Supporting Information for A nearsighted force-training approach to systematically generate training data for the machine learning of large atomic structures

Cheng Zeng, Xi Chen, Andrew A. Peterson

Contents

1 Force locality in finite-ranged ML potentials

Based on the definition of atomic energy in finite-ranged MLPs (assume a cutoff of R_c), we can write the atomic energy of atom i as a function of the positions of itself and its neighboring atoms.

$$E_i = E\left(\{\vec{R}_{ij}\}\right), \text{ where } |R_{ij}| < R_c \tag{1}$$

Where $\vec{R}_{ij}$ is the relative position vector between atom i and j. It is obvious that the atomic energy of atom i will not be affected by atoms outside the cutoff sphere. In other words, the locality of atomic energy is R_c. Next we shall show that the locality of atomic forces are different from the locality of atomic energy, or

more precisely the force locality is twice the atomic energy locality. Based on the locality of atomic energy, the force acting on atom i easily follows

$$f_i = -\frac{\partial E}{\partial R_i} = -\sum_{j}^{N} \frac{\partial E_j}{\partial R_i} = -\sum_{j}^{R_{ij}<R_c} \frac{\partial E_j}{\partial R_i} = -\sum_{j}^{R_{ij}<R_c} \frac{\partial E\left(\{R_{jm}\}\right)}{\partial R_i}, \text{ where } R_{jm} < R_c \tag{2}$$

According to Equation 2, the movement of atom m at a distance slightly less than $2R_c$ to atom i can affect the force of atom i via changing the atomic energy of atom j that sits at a distance slightly less than R_c to atom i. Therefore the force locality is $2R_c$.

2 Machine learning models

Behler-Parrinello neural network. In this work, we employed an artificial neural network architecture to fit the PES, as proposed by Behler and Parrinello [1]. We used AMP for ML potential fitting, which is an open-source code developed in our group [2]. We used n2p2 for fast force calls, which is a ready-to-use software for neural network potentials originally developed by Andreas Singraber [3]. A schematic representation of the neural network architecture is shown in Fig. S1. The atomic contribution depends on the atom's local chemical environment. Atoms of the same element type share a neural network structure. In order to preserve the translational and rotational invariance for the total energy with respect to atomic positions, the input atomic positions are transformed to features with Gaussian symmetry functions (SFs). The Gaussian SFs are basically grouped into two categories, namely G2 and G4, representing interactions of atom N with neighboring single atoms and pairs of atoms, respectively. For more details of the descriptors, readers should refer to the work by Khorshidi et al [2]. Thus, each atom 'N' or 'M' is described by SFs 'G_N' or 'G_M', which is an input vector sent to the atomic NN that contains hidden layers and a scalar output. The scalar output is linearly combined to yield the atomic energy which can be summed to obtain the total energy.

For DFT-based machine learning models, we used 20 Gaussian-type SFs selected out of 78 candidates to construct the feature vector of platinum atom. The Gaussian descriptor consists of 11 G2 SFs and 9 G4. The initial and chosen parameters for G2 and G4 SFs are presented in Table S1 and Table S2.

Table S1: The initial η, R_s, ζ and γ parameters for G2 and G4 SFs. R_s in unit Å controls the position of Gaussian in G2 SFs. For G2 SFs, we started with 20 η values ranging from 0.001 and 100, which are evenly spaced on a log scale.

G2			
R_s [Å]	0	2	4
η	0.001	$\underbrace{\cdots}_{\text{18 numbers}}$	100
G4			
η	0.005	0.1	1
ζ	1	4	16
γ	-1	1	

A cosine cutoff function with a cutoff radius of 6.5 Å was used to describe local chemical environments. A simple neural network with a structure of (20, 5, 5, 1) was employed to mitigate overfitting; that is, 20 SFs and two hidden layers of five nodes. A L_2 regularization with the L_2 term being 0.001 was applied to avoid large atomic neural network weights, and hence to reduce overfitting.

Nearsighted force training. The fitting of neural network is by minimizing a loss function. It uses the sum of square residuals as the loss function. The electronic structure calculations provide the energy and forces for the reference database. The ML potential gives the ML energy directly and the ML predicted forces are computed by taking the analytical derivative of ML energy with respect to the atomic positions. For the purpose of nearsighted force training (NFT), we modified the original loss function so that only forces on the central atoms of atomic "chunks" are included in the loss function, see Equation 3.

Table S2: The chosen 11 G2 and 9 G4 SFs, and corresponding η, R_s, ζ and γ parameters.

G2		G4		
R_s [Å]	η	η	ζ	γ
0	0.42813323987193913	0.005	1	-1
0	2.636650898730358	0.005	1	1
0	16.23776739188721	0.005	4	-1
2	0.42813323987193913	0.005	4	1
2	4.832930238571752	0.005	16	-1
2	54.555947811685144	1	1	-1
2	100	1	1	1
4	0.001	1	4	-1
4	29.763514416313193	1	4	1
4	54.555947811685144			
4	100			

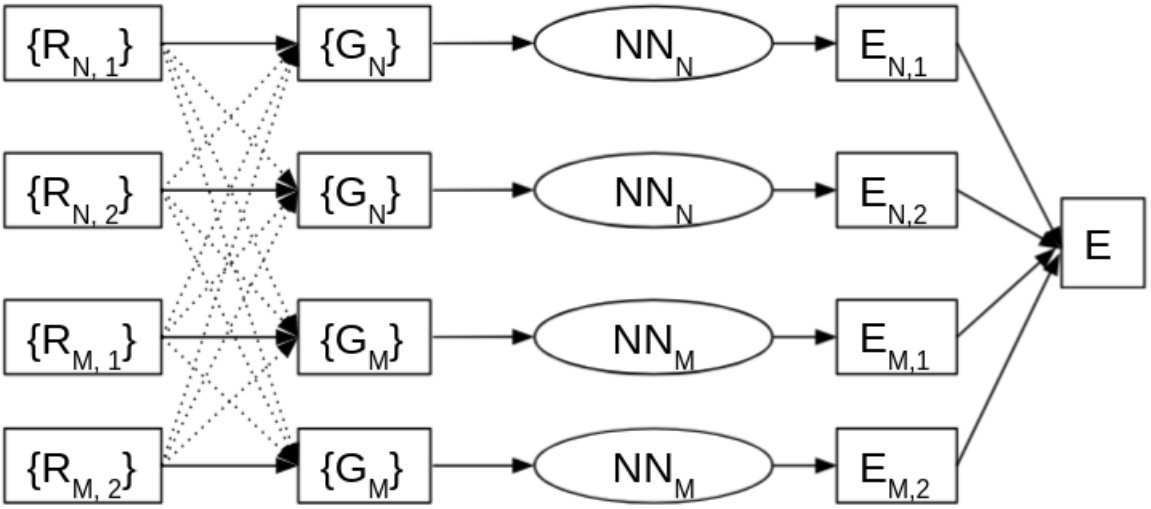

Figure S1: Schematic Representation of Generalized Neural Network for approximating high-dimensional potential energies.

$$\text{Loss} = \frac{1}{2}\sum_{j=1}^{M}\left\{\left(E_j/N_j - \widehat{E}_j/N_j\right)^2 + \frac{\alpha}{3N_j}\sum_{k=1}^{3}\sum_{i=1}^{N_j}\left(F_{ik} - \widehat{F}_{ik}\right)^2\right\} + \frac{\alpha}{6}\sum_{p=1}^{R}\sum_{k=1}^{3}\left(F_{pk} - \widehat{F}_{pk}\right)^2 \tag{3}$$

where E_j and $\widehat{E}_j$ are energies provided by electronic structure calculations and NN, respectively. F_{ik} and $\widehat{F}_{ik}$ are forces on atom i in the direction k by respective electronic structure calculations and NN. N_j is the number of atoms. α is a constant with defines the weight of forces in the loss function. M is the number of training structures (bulk cells in this work) for which both energy and forces are trained, whereas R is the number of atomic "chunks" for which only forces on the central atom are trained. α is the coefficient which controls the path of convergence while training energy and force together. A coefficient of 0.04 was used to add force residuals. Training of neural network models is terminated if either the number of optimization steps exceeds the number of epochs allowed or the convergence criteria for energy and forces are satisfied. We set the number of epochs as 3000, which is typically large enough for ML models to reach a plateau on the loss function surface. Each model was fitted until the RMSE of energy per atom is less than 0.001 eV and RMSE of force is below 0.005 eV/$\mathring{A}$. To ensure that the model will properly fit forces of new images, we also set a force maximum residual tolerance of 0.02 eV/$\mathring{A}$ for any individual training structure.

Parent data. Atomic structures were created in Atomic Simulation Environment (ASE) [4]. The initial training structures for DFT-based ML potential contains a selection of 20 platinum bulk cells, including 4 tetragonal cells with 2 atoms, 1 fcc cell with 4 atoms, 7 repeated tetragonal cells with 16 atoms, and 8 repeated fcc cells with 32 atoms. The trajectory of all bulk structures is included as a supporting information file in the extended xyz format.

Bootstrap ensemble. In this work, we use a 10-member ensemble ML model. The ensemble average is used as the ensemble prediction for forces. The ensemble standard deviation is formulated to estimate the uncertainty of force predictions. If a large halfspread is seen, the ensemble model is more likely to yield a large prediction error and *vice versa*. We sampled training images using a bootstrap technique. At each NFT iteration, we add new atomic "chunks" to the training set. We introduced new images using a accelerated resampling technique. For more details about the bootstrap sampling technique, readers should refer to the work by Peterson et al. [5]

3 Retraining procedure for structure optimization

Since the ensemble model merely trained on atomic "chunks" of Pt_{260} encountered highly uncertain structures during relaxation, a large deviation of the relative energy profile was observed between DFT and the ensemble model. We need to address the uncertain structures by adding relevant uncertain structures into the fitting database. It is arguable that the model can falsely enter a PES region where the uncertain structures will by no means be seen during relaxation. These structures, potentially with highest uncertainty, should not be included as it can lead to data pollution. In view of this concern, we chose a fragment of the ML predicted relaxation trajectory on which the uncertainty is reasonably high, for example between a threshold and one and a half of the threshold. In this work, we set the threshold as the value of the uncertainty for the initial structure, which is 0.26 eV/Å. Then only configuration whose uncertainty falls in the range between 0.26 and 0.39 eV/Å is selected. Next, we extracted atomic "chunks" from the selected structures whose uncertainty fall within the range. In this work, 604 atomic "chunks" were selected. Then we selected 10 atomic "chunks" from the 604 pieces by using CUR at the feature space spanned by features vectors of central atoms of the 604 atomic "chunks". Please refer to the 'CUR approximation' section (4.2) for more details about the CUR algorithm. As defined in Equation 8, the relative norm deviation is 0.0167% between the reduced feature matrix and original feature matrix, which indicates the selected pieces largely represent the uniqueness of the 604 pieces in the feature space. Adding the 10 atomic chunks to the fitting database, we performed a NFT on the initial structure of Pt_{260}. This completes one retraining iteration. The model was retrained for another two iterations until the maximum structure uncertainty has not been improved for two consecutive iterations.

4 Image and feature selection algorithms

Firstly of all, we summarize the situations where image and feature was used in this study. We selected the initial training set out of bulk cells sampled following the "Initialization" procedure in the main text, by using the furthest point sampling algorithm. With the selected initial images, CUR approximation or force correlation method was employed to select SFs out of a candidate SF set. In this study, CUR approximation and force correlation method were used for feature selection on initial structures calculated by EMT and DFT, respectively. The algorithm chosen for feature selection was based on the relative norm difference; see Equation 8 in the CUR approximation section. We set up a ensemble neural network model with the selected initial images and SFs. Depending on the simulation task, the model may encounter a great number of uncertain structures. For example, we selected uncertain atomic "chunks" encountered during the relaxation using the procedure detailed in the Section 3. It is arguable that CUR approximation and furthest point sampling can be used interchangeably to achieve similar regression accuracy as both of them are designed to select diverse images/fingerprints.

4.1 Farthest point sampling

Farthest point sampling was used for initial image selection because of its high interpretability and computational efficiency. Farthest point sampling is based on the idea of repeatedly placing the next sample point in the middle of the least-known area of the sampling domain. It aims to select images that are as diverse as possible. The basic idea is to find the remaining point, whose minimum distance to the selected points is the maximum among all remaining points (see Fig. S2); that is, at every iteration we select one point k which satisfies Equation 4.

$$k = \text{argmax}_j(\min_i |X_i - X_j|) \tag{4}$$

where X is given by a user-defined distance metric. We employed a two-level FPS to select diverse and representative structures. The first level is the total number of atoms. The second level is the local chemical environments characterized by the default SFs in AMP [2].

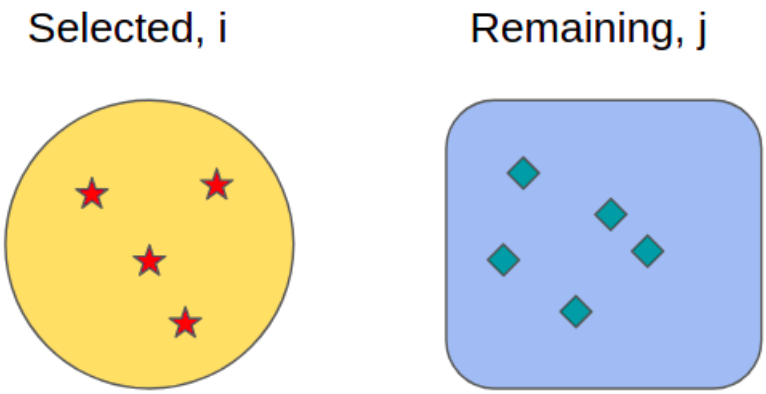

Figure S2: Schematic illustration for Farthest Point Sampling algorithm

4.2 CUR approximation

CUR approximation was used for image/feature selection because the work by Imbalzano *et al.*indicates that SFs selected by CUR approximation give the best accuracy for ML potentials using atomic neural network. [6] The CUR approximation is a dimensional reduction approach, which is particularly useful for improved data analysis [7]. (It is named after the three matrices that are multiplied in this method: C being made from the columns of the original matrix, R from the rows, and U is the transformed matrix.) It inherits the core of principal component analysis (PCA). The difference lies in that PCA generates projected directions that hold the largest variance of the features, whereas CUR uses the original vectors in the feature matrix. We can think of it as selecting as-received vectors that are close to principal components. CUR approximation is based on singular value decomposition (singular value decomposition, or SVD for short, is a factorization of a matrix based on an eigenvalue analysis.) of a feature matrix A, as shown in Fig. S3. The jth column of the feature matrix A can be expressed as a linear combination of left columns u^ξ.

$$A^j = \sum_{\xi=1}^{r} (\sigma_\xi u^\xi) v_j^\xi \tag{5}$$

where $r = \text{rank}(A)$ and u^ξ is the left column, σ_ξ is the eigenvalue and v_j^ξ is the j^{th} row and ξ^{th} column element in the right matrix. Since the eigenvalues σ decrease in the diagonal direction. We can simplify the expression by truncating the sum to the first k terms, as $A^j \approx \sum_{\xi=1}^{k} (\sigma_\xi u^\xi) v_j^\xi$. In this way, the importance score of the the column j is given by

$$\pi_j = \frac{1}{k} \sum_{\xi=1}^{k} \left(v_j^\xi\right)^2 \tag{6}$$

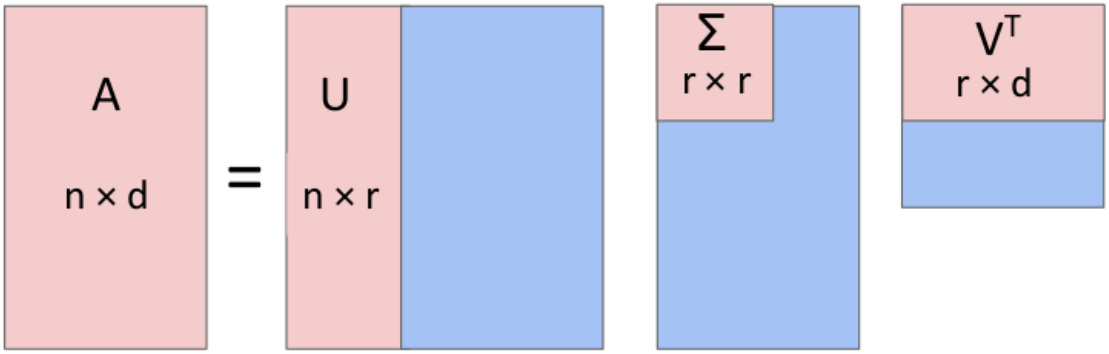

Figure S3: Components of the singular value decomposition of a matrix A.

Thus we can employ a probabilistic criterion for feature selection. At each step, we pick the column which gives the highest score. In the next step, in order to avoid select highly correlated features, every remaining column X_j is then orthogonalized relative to the selected column X_l.

$$X_j \to X_j - X_l(X_l \cdot X_j)/|X_l|^2 \tag{7}$$

In this way, we can select a reduced feature matrix. The accuracy of the selection can be evaluated by the relative norm difference between the reduced feature matrix and original feature matrix.

$$\epsilon = ||X - CUR||_F/||X||_F \tag{8}$$

where X is the original feature matrix, C and R are actual row and column matrix from X, and U can be computed using pseudo-inverse by $U = C^+AR^+$.

4.3 Force correlation method

The CUR approximation approach does not consider the output space, which is especially helpful if information of output is unavailable. However, it can also be problematic as the mapping between input and output is non-trivial and unknown. Proposing a strategy to relate the output (energy and forces) to the input (features transformed from atomic positions) is necessary to understand the effect of features on the target variables. We should note that for atomic neural network as shown in Fig. S1, each atom has its own sub neural network; hence it can have its unique feature vectors. As it is non-trivial to obtain the atomic energy, we used atomic forces to construct a mapping between input and output. That is why this approach is termed "*force correlation method*".

Each atom has its own feature vector and force (magnitude). Therefore, for atoms of the same element type, we can build a mapping from input atomic feature vector to the output atomic force vector, see Fig. S4. We select features which has the highest absolute correlation with the force vector, then we perform orthogonalization on the remaining vectors, as detailed in the CUR approximation section.

5 Computational settings of DFT calculations

All DFT reference calculations were performed using the Grid-based projector-augmented wave code (GPAW). [8] A plane wave basis set was adopted along with the Perdew-Burke-Ernzerhof (PBE) exchange-correlation functional. [9] The plane wave cutoff was set as 450 eV. A k-point density of at least $30/l$ was used for bulk structure calculations, where l is the length of corresponding dimension. Cluster calculations sampled the Brillouin-zone at the Γ-point. All PBE calculations were spin-paired. A Fermi-Dirac smearing of 0.1 eV was used for convergence and results were extrapolated to 0 K. The line search algorithm BFGS was employed for all structure optimizations until the maximum force was less than 0.05 eV/Å. For self-consistent field (SCF) calculations, convergence was achieved until the energy difference between the last two steps is less

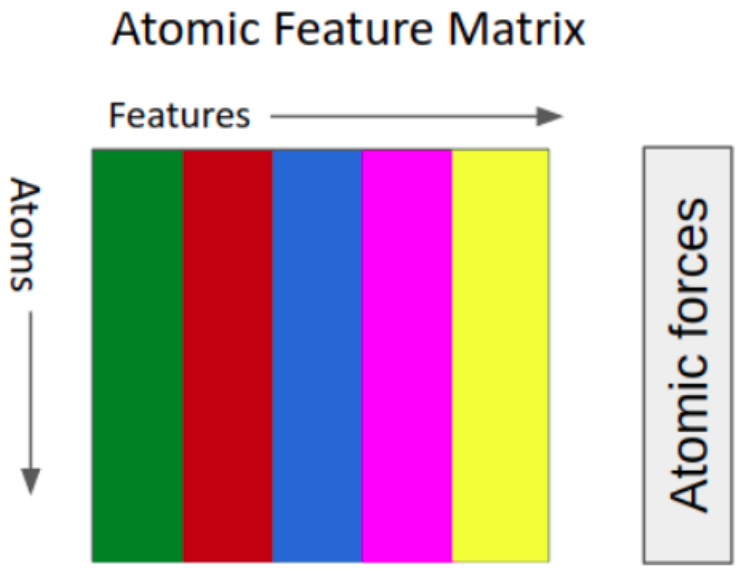

Figure S4: Atomic feature matrix and its mapping to forces.

than 0.0001 eV. For DFT force locality analysis, we employed an additional force convergence criterion for SCF; that is, the force difference between the last two steps is less than 0.01 eV/$\mathring{A}$.

6 Supplemental results

6.1 ML forces *versus* DFT true forces with the initial model

Figure S5 shows the ML-predicted forces against the true DFT forces. The MAEs were computed by taking the average of absolute force vector differences. It shows that ML-predicted forces fitted the true forces almost equally well as the DFT local forces.

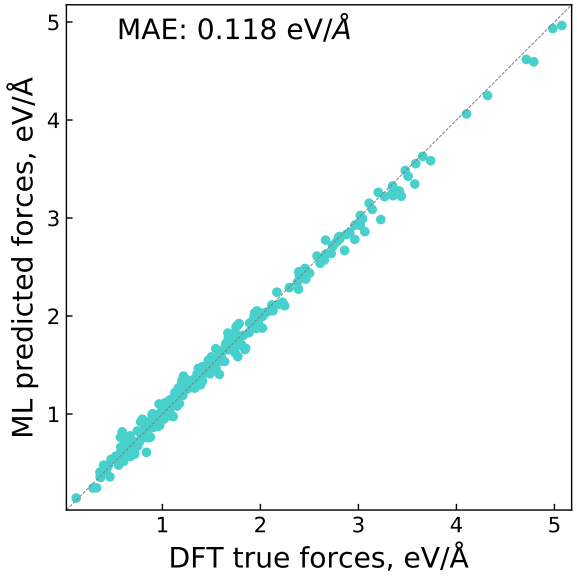

Figure S5: ML predicted forces versus DFT true forces using the initial ensemble model.

6.2 ML forces *versus* DFT forces with the retrained model

With the retrained ensemble, Figure S6 compares the ML-predicted forces with both DFT local and true forces. The training and test images in the comparison with DFT local forces are the same with the previous ones; that is, only atomic chunks from the initial structure of Pt_{260} are included in this comparison for DFT local forces. It shows that adding uncertain atomic chunks chosen from the relaxation trajectory barely changes the fitting of the initial structure.

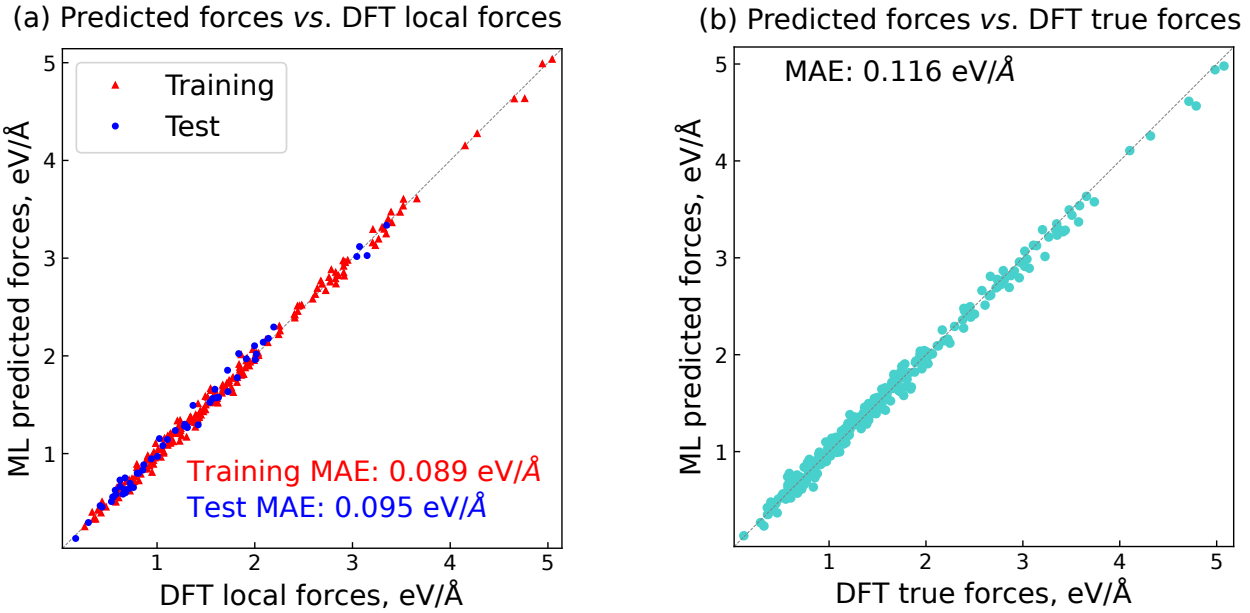

Figure S6: ML predicted forces versus DFT forces using the retrained ensemble model. (a) ML forces *versus* DFT local forces. (b) ML forces *versus* DFT true forces.

7 Application with a nearsighted parent calculator (EMT)

7.1 EMT force locality

Centering on one atom of a rattled Pt_{260}, we extracted atomic "chunks" with cutoffs ranging from 3.5 Å to 8.5 Å. The EMT force versus cutoff relation is shown in the Figure S7(a). It is clear the local forces converge at the cutoff of 6.5 Å, indicating that EMT is local. As a relative larger cutoff can offer a more desired fitting, we used 6.5 Å for both the ML models and extracting atomic chunks during nearsighted force training.

7.2 Nearsighted force training

We sampled 33 bulk cells. The 33 initial bulk cells are made up of 9 tetragonal cells with 2 atoms, 4 fcc cell with 4 atoms, 10 repeated tetragonal cells with 16 atoms, and 10 repeated fcc cells with 32 atoms. Next, we selected 20 bulk cells comprising of 6 tetragonal cells with 2 atoms, 1 fcc cell with 4 atoms, 5 repeated tetragonal cells with 16 atoms, and 8 repeated fcc cells with 32 atoms.

We trained a 10-member ensemble model with the selected initial bulk structures. 21 SFs selected out of 78 candidates were used for featurization, and a neural network structure of (21, 10, 10, 1) was used for the machine learning regression. The other neural network and loss function parameters are the same with values used for DFT-based reference data. We targeted on the same rattled Pt_{260} nanoparticle. As shown in Figure S7(b), the uncertainty goes down gradually and converges at the 10th NFT iteration where the ensemble gives a structure uncertainty of 0.019 eV/Å. Till this step, the ensemble model has seen the forces of 144 atoms. We evaluated the prediction errors on the 144 training and 116 remaining test atomic

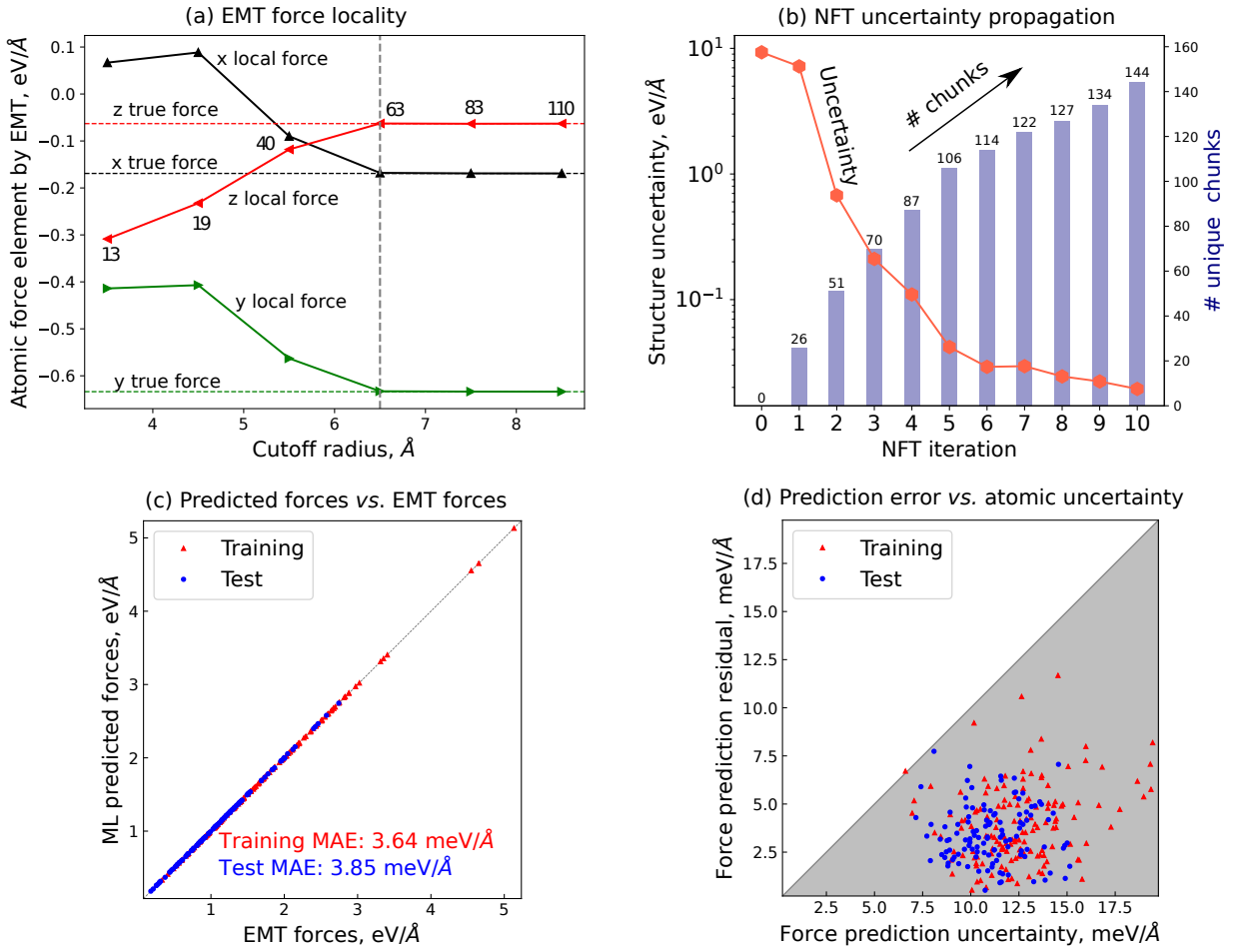

Figure S7: Nearsighted force training on Pt_{260} for EMT-calculated reference data. (a) EMT force locality analysis. The number of atoms included in the cutoff sphere is labeled in the "z local force" curve. (b) Uncertainty propagation and the cumulative number of unique atomic "chunks" at each NFT iteration. The iteration step with the lowest uncertainty is marked with a star and selected as the model for further analyses. (c) ML predicted forces *versus* EMT forces. (d) Force prediction residuals *versus* Force prediction uncertainty as defined. Force prediction residual is defined by the magnitude of force vector difference between ML predicted forces and EMT forces.

"chunks". The training and test mean absolute errors (MAE) of forces are 3.64 and 3.85 meV/Å for respective training and test atomic "chunks"; see Figure S7(c). It suggests that the ML-predicted forces matches almost perfectly well with the actual forces calculated by EMT. We examined the uncertainty metric by analyzing the relationship between force prediction residuals (force vector differences) and the atomic uncertainties. Nearly all points fall below the parity line, which implies that the as-defined atomic uncertainties can be regarded as a reliable upper bound for the force prediction residual.

7.3 Structure optimization

We examined the trained ensemble model by performing a structural relaxation on Pt_{260}. We compared the relative energy profiles by EMT and the ensemble model; see Figure S8. The relative energy change *versus* trajectory step is almost identical between the ensemble model and EMT, except that EMT takes one more step. We evaluated the ML predicted relaxed structure with EMT. The maximum atomic force is 0.048 eVÅ, further confirming the excellent agreement between ML and EMT calculators. The ML predictions were also validated by the uncertainty variation during relaxation, see Fig. S8(b). Structure uncertainties

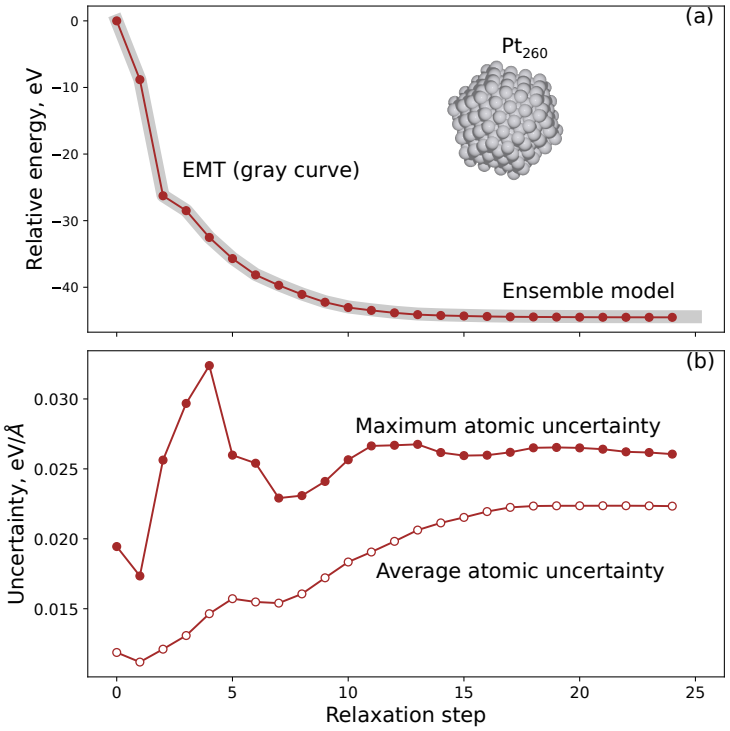

Figure S8: Structure optimization of Pt_{260} nanoparticle by using the ensemble model and EMT. (a) Relative energy *versus* relaxation step curves. (b) Uncertainty propagation during relaxation.

during relaxation are all lower than 0.035 eV/Å, indicating a high prediction confidence at all trajectory steps. Considering that the ensemble model has never seen the entire nanoparticle and any other structures during relaxation, the prediction accuracy achieved by training only uncertain atomic "chunks" of the initial structure is surprisingly satisfactory. Note that if uncertainties ramps up at a certain relaxation step, it indicates that an unknown structure, or more specifically some unknown atoms, are encountered. We should perform restraining to improve the model, as demonstrated for structure optimization with DFT. An effective retraining procedure has been detailed in Section 3.

7.3.1 Transferability to larger systems

We also performed a structure optimization on the same rattled nanoparticle Pt_{1415}. The relaxation trajectories of the ensemble model looks no different compared to EMT; see Fig. S9(a). Besides, the ML predicted last-step structure has a maximum EMT force of 0.049 eVÅ, which confirms that the ML relaxed structure is almost identical to the one predicted by EMT. The structure uncertainty profile shows that for the first 20 steps of relaxation, the machine learning model frequently enters less confident regions. However, the model can still guide the relaxation to arrive at a relaxed structure with high confidence, whose structure uncertainty is as low as 0.023 eV/$\AA$. This demonstrates that the finite ranged ML potentials can give an almost exact fitting for forces calculated by a nearsighted calculator, such as EMT, as long as the atomic chunks are able to fill the force locality of the parent calculator.

8 Example scripts

Example scripts are provided to replicate the results in this work. These scripts were verified to run with:

1. Amp commit fe6d3a182eeeab2117780a00c61a5129a7d9fe8a (January 20, 2022)
2. ASE version 3.21.0b1 (January 8, 2021)

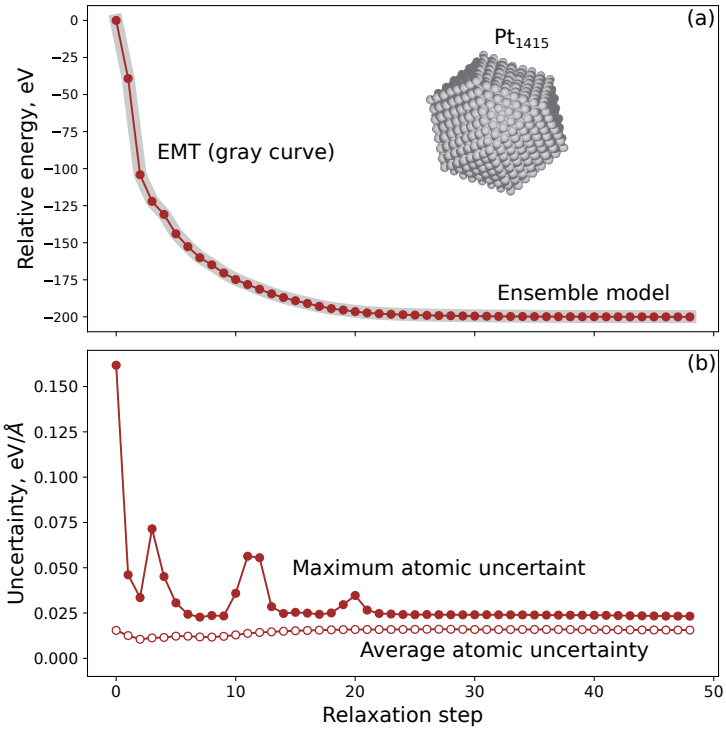

Figure S9: Structure optimization of Pt_{1415} nanoparticle with ensemble model trained on Pt_{260}. (a) Relative energy *versus* relaxation step. (b) Structure uncertainty propagation during relaxation.

3. GPAW version 20.10.1b1 (January 12, 2021)

4. n2p2 commit b45aafc12292c43ea4f575379cb9fcaf5cf20a1d (May 26, 2021)

8.1 Initialization

This example script creates the initial training set following the algorithm detailed in the main text. Code is attached here .

```
#!/usr/bin/env python3
from amp.nft.initialization import Initialization
from amp.utilities import Logger
from ase.calculators.emt import EMT

### Initialization for the EMT parent calculator
Initialization(elements=['Pt'], trajfile='initial.traj', a0=4.0,
               sgs=[225, 79], minsep=2.65, posamp=0.03,
               parent_calc=EMT(),
               log=Logger('initialization.log', overwrite=False),)


### Initialization using GPAW
# from gpaw import GPAW, FermiDirac, PW, Mixer
# def get_calc(text=None, atoms=None):
#     calc = GPAW(txt=text,
#                 mode=PW(450),
#                 xc='PBE',
```

```
#                  occupations=FermiDirac(0.1),
#                  spinpol=False,
#                  maxiter=333,
#                  mixer=Mixer(0.02, 5, 100),
#                  convergence={'energy': 1e-4})
#     return calc
# Initialization(elements=elements, trajfile='initial.traj', a0=4.0,
#                sgs=[225, 79], minsep=2.65, posamp=0.03,
#                parent_calc=get_calc(),
#                log=Logger('initialization.log'),)
```

8.2 Image selection using furthest point sampling

This example script selects 20 images by using a two-level furthest point sampling algorithm. The first level is the number of atoms, and the second level is the feature vectors. The feature space is encoded by the default Gaussian symmetry functions as implemented in AMP [2], along with a cutoff radius of 3.5Å. The selected and remaining images are saved to trajectory files named 'chosen.traj' and 'unchosen.traj', respectively. Code is attached here  .

```
#!/usr/bin/env python3
from amp.preprocess.image_selection import FurthestPointSampling
from amp.utilities import hash_images

images = 'initial.traj'
images = hash_images(images)
fps = FurthestPointSampling(images, k=20, encoder='gaussian',
                                  log='image_selection.log')
# calculate_dev indicates whether the relative norm deviation
# between the original and selected feature matrix is calculated.
chosen_images = fps.search(calculate_dev=True, save_traj='chosen.traj')
```

8.3 Feature selection

This example script selects 21 (or 20) features out of 78 candidates, using the force correlation method (or CUR approximation). The selected symmetry functions are saved to a 'json' file named 'Gs.json'. Code is attached here  .

```
#!/usr/bin/env python3
from amp.preprocess.feature_selection import FTSel
from amp.utilities import Logger
import numpy as np

traj_chosen = 'chosen.traj'

Gs_params = {"G2": {'eta': np.logspace(-3, 2, num=20),
                    'offsets': [0, 2, 4]},
             "G4": {'eta': [0.005, 0.1, 1.],
                    'zeta': [1., 4., 16.],
                    'gamma':[+1, -1]}}

params = dict(Gs=Gs_params, Rc=6.5,
              cutoff='cosine')
encoder = {'descriptor': 'gaussian',
           'params': params}
```

```
### select 21 images using the force correlation method
ftsel = FTSel(traj_chosen, k={'Pt': 21}, method='fcorr', encoder=encoder,
              log=Logger('feature_selection.log', overwrite=True),
              save_json='Gs.json' )

### select 20 images using the CUR approximation
# ftsel = FTSel(traj_chosen, k={'Pt': 20}, method='cur', encoder=encoder,
#               log=Logger('feature_selection.log', overwrite=True),
#               save_json='Gs.json' )

chosen_SFs = ftsel.search(calculate_dev=True)
```

8.4 Active learning

This example script demonstrates the active learning scheme based on the nearsighted force training approach. We assume that we have initial training images saved in 'chosen.traj', the selected symmetry functions saved in 'Gs.json', and the target image is a Pt_{260} nanoparticle defined in the trajectory file 'pt260.traj'. Running this script until the job is terminated will result in bootstrap calculators giving the best result saved as 'best.al.ensemble'. The termination criteria used in this work is detailed in the script. Atomic uncertainties, indices of chunks in the target structure, and atomic 'chunks' at each NFT iteration will be saved in a folder named "saved-info". Code is attached here  .

```
#!/usr/bin/env python3
#SBATCH --time=48:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --partition=batch
#SBATCH --mem=40g
from amp.nft.activelearner import NFT
from amp.utilities import Logger

#### Nearsighted force-training active learning
d = dict(json_file='Gs.json')

calc_text = """
import json
from amp import Amp
from amp.model import LossFunction
from amp.descriptor.cutoffs import Cosine
from amp.descriptor.gaussian import Gaussian
from amp.model.neuralnetwork import NeuralNetwork

with open('../../{json_file}', 'r') as f:
    Gs = json.load(f)
hl = [5, 5]

calc = Amp(model=NeuralNetwork(hiddenlayers=hl),
           descriptor=Gaussian(Gs=Gs, cutoff=Cosine(6.5)),
           dblabel='../../amp-data',
           cores=24)
calc.model.lossfunction = LossFunction(convergence={{'energy_rmse': 0.001,
                                                     'force_rmse':0.005,
                                                     'force_maxresid': 0.02}},
```

```
                                                    overfit=0.001, maxiter=3000)
calc.model.lossfunction.parameters['weight_duplicates'] = False
""".format(**d)

################################################################################
## Termination criteria---job terminated if either one is satisfied.
# - structure uncertainty lower than 0.10
# - Number of NFT iterations exceeds 20
# - structure uncertainty does not go down in two consecutive NFT iterations
################################################################################
# 'threshold' controls the number of atomic chunks to be sent to the parent calculator.
label = 'al'
al = NFT(stop_delta=0.10, max_iterations=20, steps_not_improved=2,
         log=Logger(f'{label}.log', overwrite=False),
         threshold=-0.8,)

traj = 'chosen.traj'
target_image = 'pt260.traj'
start_command = 'sbatch run.py'

### GPAW calculator for atomic chunks.
from gpaw import GPAW, FermiDirac, PW, Mixer
def get_calc(text=None, atoms=None):
    calc = GPAW(txt=text,
                mode=PW(450),
                xc='PBE',
                kpts = (1, 1, 1),
                occupations=FermiDirac(0.1),
                spinpol=False,
                mixer=Mixer(0.02, 5, 100),
                maxiter=333,
                convergence={'energy': 1e-4},
                symmetry={'point_group': False})
    return calc
calc = get_calc()


headerlines = """#SBATCH --account=default
#SBATCH --time=48:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=24
#SBATCH --partition=batch
#SBATCH --mem=60g
"""

al.run(images=traj,
       target_image=target_image,
       n=10,
       calc_text=calc_text,
       start_command=start_command,
       label=label,
       parent_calc=calc,
       headerlines=headerlines,
       dft_cores=24,
```

```
        dft_memory='60G',
        cutoff=8.0,
        expired=360.,
        )
```

References

[1] J. Behler and M. Parrinello, "Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces," *Physical Review Letters*, vol. 98, p. 146401, Apr. 2007.

[2] A. Khorshidi and A. A. Peterson, "Amp: A modular approach to machine learning in atomistic simulations," *Computer Physics Communications*, vol. 207, pp. 310–324, Oct. 2016.

[3] A. Singraber, J. Behler, and C. Dellago, "Library-Based LAMMPS Implementation of High-Dimensional Neural Network Potentials," *Journal of Chemical Theory and Computation*, vol. 15, pp. 1827–1840, Mar. 2019.

[4] A. Hjorth Larsen, J. Jørgen Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. Bjerre Jensen, J. Kermode, J. R. Kitchin, E. Leonhard Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. Bergmann Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment—a Python library for working with atoms," *Journal of Physics: Condensed Matter*, vol. 29, p. 273002, jul 2017.

[5] A. A. Peterson, R. Christensen, and A. Khorshidi, "Addressing uncertainty in atomistic machine learning," *Physical Chemistry Chemical Physics*, vol. 19, pp. 10978–10985, May 2017.

[6] G. Imbalzano, A. Anelli, D. Giofré, S. Klees, J. Behler, and M. Ceriotti, "Automatic selection of atomic fingerprints and reference configurations for machine-learning potentials," *The Journal of Chemical Physics*, vol. 148, p. 241730, Apr. 2018.

[7] M. W. Mahoney and P. Drineas, "CUR matrix decompositions for improved data analysis," *Proceedings of the National Academy of Sciences*, vol. 106, pp. 697–702, Jan. 2009.

[8] J. Enkovaara, C. Rostgaard, J. J. Mortensen, J. Chen, M. Dułak, L. Ferrighi, J. Gavnholt, C. Glinsvad, V. Haikola, H. A. Hansen, H. H. Kristoffersen, M. Kuisma, A. H. Larsen, L. Lehtovaara, M. Ljungberg, O. Lopez-Acevedo, P. G. Moses, J. Ojanen, T. Olsen, V. Petzold, N. A. Romero, J. Stausholm-Møller, M. Strange, G. A. Tritsaris, M. Vanin, M. Walter, B. Hammer, H. Häkkinen, G. K. H. Madsen, R. M. Nieminen, J. K. Nørskov, M. Puska, T. T. Rantala, J. Schiøtz, K. S. Thygesen, and K. W. Jacobsen, "Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method," *Journal of Physics: Condensed Matter*, vol. 22, p. 253202, June 2010.

[9] J. P. Perdew, K. Burke, and M. Ernzerhof, "Generalized Gradient Approximation Made Simple," *Physical Review Letters*, vol. 77, pp. 3865–3868, Oct. 1996.