyra, A. P. Thompson, and D. Perez, *npj Computational Materials* **8** (2022), 10.1038/s41524-022-00872-x.
- [54] M. Karabin and D. Perez, *The Journal of Chemical Physics* **153** (2020), 10.1063/5.0013059.
- [55] V. L. Deringer, M. A. Caro, and G. Csányi, *Nature Communications* **11** (2020), 10.1038/s41467-020-19168-z.
- [56] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csányi, A. V. Shapeev, A. P. Thompson, M. A. Wood, and S. P. Ong, *The Journal of Physical Chemistry A* **124**, 731–745 (2020).
- [57] X. Jia, A. Lynch, Y. Huang, M. Danielson, I. Lang’at, A. Milder, A. E. Ruby, H. Wang, S. A. Friedler, A. J. Norquist, and J. Schrier, *Nature* **573**, 251–255 (2019).
- [58] F. Musil, S. De, J. Yang, J. E. Campbell, G. M. Day, and M. Ceriotti, *Chem. Sci.* **9**, 1289 (2018).
- [59] J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev, and A. E. Roitberg, *J. Chem. Phys.* **148**, 241733 (2018).
- [60] T. Gould, B. Chan, S. G. Dale, and S. Vuckovic, *Chemical Science* **15**, 11122–11133 (2024).
- [61] M. Korth and S. Grimme, *Journal of Chemical Theory and Computation* **5**, 993–1003 (2009).
- [62] R. P. Feynman, *Physical Review* **56**, 340–343 (1939).
- [63] G. C. Schatz, in *Lecture Notes in Chemistry* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2000) pp. 15–32.
- [64] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, *Sci Adv* **3**, e1603015 (2017).
- [65] F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, *J. Chem. Phys.* **148**, 241717 (2018).

- [66] T. D. Huan, R. Batra, J. Chapman, S. Krishnan, L. Chen, and R. Ramprasad, npj Computational Materials **3** (2017), 10.1038/s41524-017-0042-y.
- [67] Panknin, Danny, Stefan Chmiela, Klaus Robert Muller, and Shinichi Nakajima., Transactions on Machine Learning Research. (2023).
- [68] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond* (MIT press, 2002).
- [69] D. G. Krige, Journal of the Southern African Institute of Mining and Metallurgy **52**, 119 (1951).
- [70] M. Jamil and X.-S. Yang, International Journal of Mathematical Modelling and Numerical Optimisation **4**, 150 (2013).
- [71] M. Styblinski and T.-S. Tang, Neural Networks **3**, 467 (1990).
- [72] A. S. Christensen, L. A. Bratholm, F. A. Faber, and O. Anatole von Lilienfeld, The Journal of Chemical Physics **152** (2020), 10.1063/1.5126701.
- [73] F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, The Journal of Chemical Physics **148**, 241717 (2018).
- [74] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, Phys. Rev. Lett. **104**, 136403 (2010).
- [75] N. Altman and M. Krzywinski, Nat Methods **15**, 399 (2018).
- [76] R. Bellman, Press, NJ (1961).

V. APPENDIX

Algorithm 1: Gradient-Guided Furthest Point Sampling (GGFPS). For a constant β' value, lines 5–9 can be left out and line 11 will change to $\beta \leftarrow \beta'$.

Input: $\mathbf{g} \in \mathbb{R}^N$ (gradient norms), $\mathbf{D} \in \mathbb{R}^{N \times N}$ (distance matrix), N (target size), $\beta \geq 0$ (gradient exponent)

Output: $\mathcal{T} \subset \mathcal{L}$ (selected indices)

```

1 Initialize  $\mathcal{T} \leftarrow \emptyset$ ,  $\mathcal{A} \leftarrow \mathcal{L}$ ,  $d_j \leftarrow \infty \forall j \in \mathcal{L}$ 
2 Compute  $p_j = g_j / \sum_i g_i \forall j \in \mathcal{L}$ 
3 Sample  $c \sim \text{Categorical}(\mathbf{p})$ 
4 Update  $\mathcal{T} \leftarrow \{c\}$ ,  $\mathcal{A} \leftarrow \mathcal{L} \setminus \{c\}$ ,  $d_j \leftarrow D_{c,j} \forall j \in \mathcal{A}$ 
5 Generate alternating  $\beta$  sequence:
6 Partition  $[-\beta, \beta]$  into  $N$  linearly spaced values:
7    $\beta_{\text{list}} = [\beta_1, \beta_2, \dots, \beta_N]$  where  $\beta_1 = -\beta, \beta_N = \beta$ 
8 Reindex  $\beta_{\text{list}}$  to create alternating sequence  $\{\beta_k\}_{k=1}^N$ :
9    $\beta_k = \begin{cases} \beta_{N - \lfloor (k-1)/2 \rfloor} & \text{if } k \text{ odd} \\ -\beta_{\lfloor k/2 \rfloor + 1} & \text{if } k \text{ even} \end{cases}$ 
10 for  $k = 2$  to  $N$  do
11    $\beta_k \leftarrow$  next value in alternating  $\beta$  sequence
12   foreach  $j \in \mathcal{A}$  do
13      $s_j \leftarrow g_j^{\beta_k} \cdot d_j$ 
14   end
15    $c \leftarrow \text{argmax}_j s_j$ 
16   Update  $\mathcal{T} \leftarrow \mathcal{T} \cup \{c\}$ ,  $\mathcal{A} \leftarrow \mathcal{A} \setminus \{c\}$ 
17   foreach  $j \in \mathcal{A}$  do
18      $d_j \leftarrow \min(d_j, D_{c,j})$ 
19   end
20 end
21 return  $\mathcal{T}$ 

```

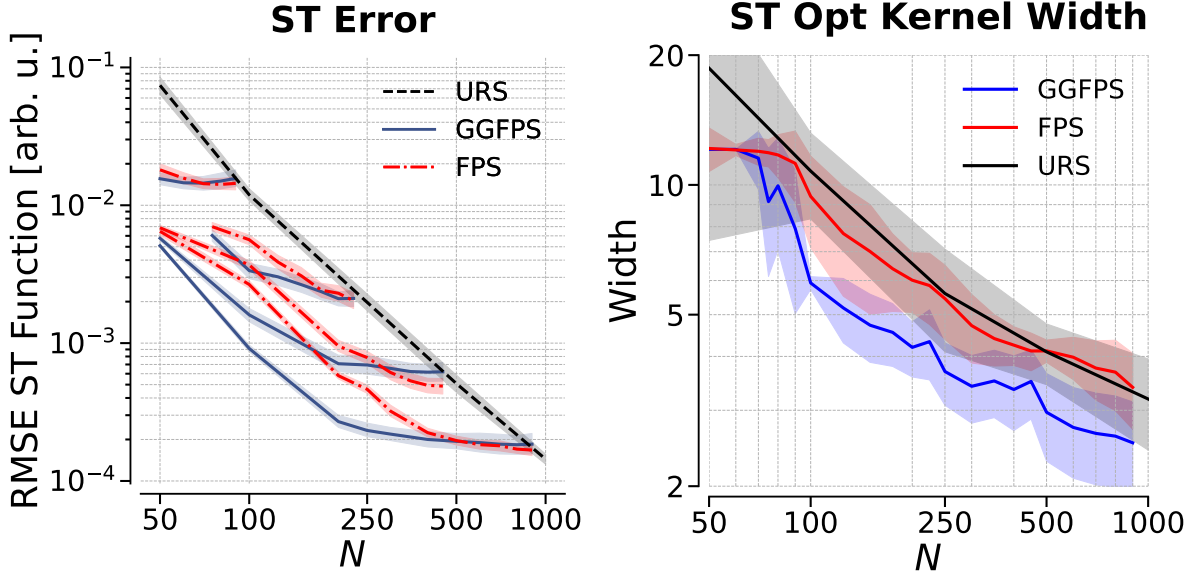


Figure 9. Left: The RMSE predictions of the 2D ST function surface with respect to training set size for URS, FPS, and GGFPS (black dashed, red dash-dotted and blue solid lines). The GGFPS β values are optimized per training set fold via grid search. The FPS and GGFPS learning curves are read backwards. E.g. from a labeled set size of 1,000 data points (whose error is shown via URS), FPS and GGFPS sub-select from 950 to 50 training points. Right: Corresponding optimal kernel widths selected via cross-validation.

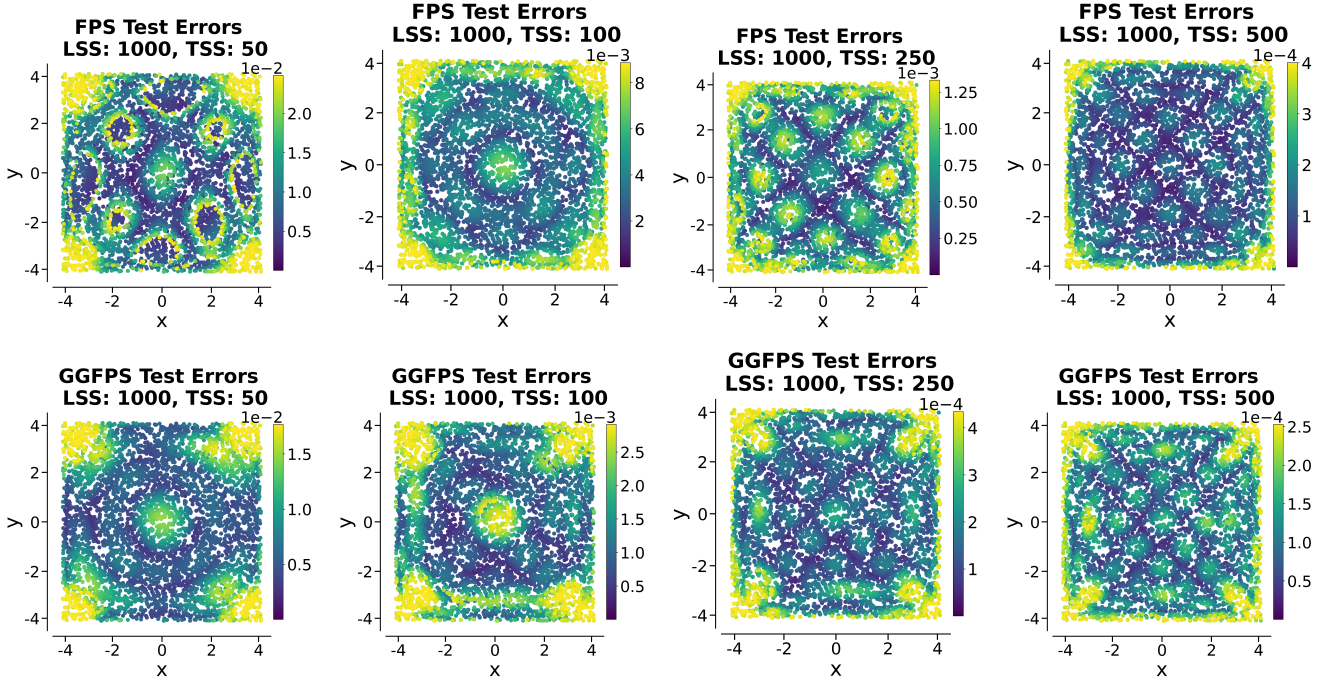


Figure 10. Scatter plots of the 2D ST function test errors corresponding to models trained on FPS (top row) and GGFPS (bottom row), with training set sizes (TSS) ranging from 50 (leftmost column) to 500 (rightmost column). All models start with an initial labeled set of 1000 points (LSS: 1000).

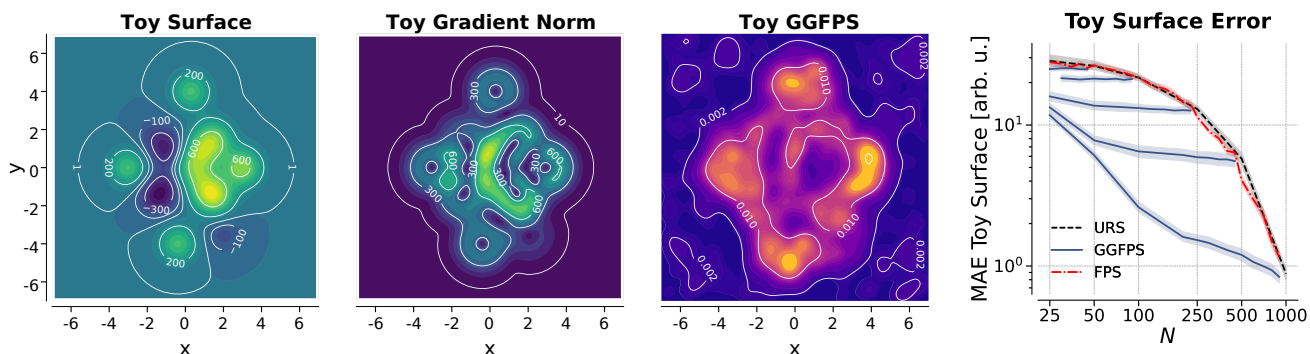


Figure 11. Left: A contour plot of the toy function surface in two dimensions. Center-left: The toy function gradient norm surface. Center-right: a heatmap of 50 toy function training sets generated by GGFPS, each with 100 data points, with a β value of 0.3. In each of the three, a lighter color denotes a higher value. Right: The MAE predictions of the toy function surface with respect to training set size for URS, FPS, and GGFPS (black dashed, red dash-dotted and blue solid lines). The GGFPS β values are optimized per training set fold via grid search.

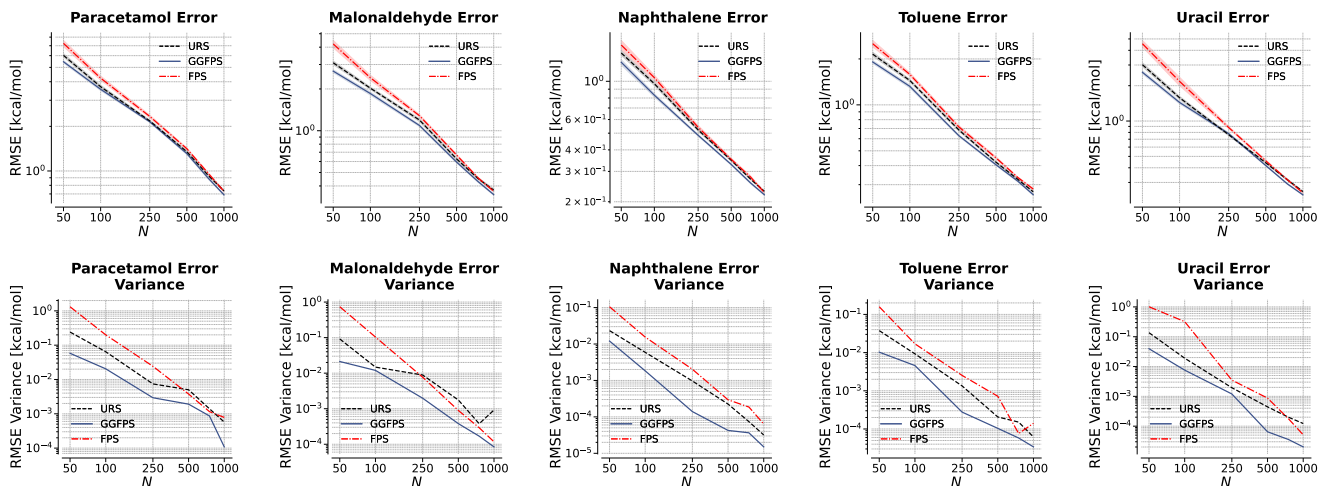


Figure 12. Top row: The RMSE predictions of the MD17 trajectories with respect to training set size, N , for URS, FPS, and GGFPS (black dashed, red dot-dashed, and blue solid lines, respectively). Bottom row: The RMSE prediction variances of the MD17 trajectories with respect to training set size, N , for URS, FPS, and GGFPS (black dashed, red dot-dashed, and blue solid lines).

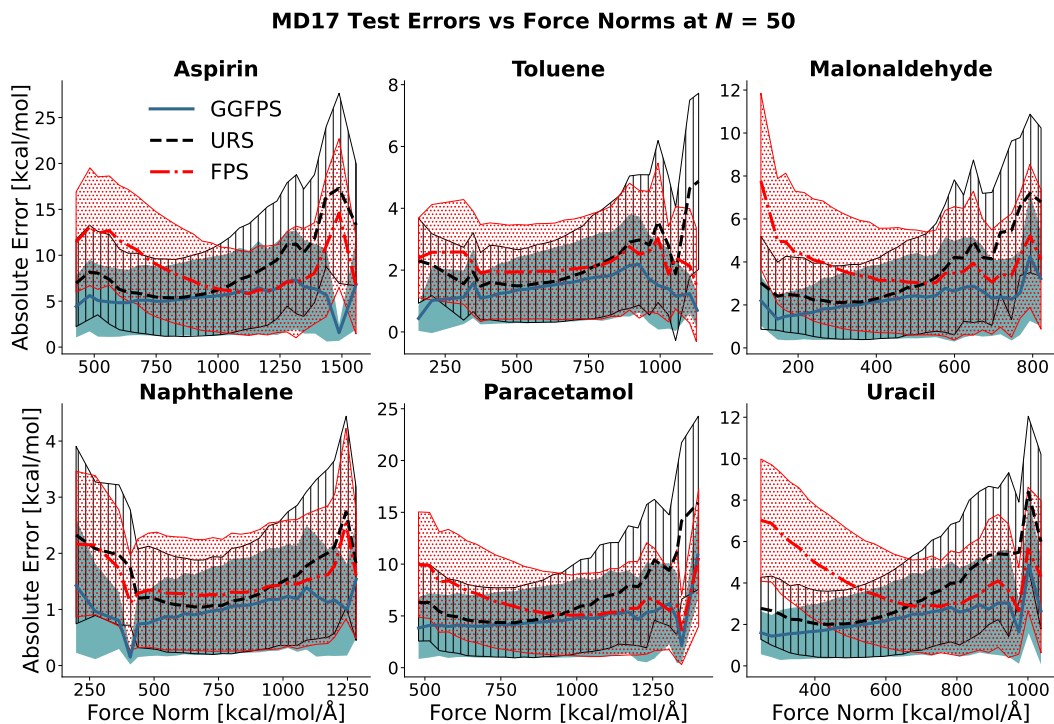


Figure 13. MD17 trajectory test errors versus force norms for FPS, GGFPS, and URS training sets (blue, orange, green), with a training set size (N) of 50 configurations.

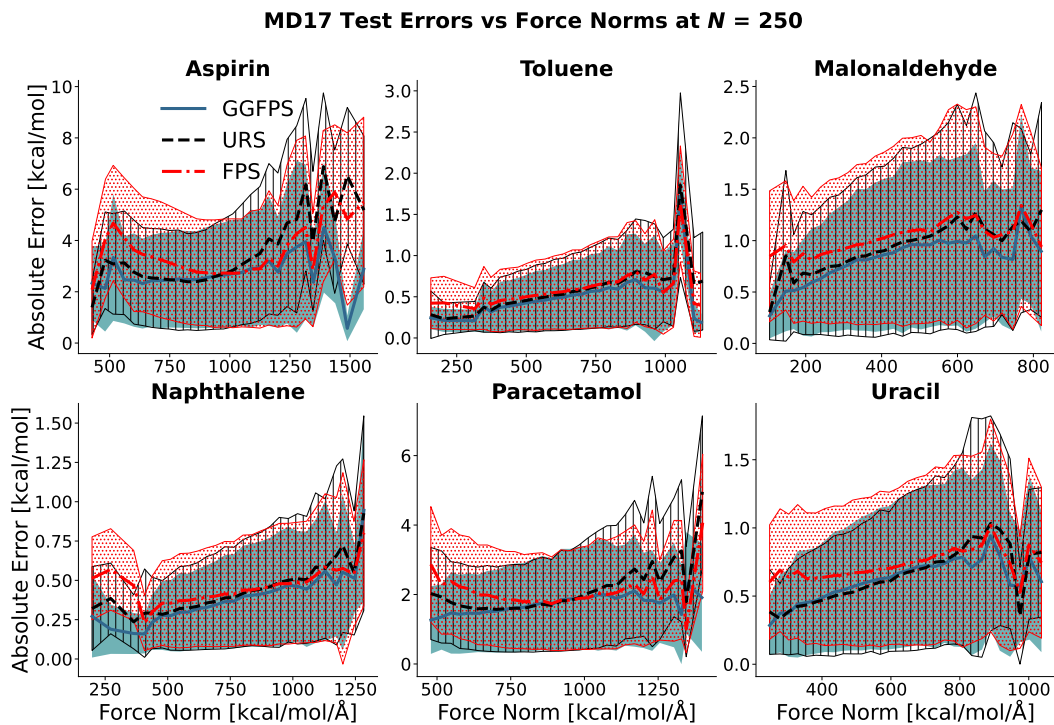


Figure 14. MD17 trajectory test errors versus force norms for FPS, GGFPS, and URS training sets (blue, orange, green), with a training set size (N) of 250 configurations.

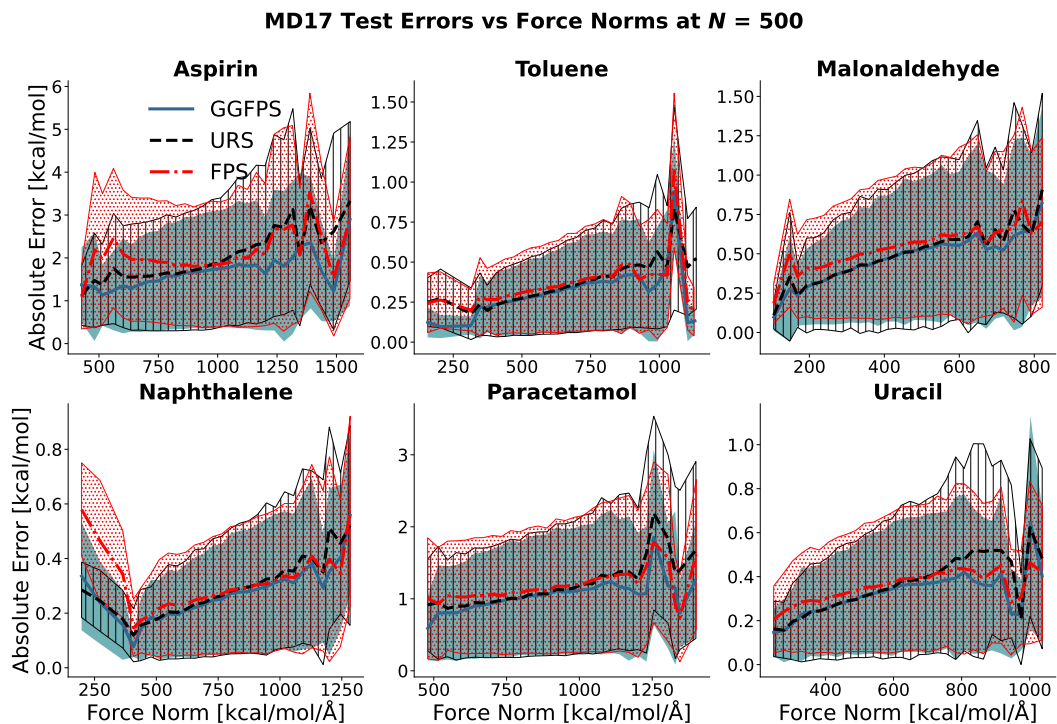


Figure 15. MD17 trajectory test errors versus force norms for FPS, GGFPS, and URS training sets (blue, orange, green), with a training set size (N) of 500 configurations.

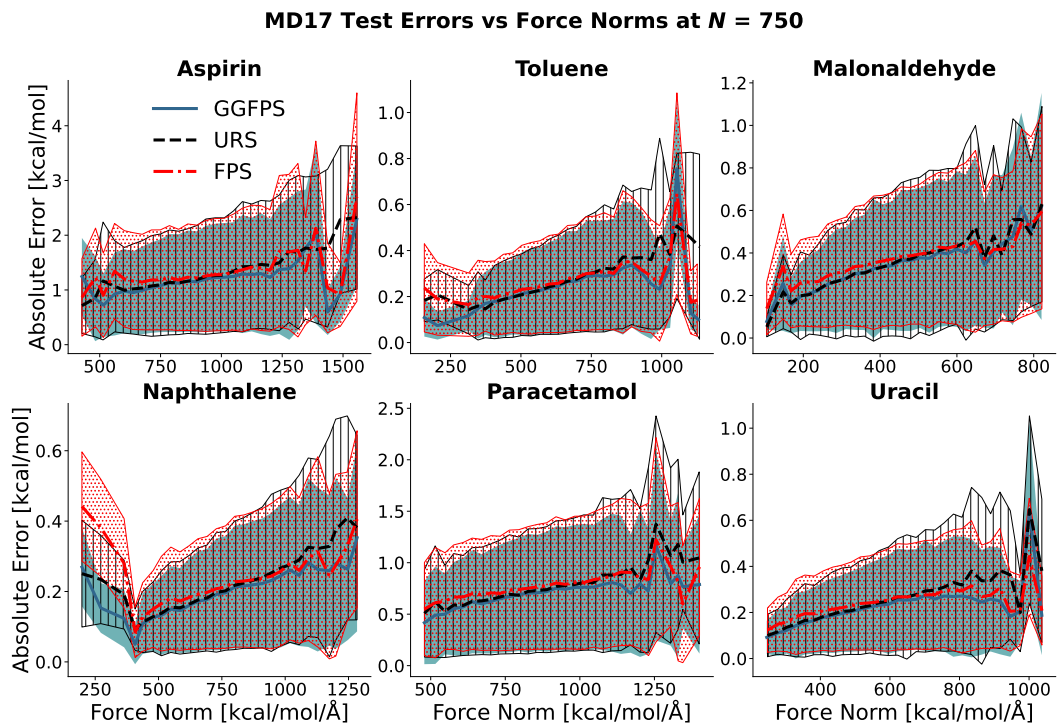


Figure 16. MD17 trajectory test errors versus force norms for FPS, GGFPS, and URS training sets (blue, orange, green), with a training set size (N) of 750 configurations.

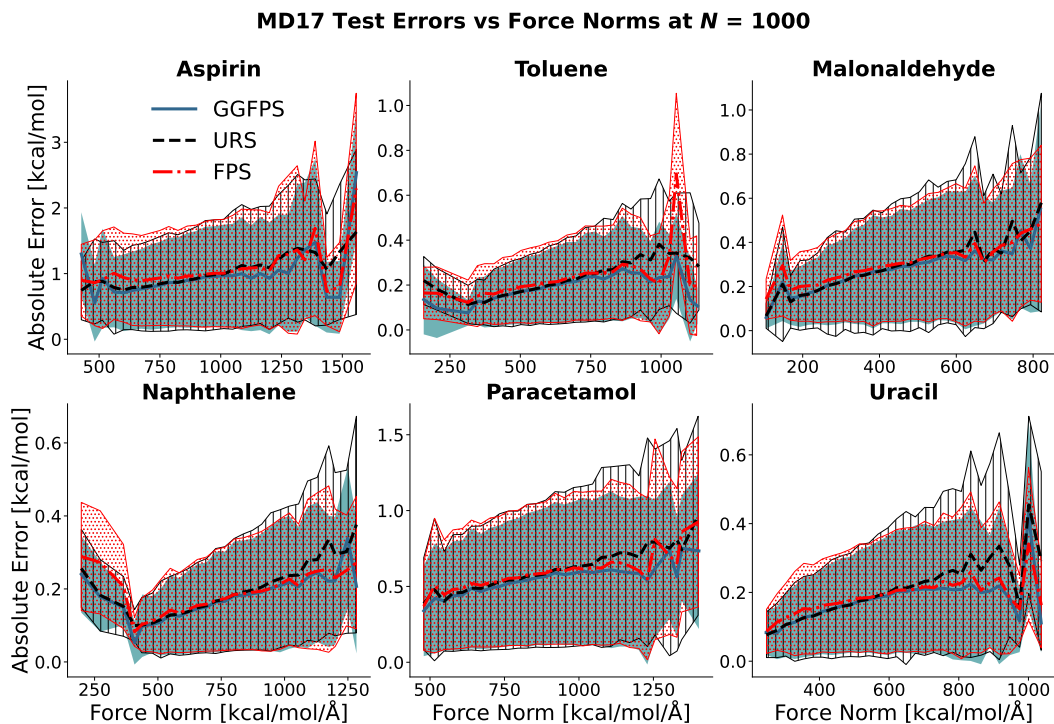


Figure 17. MD17 trajectory test errors versus force norms for FPS, GGFPS, and URS training sets (blue, orange, green), with a training set size (N) of 1,000 configurations.

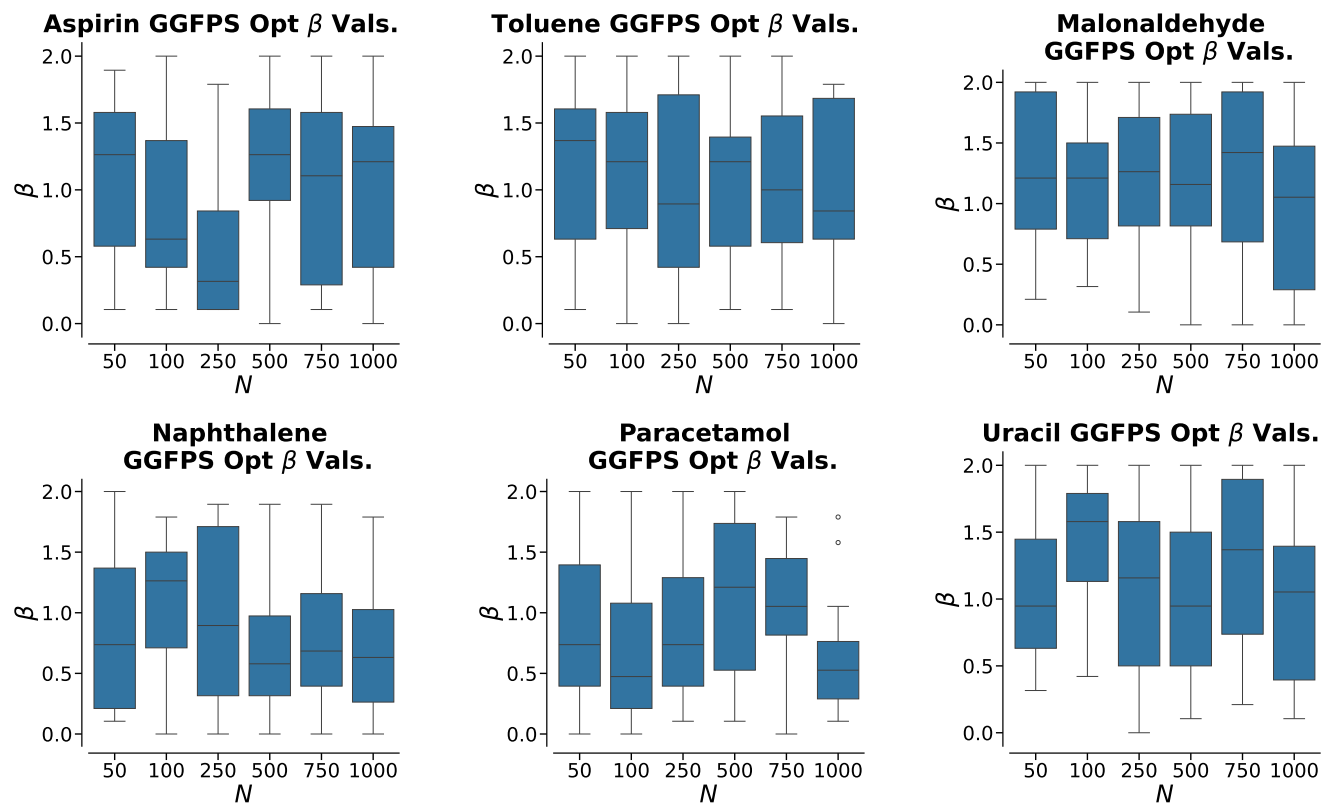


Figure 18. MD17 FCHL19 optimal cross-validated GGFPS β distributions w.r.t. training set sizes.