ALKPU: an active learning method for the DeePMD model with Kalman filter

Haibo Li[a], Xingxing Wu[b], Liping Liu[c,f], Lin-Wang Wang[c,f], Long Wang[d], Guangming Tan[e,f], Weile Jia[e,f,*]

[a] *School of Mathematics and Statistics, The University of Melbourne, Melbourne, VIC, 3010, Australia*
[b] *Technology Department, Longxun Quantum Inc., Beijing, 100192, China*
[c] *Institute of Semiconductors, Chinese Academy of Sciences, Beijing, 100083, China*
[d] *Computing System Optimization Lab, Huawei Technologies Co., Ltd., Beijing, 100094, China*
[e] *State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China*
[f] *University of Chinese Academy of Sciences, Beijing, 101408, China*

Abstract

Neural network force field models such as DeePMD have enabled highly efficient large-scale molecular dynamics simulations with *ab initio* accuracy. However, building such models heavily depends on the training data obtained by costly electronic structure calculations, thereby it is crucial to carefully select and label the most representative configurations during model training to improve both extrapolation capability and training efficiency. To address this challenge, based on the Kalman filter theory we propose the *Kalman Prediction Uncertainty (KPU)* to quantify uncertainty of the model's prediction. With KPU we design the *Active Learning by KPU (ALKPU)* method, which can efficiently select representative configurations that should be labelled during model training. We prove that ALKPU locally leads to the fastest reduction of model's uncertainty, which reveals its rationality as a general active learning method. We test the ALKPU method using various physical system simulations and demonstrate that it can efficiently coverage the system's configuration space. Our work demonstrates the benefits of ALKPU as a novel active learning method, enhancing training efficiency and reducing computational resource demands.

Keywords: molecular dynamics, neural network force field, active learning, Kalman filter, uncertainty quantification, Kalman prediction uncertainty

1. Introduction

The *ab initio* molecular dynamics (AIMD) is a powerful computational tool that allows the simulation of dynamical processes involving atoms and molecules to gain a detailed understanding of physical and chemical properties of complex systems with quantum mechanics precision, such as chemical bonding and formation of reaction intermediates for metal alloys, ceramics and polymers [47, 62, 39, 53, 2]. However, AIMD is computationally expensive due to the need for electronic structure calculations to obtain the energy and forces of the system at each step, which limits the scale of the simulated system and time duration [14, 46, 37]. Recently, the deep learning methods provide tools to combine traditional AIMD with a neural network (NN) to develop neural network force field (NNFF) [12, 11, 6, 10], in which an NN model is designed that maps configurations of the simulated system to their respective potential energies and forces. The well-trained NNFF model can be used to rapidly generate energies and forces to drive molecular dynamics (MD), improving the speed and scale of simulations while maintaining *ab initio* accuracy. This approach has been effectively used for simulating complex condensed-phase systems as well as molecular transport studies [44, 66, 22].

The performance of an NNFF depends heavily on the quality of training data generated by electronic structure calculation softwares such as VASP and PWmat [34, 37, 36], and efficient sampling strategies play a crucial role in

*Corresponding author

Email addresses: haibo.li@unimelb.edu.au (Haibo Li), stars_sparkling@163.com (Xingxing Wu), liuliping@semi.ac.cn (Liping Liu), lwwang@semi.ac.cn (Lin-Wang Wang), wanglong82@huawei.com (Long Wang), tgm@ict.ac.cn (Guangming Tan), jiaweile@ict.ac.cn (Weile Jia)

ensuring that the model is accurate and reliable across the entire range of configurations of interest [67]. Thus, it is imperative to find such a method that selects informative and representative configurations to build a relative small training dataset with which we can train a model with the desired performance [61, 71]. Active learning is a technique that concurrently selects the most informative data samples for labeling during training, which can significantly reduce the labeling effort and save much training time required to reach a given level of performance.

Many NNFF models and packages have been developed, such as HDNNP [7, 6, 9], BIM-NN [65], Schnet [56], Enn-s2s [27], DeepMD-kit [64], SIMPLE-NN [42], DimeNet++ [26, 25], PaiNN [55], SpookyNet [63], CabanaMD-NNP [20], NewtonNet [29] and NequIP [5]. The Deep Potential Molecular Dynamics (DeePMD) is one of the state-of-the-art model among them [69, 64, 70, 38]. It contains an embedding network to learning symmetric invariant features [6, 4] from input configurations and a fitting network to map the feature to potential energies and forces [70]. The model is trained using the Adam optimizer [41] and an active learning method called DP-GEN is designed to select the most relevant portions of the high-dimensional configuration space [72]. DP-GEN has two main parts: exploration and selection. Exploration involves using the temporary trained model to run several steps MD, while selection selects representative configurations from this MD trajectory that should be labeled to enlarge the training dataset. DP-GEN trains four models with different initial NN parameters, then uses the predictions of the trained model to calculate an empirical variance, and selects data points with high variances. Although it has been applied to many physical system simulations, DP-GEN has two main problems. Firstly, it incurs substantial computational expense due to the training of four models. Secondly, there is a consistency-error problem when using multiple models to calculate variance, which means that all four trained models may make consistently incorrect predictions for a given configuration [68], potentially leading to unnecessary redundant or missed acquisitions of configurations that the model has not covered.

In this paper, based on the RLEKF optimizer [33] that exploits the extended Kalman filter (EKF) [40, 50, 31, 30, 15] to train DeePMD, we develop a new active learning method. By taking advantages of the Kalman filter's capabilities of estimating the probability distribution of NN parameters, we propose a metric to quantify uncertainty of the model's prediction called the Kalman Prediction Uncertainty (KPU), which is an approximate variance of the model's prediction. In the framework of the RLEKF optimizer, we propose an efficient algorithm to calculate KPU, then with KPU we design the Active Learning by KPU (ALKPU) method for DeePMD. After the exploration step, ALKPU computes the KPU for each configuration in the new MD trajectory, and configurations with higher KPUs are selected for labeling and then added to the training dataset concurrently during model training. Comparing with DP-GEN's four-model method, ALKPU is more appropriate to quantify uncertainty of the model's prediction and trains only one model, thus can save large computational resource overheads while lead to a better selection of representative data points. The main contributions of this work are summarized in the following.

1.1. Contributions

- Based on the EKF for training NN, we propose the KPU to quantify uncertainty of the NN's predictions. In the framework of the RLEKF optimizer for training DeePMD, we design an efficient algorithm to calculate the KPU of forces predicted by the temporary trained model.
- We propose the ALKPU active learning method for DeePMD and develop a sampling platform that can concurrently exploring, selecting and labeling representative configurations during training. Several typical physical system are used to test the ALKPU method and show its advantages as a new active learning method.
- We establish some theoretical results about ALKPU with mathematical proofs by using the Fisher information matrix and Shannon entropy. The results show that ALKPU is essentially a locally fastest reduction procedure of NN model's uncertainty, which reveals the rationality of ALKPU as a general active learning method.

1.2. Related work

Active learning for neural networks. Active learning is a widely studied topic, with early work found in [57]. The center of active learning is the query strategy that decides which data points are informative and representative that should be selected and labelled [3]. DP-GEN belongs to the "query by committee" family where a variety of models are trained on the current labeled data and then votes on the output for unlabeled data and labels those points for which the "committee" disagrees the most [59]. ALKPU belongs to the "uncertainty sampling" family that labels those points for which the current model is the least certain about regarding the correct output [21]. Other query strategies include the variance reduction that labels points that would minimize output variance [45, 16, 17], the expected error reduction that label points that would mostly reduce the model's generalization error [54] and the expected model change that labels points that would mostly change the current model [58]. Statistical learning methods such as the Gaussian regression process and Bayesian neural network can give better uncertainty measures [52, 19], but it is hard to directly applied them to deep neural networks. Recent research [23, 43, 24] has demonstrated connections between the dropout based active learning and approximate Bayesian inference, making it possible to apply Bayesian active learning methods to deep learning.

Active learning in MD simulations. Active learning is particularly useful for data-extensive scenarios such as chemical-space exploration in MD simulations. One effective algorithm proposed in [13] to actively learn energies and forces for MD of bulk aluminium and aluminium surfaces employs Euclidean distance between configurations as a selection criterion. The "query by committee" was applied first for NNFFs in [8] where a committee of NN models with different architectures but comparable average performances is selected. In contrast to NNs, in [60] the prediction variance of Gaussian regression process was used to learn complex chemical reaction mechanisms. This approach was further extended to enable the generation of multiple labels simultaneously, providing a significant acceleration of matter simulations [49]. For the linearly parametrized interatomic potential [48], an active learning approach using the D-optimality criterion for selecting configurations was designed. Additionally, active learning approaches have been applied to accelerate chemical-space exploration based on the Thompson sampling [32] and to speed up exploration of potential energy surfaces by synthesizing representations of unseen configurations [28].

2. The DeePMD model and RLEKF optimizer

The state-of-the-art NNFF used in this work is the DeePMD model [69, 70]. The two main parts of DeePMD are the embedding net and fitting net. For each atom i in a given configuration, the physical symmetries such as translational, rotational, and permutational invariance are integrated into the descriptor $\mathcal{D}_i$ through a fully connected network named embedding net. Then the descriptor passes through a fully connected network named fitting net to approximate local atomic energy. The network structure can be described as follows.

(1). For each configuration of the given system consisting Cartesian coordinates $\mathbf{r}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ for $i = 1, \ldots, N_a$, the input of DeePMD is the neighbor lists $\mathcal{R}_i = \{\mathbf{r}_{ij} = \mathbf{r}_j - \mathbf{r}_i : |\mathbf{r}_{ij}| < r_c\}$ for all i, where r_c is the cutoff radius. Then $\mathcal{R}_i$ is translated into the smooth version $\tilde{\mathcal{R}}_i \in \mathbb{R}^{N_m \times 4}$ with $(\tilde{\mathcal{R}}_i)_j = s(|\mathbf{r}_{ij}|)(1, \mathbf{r}_{ij}/|\mathbf{r}_{ij}|) \in \mathbb{R}^4$, where N_m is the maximum length of all neighbor lists and j is the neighbor index of atom i. The smoothing function $s(r)$ satisfies $s(r) = 1/r$ for $x < r_{cs}$ and $s(r) = 0$ for $x > r_c$, and it decaying smoothly between the two thresholds.

(2). Define the mapping of the embedding network as $\mathcal{G} = \mathcal{E}_2 \circ \mathcal{E}_1 \circ \mathcal{E}_0$, where $\mathcal{E}_0(\boldsymbol{x}) = \tanh(\boldsymbol{x} \otimes W_0 + \mathbf{1} \otimes w_0)$ with $W_0 \in \mathbb{R}^{M_1}$ and $\mathcal{E}_l(X) = X + \tanh(XW_l + \mathbf{1} \otimes w_l)$ with $W_l \in \mathbb{R}^{N_m \times M_1}$ for $l = 1, 2$. The notation "$\otimes$" is the cross product between vectors $\mathbf{1} \in \mathbb{R}^{N_m}$ and $w_l \in \mathbb{R}^{M_1}$, and tanh is element-wise. Then we define the local embedding matrix $\mathcal{G}_i \in \mathbb{R}^{N_m \times M_1}$ with its elements $(\mathcal{G}_i)_{jk} = \mathcal{G}(s(|\mathbf{r}_{ij}|))_k$. The descriptor of atom i is defined as $\mathcal{D}_i := \mathcal{G}_{i1}^T \tilde{\mathcal{R}}_i \tilde{\mathcal{R}}_i^T \mathcal{G}_{i2} \in \mathbb{R}^{M_1 \times M_2}$. In practice, we take $\mathcal{G}_{i1} = \mathcal{G}_i$ and take the first M_2 $(< M_1)$ columns of $\mathcal{G}_i$ to form $\mathcal{G}_{i2}$.

(3). The mapping of the fitting network is defined as $\mathcal{F}(\mathcal{D}_i) = \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1 \circ \mathcal{F}_0(\mathcal{D}_i)$ with $\mathcal{D}_i$ reshaped into a vector of form $\mathbb{R}^{M_1 M_2}$, where $\mathcal{F}_0(\boldsymbol{x}) = \tanh\left(\tilde{W}_0 \boldsymbol{x} + \tilde{w}_0\right)$ with $\tilde{w}_0 \in \mathbb{R}^d$, $\tilde{W}_0 \in \mathbb{R}^{d \times M_1 M_2}$, $\mathcal{F}_3(\boldsymbol{x}) = \tilde{W}_3 \boldsymbol{x} + \tilde{w}_3$ with $\tilde{w}_3 \in \mathbb{R}$, $\tilde{W}_3 \in \mathbb{R}^{1 \times d}$ and $\mathcal{F}_l(\boldsymbol{x}) = \boldsymbol{x} + \tanh\left(\tilde{W}_l \boldsymbol{x} + \tilde{w}_l\right)$ with $\tilde{w}_l \in \mathbb{R}^d$, $\tilde{W}_l \in \mathbb{R}^{d \times d}$ for $l = 1, 2$.

(4). The output $E_i = \mathcal{F}(\mathcal{D}_i)$ is the local energy corresponding to atom i. Finally the total energy of this system is $E = \sum_i E_i$ and the force for each atom is $\boldsymbol{F}_i = -\nabla_{\mathbf{r}_i} E$ that can be computed by backward propagation.

The DeePMD-kit software [64] exploits the Adam optimizer [41] to train the NN parameters with the loss function be a weighted sum of energy loss and force loss. In this work, the model is trained by the RLEKF optimizer proposed in [33], which is the abbreviation of the *reorganized layer extended Kalman filter*. The idea of exploiting EKF to train NN is as follows. Let $\boldsymbol{w}$ denotes the vector including all the well-trained parameters. The NN model can be modelled as

$$\begin{cases} \boldsymbol{v}_t = \boldsymbol{v}_{t-1} = \boldsymbol{w}, \\ y_t = h(\boldsymbol{v}_t, x_t) + \eta_t, \end{cases}$$

where $\{(x_t, y_t)\}_{t\in\mathbb{N}}$ are training data pairs, $h(\cdot,\cdot)$ is the nonlinear NN mapping, and η_t is the noise for the NN model. In order to avoid overfitting of earlier data, a fading memory strategy is used, which leads to the following model:

$$\begin{cases} \boldsymbol{v}_t = \lambda_t^{-1/2}\boldsymbol{v}_{t-1}, \quad \boldsymbol{v}_0 \sim \mathcal{N}(\hat{\boldsymbol{w}}_0, P_0), \\ y_t = h(\alpha_t^{-1}\boldsymbol{v}_t, x_t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, R_t), \end{cases} \tag{2.1}$$

where $0 < \lambda_i \leq 1$ are called fading memory factors, $\boldsymbol{v}_0 = \boldsymbol{w}$ and $\mathcal{N}(\hat{\boldsymbol{w}}_0, P_0)$ is the initial distribution of $\boldsymbol{w}$. Using the relation $\boldsymbol{v}_t = \alpha_t\boldsymbol{w}$ with $\alpha_t = \lambda_t^{-1/2}\cdots\lambda_1^{-1/2}$, the t-th trained parameter $\hat{\boldsymbol{w}}_t$ is defined as $\hat{\boldsymbol{w}}_t = \alpha_t^{-1}\hat{\boldsymbol{v}}_t$ with $\hat{\boldsymbol{v}}_t$ computed by EKF. The procedure is summarized in Algorithm 1.

Algorithm 1 EKF with fading memory (EKFfm) for training NN

Input: $\hat{\boldsymbol{w}}_0, P_0, \lambda_1, \nu$

```
for t = 1, 2, ..., T do
    P̄_t = P_{t-1} λ_t^{-1}
    H_t = D_w h(ŵ_{t-1}, x_t)                          ▷ Differential (Jacobian) with respect to w
    A_t = H_t P̄_t H_t^T + α_t^2 R_t,   α_t = λ_t^{-1/2} ··· λ_1^{-1/2}
    K_t = P̄_t H_t^T A_t^{-1}
    P_t = (I − K_t H_t) P̄_t                             ▷ Covariance matrix of estimated v̂_t
    d_t = y_t − h(ŵ_{t-1}, x_t)
    ŵ_t = ŵ_{t-1} + K_t d_t                             ▷ Trained parameter
    λ_{t+1} = λ_t ν + 1 − ν                             ▷ 0 < ν < 1
end for
```

Output: $\hat{\boldsymbol{w}}_T, P_T, \lambda_{T+1}$

The EKFfm is further optimized by elaborately reorganizing the NN layers by splitting big and gathering adjacent small ones to approximate a dense matrix by a sparse diagonal block matrix, and the energy and force are trained alternatively where the prediction for force is transformed to a scalar, which makes A_t a scalar to avoid matrix inversions. After code optimizations, RLEKF converges faster with slightly better accuracy for both force and energy than Adam [33]. The detailed derivation of EKFfm and descriptions of RLEKF are described in Appendix A.

3. Active learning by Kalman Prediction Uncertainty

Active learning minimizes labeling costs while maximizing modeling accuracy. While there are various methods in active learning literature, we propose to label the data point whose model uncertainty is the highest. Thanks to the EKF that estimates the distribution of trained NN parameter, we can estimate the variance around the model's prediction and use it as an uncertainty measure.

3.1. Kalman Prediction Uncertainty

After t iterations of training by EKFfm, we have estimated parameter $\boldsymbol{w}$ with expectation $\hat{\boldsymbol{w}}_t$ and covariance $\hat{P}_t = \alpha_t^{-2}P_t$ due to $\hat{\boldsymbol{w}}_t = \alpha_t^{-1}\hat{\boldsymbol{v}}_t$. If we treat the t-th trained $\boldsymbol{w}$ as Gaussian $\boldsymbol{w} \sim \mathcal{N}(\hat{\boldsymbol{w}}_t, P_t)$ and approximate $h(\boldsymbol{w}, x)$ near $\hat{\boldsymbol{w}}_t$ by the one-order Taylor expansion $h(\boldsymbol{w}, x) \approx h(\hat{\boldsymbol{w}}_t, x) + D_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)$, then $h(\boldsymbol{w}, x)$ approximately obeys the distribution

$$\mathcal{N}(h(\hat{\boldsymbol{w}}_t, x), \mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)\hat{P}_t\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)^T) = \mathcal{N}(h(\hat{\boldsymbol{w}}_t, x), \alpha_t^{-2}\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)P_t\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)^T).$$

By neglecting the scalar factor α_t^{-2} at each iteration, we define the *Kalman Prediction Uncertainty (KPU)* as follows to quantity uncertainty of the model's prediction for an input x.

Definition 3.1 (KPU). *For an NN model trained by EKFfm, define the KPU of an input x with respect to the t-th trained parameters as*

$$\mathrm{KPU}_t(x) := \mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)P_t\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)^T. \tag{3.1}$$

Algorithm 2 KPU calculation in RLEKF (`KPU_cal`)

Input: $\hat{\boldsymbol{w}}_t$, $P_t = \mathrm{diag}(P_{t1}, \ldots, P_{tL})$, input x ▷ P_t is block diagonal
1: $H = \mathrm{D}_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)$ ▷ Backward propagation
2: Split H corresponding to P_t: $\{H_1, \ldots, H_L\} = \mathtt{split}(H)$
3: **for** $l = 1, \ldots, L$ **do** ▷ L is the number of reorganized layers
4: $\quad KPU_l = H_l P_{tl} H_l^T$ ▷ KPU for each reorganized layer
5: **end for**
Output: $\mathrm{KPU}_t(x) = \sum_{l=1}^{L} KPU_l$

Algorithm 3 KPU for atomic force ($\mathtt{KPU_{force}}$)

Input: $\hat{\boldsymbol{w}}_t$, P_t, configuration $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^{N_a}$
1: $E = \sum_i h_E(\hat{\boldsymbol{w}}_t, \mathcal{R}_i)$ ▷ Predicted total energy
2: $\boldsymbol{F}_i = -\nabla_{\mathbf{r}_i} E$ ▷ Predicted atomic force
3: $\{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_n\} = \mathtt{rand\_select}(N_a, \mathtt{frac})$ ▷ Randomly select $n = \lfloor N_a \cdot \mathtt{frac} \rfloor$ atoms
4: **for** $j = 1, \ldots, n$ **do**
5: $\quad$ **for** $k = 1, 2, 3$ **do**
6: $\quad\quad KPU_{F_{jk}} = \mathtt{KPU\_cal}(\hat{\boldsymbol{w}}_t, P_t, \mathcal{R}; F_{jk})$ ▷ KPU for force component F_{jk}
7: $\quad$ **end for**
8: **end for**
Output: $\mathrm{KPU}_{\mathrm{force}} = \sum_{j=1}^{n} \sum_{k=1}^{3} KPU_{F_{jk}}/3n$ ▷ Average of KPU for forces

When the prediction has dimension 1, the KPU becomes $\mathrm{KPU}_t(x) = \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T P_t \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)$ that is a nonnegative number. Note that $\nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x) = \mathrm{D}_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T$. For the RLEKF optimizer, the computed P_t has a block diagonal form corresponding to the reorganized layer strategy, thus KPU can be calculated as Algorithm 2. Since RLEKF is a block diagonal approximation to EKFfm, the computed KPU in RLEKF is an approximation to the real KPU defined in (3.1).

Although KPU for both total energy and atomic forces can be calculated, we find that using KPU for force is generally better in the practical active learning procedure. This is because forces represent local information of atoms which are more sensitive to the changes of configurations, while energy is a global quantity and does not seem to provide sufficient information. Since a physical system involves many atoms, to reduce computational cost and make KPU to be a scalar, a subset of atoms is randomly selected to calculate and sum the KPU for the three components of these atomic forces. Denote by $h_E(\cdot, \cdot)$ the nonlinear mapping from $\mathcal{R}_i$ to the atomic energy E_i, the calculation of KPU for force is described in Algorithm 3. The output $\mathrm{KPU}_{\mathrm{force}}$ is a nonnegative number that can measure uncertainty for the predicted forces.

3.2. ALKPU for DeePMD

During model training, the active learning by KPU (ALKPU) method concurrently selects the next data points with higher KPUs. This gives the following procedure of ALKPU for DeePMD:

1. Exploration. With the initial training dataset, a temporary model is trained by RLEKF. Then several steps of MD is run using the energy and forces generated by this NN model. This procedure is used to explore the high-dimensional configuration space to obtain some configurations not too far away from the real AIMD trajectory.
2. Selection. For the configurations obtained by the exploration procedure, we need to select the most representative ones. The $\mathtt{KPU_{force}}$ of each configuration is computed and then judged whether $\sigma_0 \leq \mathtt{KPU_{force}} \leq \sigma_1$ with $\sigma_0 = c_0 \mathrm{KPU}_{\mathrm{trained}}$ and $\sigma_1 = c_1 \mathrm{KPU}_{\mathrm{trained}}$, where $\mathrm{KPU}_{\mathrm{trained}}$ is the average KPU of some training data at the last training step. The threshold σ_0 is used to select configurations with higher KPUs that needed to be labelled. If the KPU is higher than σ_1, it means that this configuration deviates too far from the real AIMD trajectory and thus it should be discarded.
3. Labelling. For those selected configurations, labelling can be done by using an electronic structure calculation software to compute the corresponding total energy and atomic forces.

4. Training. The above labelled data are added to the previous training dataset. With the enlarged training dataset, the model is trained again. This cycle can be repeated multiple times until a well-trained model with desired performance is obtained.

The workflow of the whole procedure is shown in Figure 1.

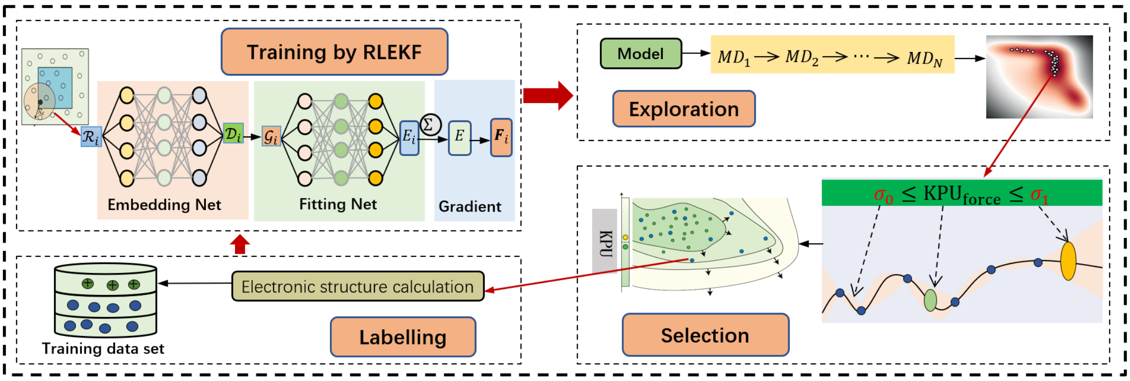


Figure 1: Workflow of ALKPU for DeePMD

4. Theoretical analysis of ALKPU

The ALKPU method is essentially a locally fastest reduction procedure of model's uncertainty. This can be revealed with the help of the Fisher information matrix (FIM) and Shannon entropy. The proofs of all the results are in Appendix C. Suppose the training data $\{(x_t, y_t)\}_{t\in\mathbb{N}}$ satisfies the parametrized statistical model

$$y_t = h(\boldsymbol{w}, x_t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, R_t). \tag{4.1}$$

The following result establishes a correspondence between P_t and FIM. Note that $v^{\otimes 2} := vv^T$ is the cross product for a column vector v.

Theorem 4.1. *For the EKFfm with training data $\{(x_t, y_t)\}_{t\in\mathbb{N}}$, denote by $p(y_t|\boldsymbol{w})$ the probability density function and let $J_0 = \gamma_0 P_0^{-1}$ for any arbitrary $\gamma_0 > 0$. Then at each step, the following relation holds:*

$$J_t = \beta_t J_{t-1} + (1-\beta_t)\mathbb{E}_{y_t\sim p(y_t|\hat{\boldsymbol{w}}_{t-1})}\left[\nabla_{\boldsymbol{w}} \ln p(y_t|\hat{\boldsymbol{w}}_{t-1})^{\otimes 2}\right], \quad J_t = \gamma_t P_t^{-1}, \tag{4.2}$$

where

$$\beta_t = \frac{\lambda_t}{\lambda_t + \alpha_t^{-2}\gamma_{t-1}}, \tag{4.3}$$

and γ_t satisfies the recursion

$$\gamma_t = \frac{\gamma_{t-1}}{\lambda_t + \alpha_t^{-2}\gamma_{t-1}}, \quad t \ge 1. \tag{4.4}$$

Proposition 4.1. *For $0 < \nu < 1$, the asymptotic behavior of γ_t and β_t satisfies*

$$\lim_{t\to\infty} \gamma_t = 0, \quad \lim_{t\to\infty} \beta_t = 1. \tag{4.5}$$

For model (4.1), the FIM is symmetric semi-definite matrix defined as

$$J(\boldsymbol{w}) := \mathbb{E}_{y_t\sim p(y_t|\boldsymbol{w})}\left[\nabla_{\boldsymbol{w}} \ln p(y_t|\boldsymbol{w})^{\otimes 2}\right] = -\mathbb{E}_{y_t\sim p(y_t|\boldsymbol{w})}\left[\nabla^2_{\boldsymbol{w}} \ln p(y_t|\boldsymbol{w})\right], \tag{4.6}$$

where $\nabla_{\boldsymbol{w}}^2$ denotes the Hessian with respect to $\boldsymbol{w}$ [1]. Theorem 4.1 implies that J_t is a weighted sum of the FIMs with respect to every estimated parameter $\hat{\boldsymbol{w}}_i$ during training steps $i = 1, \ldots, t$. The FIM can be used to measure the amount of information that the training data carry about the unknown parameter $\boldsymbol{w}$, since the inverse of FIM sets a lower bound on the variance of the model's parameter estimate, which is known as the Cramér-Rao inequality [51, 18]. Relation (4.2) implies that a smaller J_t^{-1} is equivalent to a smaller P_t, meaning a lower variance of the t-th estimated $\boldsymbol{w}$. Note that for a multidimensional parameter $\boldsymbol{w}$, one has to map the FIM to a scalar for comparison.

Suppose we have the t-th trained model, in order to further increase model's extrapolation ability, the next selected data should be less similar as which the t-th model can predict, thereby an active learner should select a new training data that leads to a big reduction of the model's uncertainty. Suppose we have a prior distribution of $\boldsymbol{w}$ and the data (x, y) obeys the parametrized model

$$y = h(\boldsymbol{w}, x) + \eta. \tag{4.7}$$

After receiving the data (x, y), the posterior distribution of $\boldsymbol{w}$ becomes $\mathbb{P}(\boldsymbol{w}|y) = \mathbb{P}(y|\boldsymbol{w})\mathbb{P}(\boldsymbol{w})/\mathbb{P}(y)$ by the Bayes' formula. The reduction of uncertainty of $\boldsymbol{w}$ can be measured by the change of Shannon entropy defined as follows.

Definition 4.1 (Change of entropy)**.** *For the data* (x, y) *obeys the model* (4.7)*, suppose the probability density functions of* $\boldsymbol{w}$ *before and after receiving data* (x, y) *are* $p(\boldsymbol{w})$ *and* $p(\boldsymbol{w}|y)$*. Define the change of entropy of* $\boldsymbol{w}$ *as*

$$S := \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w}|y)}[\ln p(\boldsymbol{w}|y)] - \mathbb{E}_{\boldsymbol{w}\sim p(\boldsymbol{w})}[\ln p(\boldsymbol{w})], \tag{4.8}$$

which is the difference of Shannon entropy between $p(\boldsymbol{w})$ *and* $p(\boldsymbol{w}|y)$*.*

Theorem 4.2. *Suppose the data* (x, y) *obeys the model* (4.7) *with* $\eta \sim \mathcal{N}(0, \kappa^{-1}I)$ *and* $\boldsymbol{w} \sim \mathcal{N}(\hat{\boldsymbol{w}}_t, \hat{P}_t)$ *at the t-th EKFfm iteration. If we use the approximation*

$$\frac{1}{2}\nabla_{\boldsymbol{w}}^2\|y - h(\boldsymbol{w}, x)\|^2 \approx \nabla_{\boldsymbol{w}} h(\boldsymbol{w}, x)\nabla_{\boldsymbol{w}} h(\boldsymbol{w}, x)^T \tag{4.9}$$

by omitting $\partial^2/\partial w_i \partial w_j$ *and higher order derivative terms, then the change of entropy of* $\boldsymbol{w}$ *at the t-th iteration is*

$$S_t = \frac{1}{2}\ln\left(\det\left(I + \kappa\alpha_t^{-2}\mathrm{KPU}_t(x)\right)\right). \tag{4.10}$$

A similar result is proposed in [45], where the approximation (4.9) is also used if (x, y) falls into a flat region around $\boldsymbol{w}$. Note that $S_t = \frac{1}{2}\ln\left(1 + \kappa\alpha_t^{-2}\mathrm{KPU}_t(x)\right)$ when the prediction y has dimension 1. Theorem 4.2 indicates that ALKPU selects the new training data that maximizes the reduction of entropy from the prior to the posterior, thus leads to a locally fastest reduction of model's uncertainty.

For a well-trained model that can represent the data, there is a relation between the KPU and expected loss that corresponds to mean square error (MSE). For simplicity, the result presented below assumes that y has dimension 1.

Theorem 4.3. *Suppose data* (x, y) *obeys the model* (4.7) *with* $\boldsymbol{w} \sim \mathcal{N}(\hat{\boldsymbol{w}}_t, \hat{P}_t)$*. Let* $\hat{y}_t = h(\hat{\boldsymbol{w}}_t, x)$*. If we use the approximation* $h(\boldsymbol{w}, x) \approx h(\hat{\boldsymbol{w}}_t, x) + \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)$*, then*

$$\mathbb{E}_{\boldsymbol{w}}[|y - \hat{y}_t|^2] \geq \alpha_t^{-2}\mathrm{KPU}_t(x). \tag{4.11}$$

The linear approximation of $h(\boldsymbol{w}, x)$ around $\hat{\boldsymbol{w}}_t$ is reasonable given that the model's prediction is fairly good. Theorem 4.3 implies that for the data well covered by the trained model, KPU can be used to predict the MSE loss.

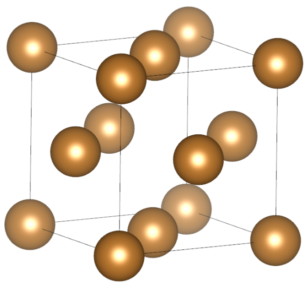
5. Experimental results

In the experiments, the network configuration is $[1, 25, 25, 25]$ for the embedding net and $[400, 50, 50, 50, 1]$ for the fitting net, and the setting of r_c, N_m, M_1, M_2 and $s(\cdot)$ is consistent with DeePMD-kit [64]. For RLEKF, we set $\lambda_1 = 0.98$, $\nu = 0.9987$ and $P_0 = I$, while the initialization $\hat{\boldsymbol{w}}_0$ is consistent with DeePMD-kit and the reorganizing-layer strategy is consistent with [33]. For the KPU calculation procedure $\mathtt{KPU}_{\mathrm{force}}$, we set `frac` = 0.5. The two threshold c_0 and c_1 in ALKPU are set as $c_0 = 1.25$ and $c_1 = 2.0$. The AIMD and electronic structure calculations based on density functional theory (DFT) are implemented using the PWmat software [37, 36].

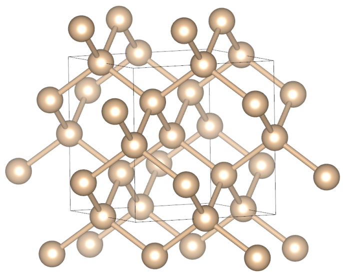
The experimental environment is a high-performance cluster environment, including six computing nodes and one proxy node; see Table 1 for a detailed description. Each computing node has four high-performance graphics cards, including four nodes partitioned as “A100”, each with four high-performance A100 graphics cards where the memory of a single card is 40GB. Two other nodes are partitioned as “3090”, each with four GeForce RTX 3090 graphics cards, where the memory of a single card is 24GB. In the experiments of ALKPU and DP-GEN, the main process runs on the proxy node, responsible for task distribution, result handling, and driving the iterative active learning process. Model training and DFT calculation were run on the computing nodes.

Table 1: Description of the high performance cluster environment

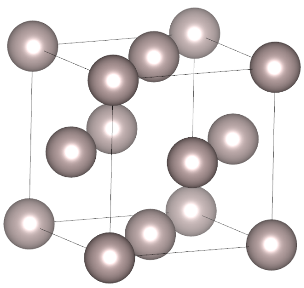
Computing environment	Proxy node	Partition “A100”		Partition “3090”
		Node 1, 2, 3	Node 4	Node 5, 6
CPU	Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz	Intel(R) Xeon(R) Platinum 8375C CPU @ 2.90GHz	Intel(R) Xeon(R) Gold 6248R CPU @ 3.00GHz	Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz
Number of CPU	2	2	2	2
CPU cores	10	32	24	16
Logistical CPU	40	64	48	32
Graphics card	no	GRID A100 PCIe 40GB * 4		GeForce RTX 3090 24G * 4



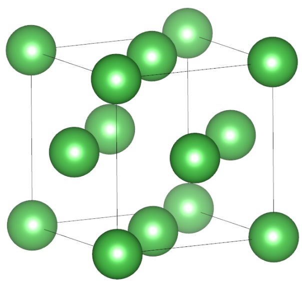
(a) Cu



(b) Si



(c) Al



(d) Ni

Figure 2: Crystal structure of the four test systems: (a). Cu-108, FCC; (b). Si-64, diamond; (c). Al-108, FCC; (d). Ni-108, FCC.

The crystal structure of the four systems used in the tests are shown in Figure 2. The Cu Si, Al, and Ni system includes 108, 64, 108 and 108 atoms, respectively, and the initial training datasets are generated by running 500 AIMD steps/2fs at 300K, 500K, 300K and 500K, respectively.

Now we show the test results for ALKPU and the comparison with DP-GEN on the Cu system. During the exploration step, MD is run with a 2fs time interval using energies and forces predicted by the temporary trained model, and 10% configurations are uniformly chosen from this MD trajectory. Note that the melting point of Cu is 1357.77K, thus a large amount of extra configurations should be select as the temporary increases from 300K to 1400K. Table 2

Table 2: Comparison between ALKPU and DP-GEN for selecting configurations of Cu-108 system

Exploration by MD				ALKPU			DP-GEN		
Round	Temperature	Step	Maximum	Reliable	Selected	Discarded	Reliable	Selected	Discarded
1	500K	30000	100	1710	1289	1	2606	393	1
2	500K	30000	100	2422	577	1	2976	25	0
3	500K	30000	100	2997	2	1	2989	11	0
4	800K	30000	100	2547	452	1	1223	1777	0
5	800K	30000	100	2826	171	3	2779	220	1
6	800K	30000	100	2995	4	1	2885	115	0
7	1200K	30000	100	2686	313	1	1229	1760	11
8	1200K	30000	100	2892	105	3	2386	612	2
9	1200K	30000	100	2995	4	1	2826	173	1
10	1400K	30000	100	2998	2	0	2292	708	0
11	1400K	30000	100	2956	42	2	2770	228	2
12	1400K	30000	100	2990	9	1	2905	95	0

compares the performance of ALKPU and DP-GEN for selecting representative configurations. For the selection step, the parameter `Maximum` is used to randomly select `Maximum` points from the selected ones if the number of selected points exceeds `Maximum`. DP-GEN also has two thresholds (σ_0, σ_1) similar to (c_0, c_1) to judge excessively low or high empirical variances, and here we set them to $(0.02, 0.2)$ as the default setting $(0.05, 0.15)$ can not work well. For both ALKPU and DP-GEN, the number of selected configurations is consistent with the increase of temperature. For example, at the 1-st, 4-th and 7-th round, a large amount of new configuration are selected and labelled, this is because the temporary trained model's prediction accuracy is poor since it does not cover the configuration space at higher temperatures. After completing the 11-th round of active learning, the model gradually covers the configuration space of interest, leading to a decrease in the number of selected configurations.

For the above two active learning procedures, Figure 3 displays the convergence behaviors of ALKPU and DP-GEN on two validation sets as the number of active learning rounds increases. The two validation sets are generated by DFT calculations of 1000 configurations in the AIMD trajectories at 1200K and 1400K, respectively, and the convergence is measured by the average of 1000 root mean square errors (RMSE) for the 1000 configurations. As the active learning proceeds, the error gradually deceases, this is because new configurations are added to enlarge the training dataset and the configuration spaces at 1000K and 1400K are gradually covered by the trained model. For both ALKPU and DP-GEN, the errors of energy and force eventually decrease at the level of *ab initio* accuracy. Note that for DP-GEN, the energy error deceases to the *ab initio* accuracy at the first round, and the decreasing trend is not very obvious afterwards. For a more detailed comparison, Figure 4 displays the RMSE of energy and force predicted by the final trained model for all the 1000 configurations of AIMD trajectories in the 1200K and 1400K validation sets. While both methods eventually achieve *ab initio* accuracy, the ALKPU trained model predicts energy with slightly smaller RMSE than DP-GEN. These comparisons demonstrate that ALKPU can effectively cover the configurations space of interest during model training. Considering that ALKPU only trains one model while DP-GEN trains four models, ALKPU has advantages in terms of reducing training time and computational resource overheads.

Table 3 summarizes the time cost statistics for one step DP-GEN and ALKPU active learning methods for the Cu-108 system. For comparison, the training was run on a single GPU. In ALKPU, the model was trained by RLEKF with a batchsize of 128, while in DP-GEN the model was trained by Adam and the batchsize is one by default of DeePMD-kit. Compared to DP-GEN, ALKPU takes more time to calculate KPU, this is due to the computation of back-propagation to get the gradient of each atomic force. One can randomly select much less atoms to calculating $\mathrm{KPU}_{\mathrm{force}}$ to reduce the computational cost of back-propagation. Additionally, since KPU of different configurations can be computed simultaneously, this task can be divided into multiple tasks, further reducing the time required for this part. As ALKPU only needs to train one model while DP-GEN needs to train four models, the overall computational resource overhead of ALKPU is still less than that of DP-GEN.

Figure 5 displays the convergence behavior of ALKPU on two validation sets of the Si-64 system, where the two

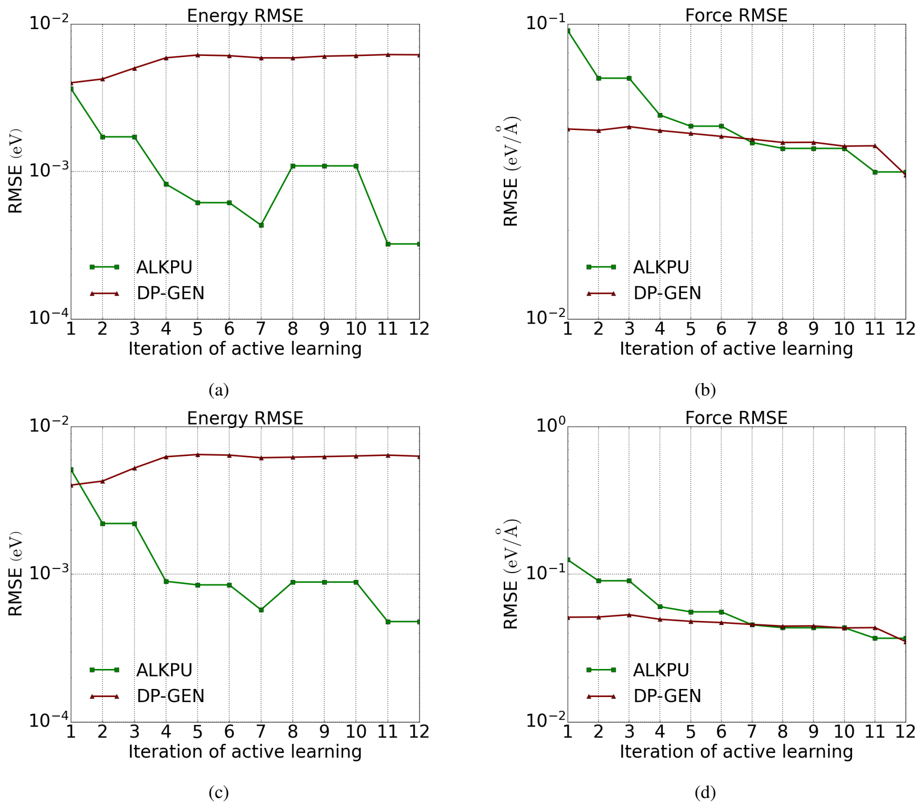
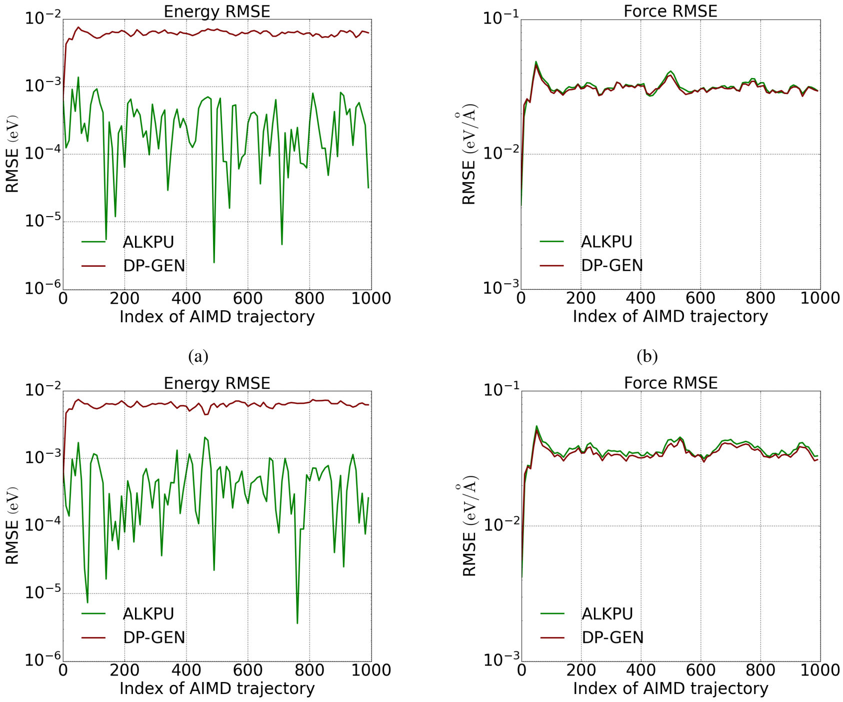


Figure 3: Convergence of predicted energy and force on two validation sets as the active learning proceeds, Cu-108 system: (a). Energy error, 1200K ; (b). Force error, 1200K ; (c). Energy error, 1400K; (d). Force error, 1400K.

Table 3: Time consuming for one step active learning, Cu-108 system.

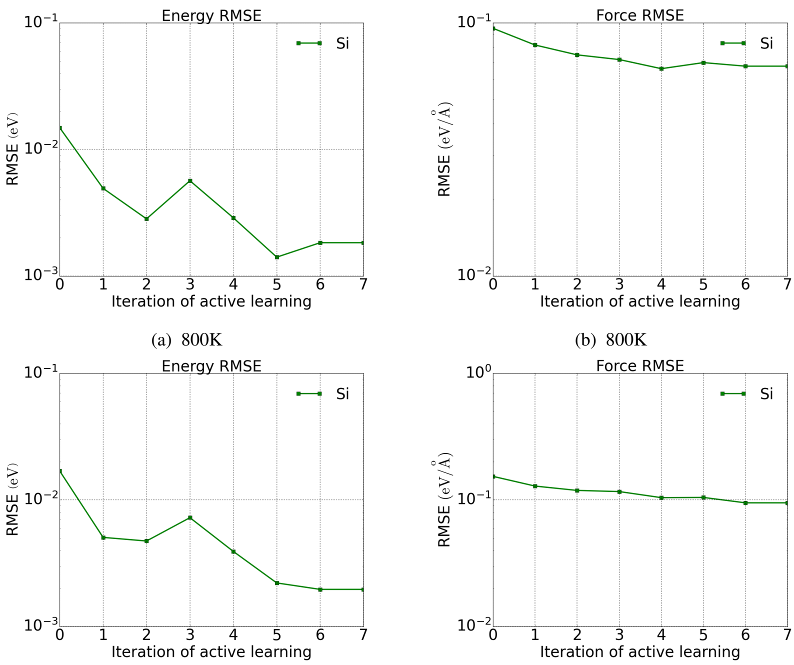
Computation	Time consuming
DP-GEN training	0.66h (1 batchsize, 400 epochs)
ALKPU training	0.34h (128 batchsize, 100 epochs)
KPU computation	0.28h (3000 configurations, 7 concurrent threads)
MD	0.4h (30000 steps/2fs, 32 CPUs)
Labeling	9.2h (single A100 GPU with 4 jobs, k-point (1,1,1))



(a) (b)

(c) (d)

Figure 4: RMSE of energy and force at each configuration in the two validation sets after the active learning has finished, Cu-108 system: (a). Energy RMSE, 1200K ; (b). Force RMSE, 1200K ; (c). Energy RMSE, 1400K; (d). Force RMSE, 1400K.



(a) 800K (b) 800K

(c) 1100K (d) 1100K

Figure 5: Convergence of predicted energy and force on two validation sets of as the active learning proceeds, Si-64 system: (a). Energy error, 800K; (b). Force error, 800K; (c). Energy error, 1100K; (d). Force error, 1100K.

validation sets are generated by DFT calculations of 1000 configurations in the AIMD trajectories at 800K and 1100K, respectively, and the convergence is measured by the average of 1000 root mean square errors (RMSE) for the 1000 configurations. The convergence behavior for Al-108 and Ni-108 are similar, and thus we do not display them anymore.

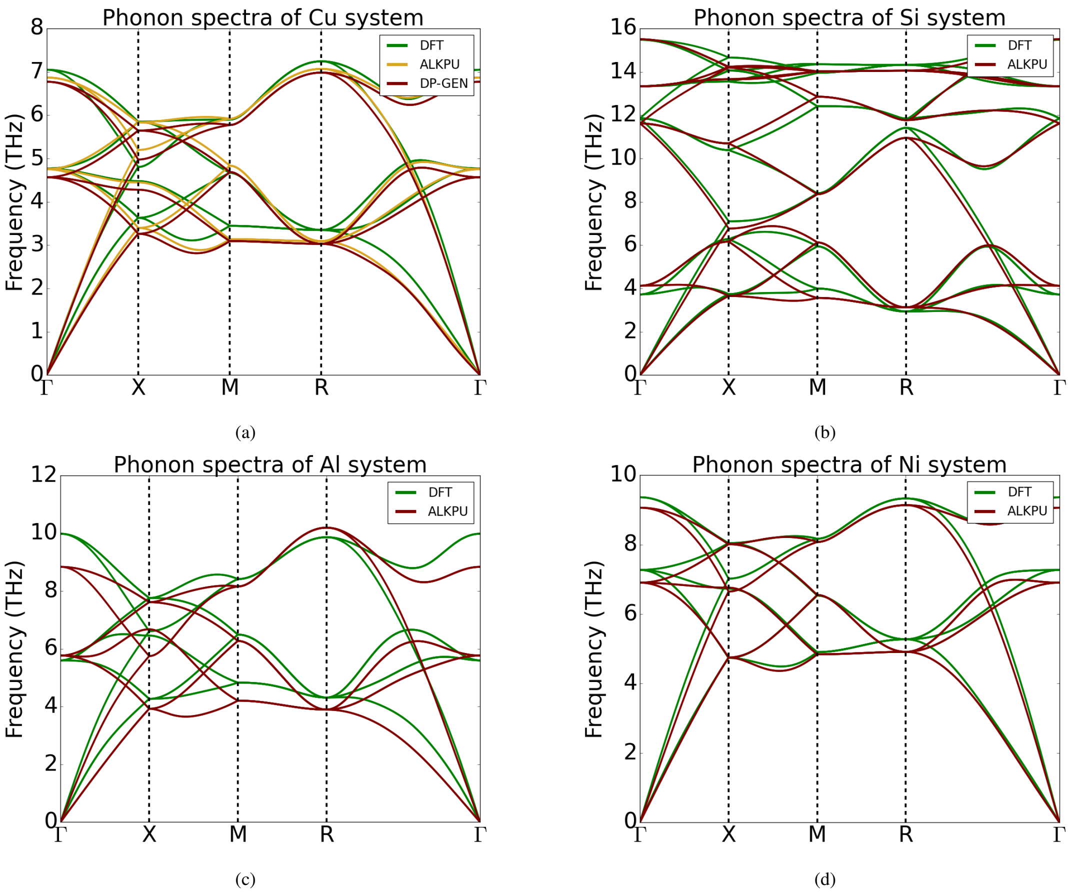


Figure 6: Comparison of phonon spectra calculated by DFT and DeePMD with ALKPU: (a). Cu-108, FCC; (b). Si-64, diamond; (c). Al-108, FCC; (d). Ni-108, FCC.

In order to test the final trained model, we compare the photon spectra calculated by DFT and DeePMD with ALKPU for Cu, Si, Al and Ni systems. These test systems are chosen from the material project dataset [35]. Figure 6 shows that the trained model reproduces well the DFT results for photon spectra of the four systems (the results of Al are slightly different). We also compare the DP-GEN and ALKPU trained models for calculating Cu's photon spectra in Figure 6 (a), and the two results are very similar.

Overall, the test results indicate that ALKPU can effectively select representative data points to cover the configuration space during training. It has great potential for improving training efficiency to build a final usable model and can significantly accelerate MD simulations while maintaining *ab initio* accuracy. In our future work, the ALKPU method will be tested on more complex systems with much larger scales to further explore its capabilities.

6. Conclusion and outlook

To enhance the extrapolation capability and improve training efficiency for DeePMD, we have proposed ALKPU, an active learning method that can concurrently select representative configurations for labelling during model training. his approach uses the efficient RLEKF optimizer based on a Kalman filter and introduces Kalman Prediction Uncertainty (KPU) to quantify model prediction uncertainty. We developed an efficient algorithm to calculate KPU for atomic forces and an ALKPU sampling platform that concurrently explores, selects, and labels representative configurations during training. We have shown that ALKPU is essentially a locally fastest reduction procedure of model's uncertainty. We test the ALKPU method with various physical system simulations and compare it with DeePMD's original DP-GEN active learning method. Our results show that ALKPU improves training efficiency and reduces computational overhead, offering significant advantages as a new active learning approach.

Although ALKPU is designed for DeePMD, KPU is a general metric for quantifying uncertainty in NN predictions. When combined with an efficient Kalman filter-based optimizer, ALKPU can be applied to other "AI for Science" problems where model training is time-consuming and acquiring training data is computationally expensive.

Appendix A. The extended Kalman filter and RLEKF optimizer

We first review the basic algorithm of the extended Kalman filter (EKF) and then derive the EKFfm algorithm. For notational simplicity, some notations in (A.1)–(A.10) may conflict with the notations in the main text, but the readers can easily find the differences between them.

EKF. Write a discrete stochastic dynamical system with observations as

$$\begin{cases} s_t = f(s_{t-1}, u_t) + \xi_t, \quad s_0 \sim \mathcal{N}(s_0, P_0), \\ y_t = h(s_t, u_t) + \eta_t, \quad j = 1, 2 \dots \end{cases} \tag{A.1}$$

where $f \in C^1(\mathbb{R}^n, \mathbb{R}^n)$, $h \in C^1(\mathbb{R}^n, \mathbb{R}^m)$ and $\xi_t \sim \mathcal{N}(0, Q_t)$, $\eta_t \sim \mathcal{N}(0, R_t)$ are noises. The first equation describes the evolution of the system with initial state s_0, while the second equation describes the observations, and it is usually assumed that s_0, $\{\xi_t\}_{t\in\mathbb{N}}$ and $\{\eta_t\}_{t\in\mathbb{N}}$ are independent. The Kalman filter is an iterative algorithm that estimates the distribution $\mathbb{P}\left(s_t|\{y_i\}_{i=1}^{i=t}\right)$ by computing its expectation and covariance [40, 15]. For nonlinear mappings f and g, the EKF applies the linear Kalman filter to a linear Gaussian system approximated by using one-order Taylor expansion, and the output is an approximation to the real distribution of $\mathbb{P}\left(s_t|\{y_i\}_{i=1}^{t}\right)$.

Given the initial state $s_0 \sim \mathcal{N}(s_0, P_0)$, for $t = 1, 2, \dots$, the prediction step is

$$F_{t-1} = \mathrm{D}_s f(\hat{s}_{t-1}, u_t), \tag{A.2}$$
$$\bar{s}_t = f(\hat{s}_{t-1}, u_t), \tag{A.3}$$
$$\bar{P}_t = F_{t-1} P_{t-1} F_{t-1}^T + Q_t. \tag{A.4}$$

The correction step after observing y_t is

$$H_t = \mathrm{D}_s h(\bar{s}_t, u_t), \tag{A.5}$$
$$A_t = H_t \bar{P}_t H_t^T + R_t, \tag{A.6}$$
$$K_t = \bar{P}_t H_t^T A_t^{-1}, \tag{A.7}$$
$$P_t = (I - K_t H_t)\bar{P}_t, \tag{A.8}$$
$$d_t = y_t - h(\bar{s}_t, u_t), \tag{A.9}$$
$$\hat{s}_t = \bar{s}_t + K_t d_t. \tag{A.10}$$

The desired outputs are the expectation $\hat{s}_t$ and covariance P_t.

Derivation of EKFfm. Notations in this derivation follow those in the main text. By the EKF algorithm, we can estimate the state $\boldsymbol{v}_t$ and then let $\hat{\boldsymbol{w}}_t = \alpha_t^{-1}\hat{\boldsymbol{v}}_t$. First, we have $\bar{\boldsymbol{v}}_t = \lambda_t^{-1/2}\hat{\boldsymbol{v}}_{t-1}$ and $F_{t-1} = \lambda_t^{-1/2} I$, and thus $\bar{P}_t = \lambda_t^{-1} P_{t-1}$. Then by (A.5) we have

$$\tilde{H}_t := \mathrm{D}_{\boldsymbol{v}} h(\alpha_t^{-1}\bar{\boldsymbol{v}}_t, x_t) = \alpha_t^{-1}\mathrm{D}_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_{t-1}, x_t) = \alpha_t^{-1} H_t$$

since $\alpha_t^{-1}\bar{\boldsymbol{v}}_t = \alpha_t^{-1}\lambda_t^{-1/2}\alpha_{t-1}\hat{\boldsymbol{w}}_{t-1} = \hat{\boldsymbol{w}}_{t-1}$. By (A.6) we have

$$\tilde{A}_t := \tilde{H}_t \bar{P}_t \tilde{H}_t^T + R_t = \alpha_t^{-2}(H_t \bar{P}_t H_t^T + \alpha_t^2 R_t) = \alpha_t^{-2} A_t.$$

By (A.7) we have

$$\tilde{K}_t = \bar{P}_t \tilde{H}_t^T \tilde{A}_t^{-1} = \alpha_t (\bar{P}_t H_t^{-1} A_t^{-1}) = \alpha_t K_t.$$

Therefore, the estimated covariance of $\boldsymbol{v}_t|\{y_i\}_{i=1}^t$ is

$$P_t = (I - \tilde{K}_t \tilde{H}_t)\bar{P}_t = (I - K_t H_t)\bar{P}_t.$$

By (A.9) we have

$$d_t = y_t - h(\alpha_t^{-1}\bar{\boldsymbol{v}}_t, x_t) = y_t - h(\hat{\boldsymbol{w}}_{t-1}, x_t).$$

Therefore, the estimated expectation of $\boldsymbol{v}_t|\{y_i\}_{i=1}^t$ is

$$\hat{\boldsymbol{v}}_t = \bar{\boldsymbol{v}}_t + \tilde{K}_t d_t = \lambda_t^{-1/2}\hat{\boldsymbol{v}}_{t-1} + \alpha_t K_t d_t,$$

which finally leads to

$$\hat{\boldsymbol{w}}_t = \alpha_t^{-1}\hat{\boldsymbol{v}}_t = \alpha_t^{-1}\lambda_t^{-1/2}\alpha_{t-1}\hat{\boldsymbol{w}}_{t-1} + K_t d_t = \hat{\boldsymbol{w}}_{t-1} + K_t d_t.$$

One can check the correspondance of computations in the above and in Algorithm 1. From the derivation we known that the estimated covariance of $\boldsymbol{w}$ at the t-th step is

$$\hat{P}_t = \alpha_t^{-2} P_t.$$

RLEKF optimizer. The RLEKF approximates dense matrices in EKFfm with sparse diagonal block ones to reduce the computational cost of matrix-matrix/vector multiplications. Technically, all the layers of the NN are reorganized by splitting big and gathering adjacent small layers.

The layerwise weight-updating procedure of RLEKF for training DeePMD in shown in Algorithm 4. In lines 1–6, we first transform a vector Y into a scalar such that the prediction of the NN model has dimension 1. `split` is a function for splitting and reorganizing layers, while `gather` is a function for gathering all reorganized layers together. Some computations have been modified to make the practical implementation more robust; for details see [33].

Algorithm 4 $\texttt{RLEKF}(\boldsymbol{Y}, \boldsymbol{Y}_{\mathrm{DFT}}, \boldsymbol{w}^{in}, P^{in}, \lambda^{in})$

Input: $\boldsymbol{w}^{in}, P^{in}, \lambda^{in}$

1: **for** $j = 1, 2, \ldots, \texttt{length}(\boldsymbol{Y})$ **do**
2:     **if** $Y_j \geq Y_{j,\mathrm{DFT}}$ **then**
3:         $Y_j = -Y_j;\quad Y_{j,\mathrm{DFT}} = -Y_{j,\mathrm{DFT}}$
4:     **end if**
5: **end for**
6: $Y = \sum_j Y_j/\texttt{length}(\boldsymbol{Y});\quad \mathrm{er} = \sum_j (Y_{j,\mathrm{DFT}} - Y_j)/\texttt{length}(\boldsymbol{Y})$
7: $H = \nabla_{\boldsymbol{w}} Y\,|_{w=w^{in}}$ ▷ Backward propagation
8: $\alpha = 1/(\lambda^{in} L + H^T P^{in} H)$ ▷ L is the number of reorganized layers
9: $\{P_1^{in}, \ldots, P_L^{in}\} = \texttt{split}(P^{in});\quad \{\boldsymbol{w}_1^{in}, \ldots, \boldsymbol{w}_L^{in}\} = \texttt{split}(P^{in});\quad \{H_1, \ldots, H_L\} = \texttt{split}(H)$ ▷ Reorganizing by layers
10: **for** $l = 1, 2, \ldots, L$ **do**
11:     $K = P_l^{in} H_l$
12:     $P_l^{out} = (P_l^{in} - \alpha K K^T)/\lambda^{in}$
13:     $\boldsymbol{w}_l^{out} = \boldsymbol{w}_l^{in} + \alpha \mathrm{er} K Y$
14: **end for**
15: $\boldsymbol{w}^{out} = \texttt{gather}(\boldsymbol{w}_1^{out}, \ldots, \boldsymbol{w}_L^{out})$
16: $P^{out} = \texttt{gather}(P_1^{out}, \ldots, P_L^{out})$ ▷ Gathering all layers
17: $\lambda^{out} = \lambda^{in}\nu + 1 - \nu$

Output: $\boldsymbol{w}^{out}, P^{out}, \lambda^{out}$

The overall structure of the training procedure is outlined in Algorithm 5. The algorithm begins by initializing $\hat{\boldsymbol{w}}_0$, P_0 and λ_1. At the t-th step, the energy process fits the predicted total energy to the labeled energy of configuration $\mathcal{R}^{(t)}$ by nonlinear NN mapping h_E, while the force process fits the predicted atom forces to the labeled forces by nonlinear mapping h_F that is the gradient of h_E with respect to coordinates. During the force process, a set of n_a atoms is randomly chosen from the configuration and the corresponding force vectors are concatenated into a scalar (i.e. lines 1–6 of Algorithm 4). This process (lines 4–5) is repeated n_F times in one step. In the practical implementation we set $n_a = 6$ and $n_F = 4$.

Algorithm 5 High-level structure of training by RLEKF

Input: $\hat{\boldsymbol{w}}_0$, P_0, λ_1, ν, training data $\{\mathcal{R}^{(t)}, E^{(t)}_{\text{DFT}}, \{\boldsymbol{F}^{(t)}_{i,\text{DFT}}\}\}_{t\in\mathbb{N}}$

for $t = 1, 2, \ldots, T$ **do**

  $E = h_E(\hat{\boldsymbol{w}}_{t-1}, \mathcal{R}^{(t)})$ ▹ Predicted total energy

  $\hat{\boldsymbol{w}}_+, P_+, \lambda_+ = \texttt{RLEKF}(E, E^{(t)}_{\text{DFT}}, \hat{\boldsymbol{w}}_{t-1}, \mathbf{P}_{t-1}, \lambda_t)$ ▹ Training by energy

  $\{\boldsymbol{F}_i\} = h_F(\hat{\boldsymbol{w}}_+, \mathcal{R}^{(t)})$ ▹ Predicted forces

  $\hat{\boldsymbol{w}}_t, P_t, \lambda_{t+1} = \texttt{RLEKF}(\{\boldsymbol{F}_i\}, \{\boldsymbol{F}^{(t)}_{i,\text{DFT}}\}, w_+, \mathbf{P}_+, \lambda_+)$ ▹ Training by forces

end for

Output: $\hat{\boldsymbol{w}}_T$, P_T, λ_{T+1}

Appendix B. Shannon entropy of the multivariate Gaussian distribution

A vector-valued random variable $X = [X_1, \ldots, X_n]^T \in \mathbb{R}^n$ is said to have a multivariate Gaussian distribution if its probability density function is given by

$$p(x) = \frac{1}{(2\pi)^{n/2}(\det\Sigma)^{1/2}} \exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big), \tag{B.1}$$

with the expectation (or mean) $\mu \in \mathbb{R}^n$ and covariance $\Sigma \in \mathbb{R}^{n\times n}$ where Σ is semi-definite. We write it as $X \sim \mathcal{N}(\mu, \Sigma)$.

The Shannon entropy of X can be calculated as follows. First,

$$\begin{aligned} H(X) &= -\int_{\mathbb{R}^n} p(x)\ln p(x)\mathrm{d}x \\ &= -\int_{\mathbb{R}^n} [\ln((2\pi)^{n/2}(\det\Sigma)^{1/2}) + \frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]p(x)\mathrm{d}x \\ &= \frac{n}{2}\ln(2\pi) + \frac{1}{2}\ln(\det\Sigma) + \frac{1}{2}\mathbb{E}_{x\sim p(x)}[(x-\mu)^T\Sigma^{-1}(x-\mu)]. \end{aligned}$$

Note that $(x-\mu)^T\Sigma^{-1}(x-\mu)$ is a scalar. Thus we have

$$\begin{aligned} \mathbb{E}_{x\sim p(x)}[(x-\mu)^T\Sigma^{-1}(x-\mu)] &= \mathbb{E}_{x\sim p(x)}[\mathrm{tr}((x-\mu)^T\Sigma^{-1}(x-\mu))] \\ &= \mathbb{E}_{x\sim p(x)}[\mathrm{tr}(\Sigma^{-1}(x-\mu)(x-\mu)^T)] \\ &= \mathrm{tr}(\Sigma^{-1}\mathbb{E}_{x\sim p(x)}[(x-\mu)(x-\mu)^T]) \\ &= \mathrm{tr}(\Sigma^{-1}\Sigma) \\ &= n. \end{aligned}$$

Therefore, we finally obtain

$$H(X) = \frac{n}{2}(1 + \ln(2\pi)) + \frac{1}{2}\ln(\det\Sigma). \tag{B.2}$$

Note that for the n dimensional Gaussian random vectors, the Shannon entropy depends only on its covariance but not its mean.

Appendix C. Proofs

Proof of Theorem 4.1. In Algorithm 1, at the t-th step we have

$$\begin{aligned} P_t &= \bar{P}_t - \bar{P}_t H_t^T (H_t \bar{P}_t H_t^T + \alpha_t^2 R_t)^{-1} H_t \bar{P}_t \\ &= (\bar{P}_t^{-1} + H_t^T \alpha_t^{-2} R_t^{-1} H_t)^{-1} \\ &= (\lambda_t P_{t-1}^{-1} + H_t^T \alpha_t^{-2} R_t^{-1} H_t)^{-1}, \end{aligned}$$

where the second equality is obtained by using the Sherman-Morrison-Woodbury formula, which leads to

$$P_t^{-1} = \lambda_t P_{t-1}^{-1} + \alpha_t^{-2} H_t^T R_t^{-1} H_t. \tag{C.1}$$

Let $\hat{y}_t = h(\hat{\boldsymbol{w}}_{t-1}, x_t)$ and $\ell_t(\boldsymbol{w}) = \ln p(y_t|\boldsymbol{w})$. Since $y_t = h(\boldsymbol{w}, x_t) + \eta_t$ with $\eta_t \sim \mathcal{N}(0, R_t)$, we have

$$p(y_t|\hat{\boldsymbol{w}}_{t-1}) \propto \exp\Big(-\frac{1}{2}(y_t - \hat{y}_t)^T R_t^{-1} (y_t - \hat{y}_t)\Big),$$

and thus

$$\ell_t(\hat{\boldsymbol{w}}_{t-1}) = -\frac{1}{2}(y_t - \hat{y}_t)^T R_t^{-1}(y_t - \hat{y}_t) + C_1 \tag{C.2}$$

with C_1 a constant number. Notice that

$$\nabla_{\boldsymbol{w}} \ell_t(\hat{\boldsymbol{w}}_{t-1}) = H_t^T \nabla_{\hat{y}} \ell_t \big|_{\hat{y}_t},$$

where $\nabla_{\hat{y}} \ell_t$ is the gradient with respect to the middle variable $\hat{y}$. Therefore

$$\mathbb{E}_{y_t \sim p(y_t|\hat{\boldsymbol{w}}_{t-1})}\Big[\nabla_{\boldsymbol{w}} \ell_t(\hat{\boldsymbol{w}}_{t-1})^{\otimes 2}\Big] = H_t^T \mathbb{E}_{y_t}\Big[\Big(\nabla_{\hat{y}} \ell_t\big|_{\hat{y}_t}\Big)^{\otimes 2}\Big] H_t.$$

By (C.2) we have

$$\begin{aligned} \mathbb{E}_{y_t}\Big[\Big(\nabla_{\hat{y}} \ell_t\big|_{\hat{y}_t}\Big)^{\otimes 2}\Big] &= \mathbb{E}_{y_t}[R_t^{-1}(y_t - \hat{y}_t)][R_t^{-1}(y_t - \hat{y}_t)]^T \\ &= R_t^{-1} \mathbb{E}_{y_t}[(y_t - \hat{y}_t)(y_t - \hat{y}_t)^T] R_t^{-1} \\ &= R_t^{-1}. \end{aligned}$$

Combining with (C.1) we obtain

$$P_t^{-1} = \lambda_t P_{t-1}^{-1} + \alpha_t^{-2} \mathbb{E}_{y_t \sim p(y_t|\hat{\boldsymbol{w}}_{t-1})}\Big[\nabla_{\boldsymbol{w}} \ell_t(\hat{\boldsymbol{w}}_{t-1})^{\otimes 2}\Big]. \tag{C.3}$$

By defining $J_t = \gamma_t P_t^{-1}$ we get

$$J_t = \lambda_t \frac{\gamma_t}{\gamma_{t-1}} J_{t-1} + \frac{\gamma_t}{\alpha_t^2} \mathbb{E}_{y_t}\Big[\nabla_{\boldsymbol{w}} \ell_t(\hat{\boldsymbol{w}}_{t-1})^{\otimes 2}\Big].$$

By letting $\lambda_t \frac{\gamma_t}{\gamma_{t-1}} + \frac{\gamma_t}{\alpha_t^2} = 1$ we finally obtain (4.2)–(4.4). □

Proof of Proposition 4.1. First we prove $\lim_{t\to\infty} \lambda_t = 1$ and

$$\lim_{t\to\infty} \alpha_t^{-2} = C_2 \tag{C.4}$$

with C_2 a positive number. We have

$$\begin{aligned} \lambda_t &= 1 - \nu + \nu\lambda_{t-1} = 1 - \nu + \nu(1 - \nu + \nu\lambda_{t-2}) \\ &= \cdots \\ &= (1-\nu)(1 + \nu + \cdots + \nu^{t-2}) + \nu^{t-1}\lambda_1 \\ &= 1 - \nu^{t-1} + \nu^{t-1}\lambda_1 \longrightarrow 1, \quad t \to \infty. \end{aligned}$$

Note that $\alpha_t^{-2} = \lambda_1\lambda_2\cdots\lambda_t$, which leads to

$$\begin{aligned}
\ln\alpha_t^{-2} &= \sum_{i=1}^{t}\ln(1-\nu^{i-1}(1-\lambda_1))\\
&\geq \sum_{i=1}^{t}\frac{\nu^{i-1}(1-\lambda_1)}{\nu^{i-1}(1-\lambda_1)-1}\\
&\geq \sum_{i=1}^{t}\frac{\nu^{i-1}(1-\lambda_1)}{-\lambda_1}\\
&= -\frac{1-\lambda_1}{\lambda_1}\frac{1-\nu^t}{1-\nu} \longrightarrow -\frac{1-\lambda_1}{\lambda_1(1-\nu)},\quad t\to\infty,
\end{aligned}$$

where we have used the inequality $\ln(1-x) \geq x/(x-1)$. Since $\ln\alpha_t^{-2}$ deceases and have a finite lower bound, it follows that α_t^{-2} has a positive limit. By (4.4) we have

$$\begin{aligned}
\frac{1}{\gamma_t} &= \alpha_t^{-2} + \lambda_t\frac{1}{\gamma_{t-1}} = \alpha_t^{-2} + \lambda_t(\alpha_{t-1}^{-2} + \lambda_{t-1}\frac{1}{\gamma_{t-2}})\\
&= \cdots\\
&= \alpha_t^{-2} + \lambda_t\alpha_{t-1}^{-2} + \lambda_t\lambda_{t-1}\alpha_{t-2}^{-2} + \cdots + \lambda_t\lambda_{t-1}\cdots\lambda_2\alpha_1^{-2} + \lambda_t\lambda_{t-1}\cdots\lambda_1\frac{1}{\gamma_0}\\
&= t\alpha_t^{-2} + \alpha_t^{-2}\frac{1}{\gamma_0} \longrightarrow +\infty,\quad t\to\infty
\end{aligned}$$

due to $\lim_{t\to\infty}\alpha_t^{-2} = C_2 > 0$. Thus we have $\lim_{t\to\infty}\gamma_t = 0$ and

$$\lim_{t\to\infty}\beta_t = \frac{\lim_{t\to\infty}\lambda_t}{\lim_{t\to\infty}(\lambda_t + \alpha_t^{-2}\gamma_{t-1})} = \frac{1}{1 + C_2\cdot 0} = 1,$$

which are the desired results. □

Proof of Theorem 4.2. Since $y = h(\boldsymbol{w}, x) + \eta$ with $\eta \sim \mathcal{N}(0, \kappa^{-1}I)$, by the Bayes' formula we have

$$\begin{aligned}
p(\boldsymbol{w}|y) &= \frac{p(y|\boldsymbol{w})p(\boldsymbol{w})}{p(y)}\\
&\propto \exp\Big(-\frac{\kappa}{2}(y-h(\boldsymbol{w},x))^T(y-h(\boldsymbol{w},x))\Big)\exp\Big(-\frac{1}{2}(\boldsymbol{w}-\hat{\boldsymbol{w}}_t)^T\hat{P}_t^{-1}(\boldsymbol{w}-\hat{\boldsymbol{w}}_t)\Big).
\end{aligned}$$

Under the assumption (4.9), $\ln p(\boldsymbol{w}|y)$ is a quadratic form of $\boldsymbol{w}$, thus $\boldsymbol{w}|y$ is Gaussian. Suppose $\boldsymbol{w}|y \sim \mathcal{N}(\widehat{\boldsymbol{w}}_t, \widehat{P}_t)$. Equating the quadratic terms of $\boldsymbol{w}$ in $\ln p(\boldsymbol{w}|y)$ gives

$$\widehat{P}_t^{-1} = \hat{P}_t^{-1} + \kappa \mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)^T. \tag{C.5}$$

Therefore, by (B.2) the change of entropy of $\boldsymbol{w}$ is

$$S_t = \frac{1}{2}\ln(\det\hat{P}_t) - \frac{1}{2}\ln(\det\widehat{P}_t) = \frac{1}{2}\ln\Big(\frac{\det\widehat{P}_t^{-1}}{\det\hat{P}_t^{-1}}\Big). \tag{C.6}$$

By the Weinstein–Aronszajn identity for matrix determinant, we obtain from (C.5)

$$\begin{aligned}
\det\widehat{P}_t^{-1} &= \det\hat{P}_t^{-1}\cdot\det\Big(I + \kappa\hat{P}_t\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)^T\Big)\\
&= \det\hat{P}_t^{-1}\cdot\det\Big(I + \kappa\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)^T\hat{P}_t\mathrm{D}_{\boldsymbol{w}}h(\hat{\boldsymbol{w}}_t, x)\Big)\\
&= \det\hat{P}_t^{-1}\cdot\det\Big(I + \kappa\alpha_t^{-2}\mathrm{KPU}_t(x)\Big).
\end{aligned}$$

Combining with (C.6) we obtain the expression of S_t. □

Proof of Theorem 4.3. Let $\hat{y} = h(\boldsymbol{w}, x)$. Then

$$\begin{aligned}\mathbb{E}_{\boldsymbol{w}}[|y-\hat{y}_t|^2] &= \mathbb{E}_{\boldsymbol{w}}[|y-\hat{y}|^2] + \mathbb{E}_{\boldsymbol{w}}[|\hat{y}-\hat{y}_t|^2] + 2\mathbb{E}_{\boldsymbol{w}}[(y-\hat{y})(\hat{y}-\hat{y}_t)] \\ &= \mathbb{E}_{\boldsymbol{w}}[|y-\hat{y}|^2] + \mathbb{E}_{\boldsymbol{w}}[|\hat{y}-\hat{y}_t|^2] + 2\eta\mathbb{E}_{\boldsymbol{w}}[h(\boldsymbol{w}, x) - h(\hat{\boldsymbol{w}}_t, x)].\end{aligned}$$

Using the approximation $h(\boldsymbol{w}, x) - h(\hat{\boldsymbol{w}}_t, x) \approx \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)$ we get

$$\begin{aligned}\mathbb{E}_{\boldsymbol{w}}[h(\boldsymbol{w}, x) - h(\hat{\boldsymbol{w}}_t, x)] &= \mathbb{E}_{\boldsymbol{w}}[\nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)] \\ &= \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T \mathbb{E}_{\boldsymbol{w}}[\boldsymbol{w} - \hat{\boldsymbol{w}}_t] \\ &= 0.\end{aligned}$$

Therefore, we have

$$\mathbb{E}_{\boldsymbol{w}}[|y-\hat{y}_t|^2] = \mathbb{E}_{\boldsymbol{w}}[|y-\hat{y}|^2] + \mathbb{E}_{\boldsymbol{w}}[|\hat{y}-\hat{y}_t|^2] \geq \mathbb{E}_{\boldsymbol{w}}[|\hat{y}-\hat{y}_t|^2]. \tag{C.7}$$

Using again the approximation $h(\boldsymbol{w}, x) - h(\hat{\boldsymbol{w}}_t, x) \approx \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)$ we obtain

$$\begin{aligned}\mathbb{E}_{\boldsymbol{w}}[|\hat{y}-\hat{y}_t|^2] &= \mathbb{E}_{\boldsymbol{w}}[|h(\boldsymbol{w}, x) - h(\hat{\boldsymbol{w}}_t, x)|^2] \\ &= \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T \mathbb{E}_{\boldsymbol{w}}[(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)(\boldsymbol{w} - \hat{\boldsymbol{w}}_t)^T] \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x) \\ &= \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x)^T \hat{P}_t \nabla_{\boldsymbol{w}} h(\hat{\boldsymbol{w}}_t, x) \\ &= \alpha_t^{-2}\mathrm{KPU}_t(x).\end{aligned}$$

Combining the above equality with (C.7), the desired result is obtained. □

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 3192270206).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] S.-i. Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

[2] M. Aminpour, C. Montemagno, and J. A. Tuszynski. An overview of molecular modeling for drug discovery with specific illustrative examples of applications. *Molecules*, 24(9):1693, 2019.

[3] D. Angluin. Queries revisited. In *Algorithmic Learning Theory: 12th International Conference, ALT 2001 Washington, DC, USA, November 25–28, 2001 Proceedings 12*, pages 12–31. Springer, 2001.

[4] A. P. Bartók, R. Kondor, and G. Csányi. On representing chemical environments. *Physical Review B*, 87:184115, May 2013.

[5] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, May 2022.

[6] J. Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of Chemical Physics*, 134(7):074106, 2011.

[7] J. Behler. Representing potential energy surfaces by high-dimensional neural network potentials. *Journal of Physics: Condensed Matter*, 26(18):183001, apr 2014.

[8] J. Behler. Constructing high-dimensional neural network potentials: a tutorial review. *International Journal of Quantum Chemistry*, 115(16):1032–1050, 2015.

[9] J. Behler. First principles neural network potentials for reactive simulations of large molecular and condensed systems. *Angewandte Chemie International Edition*, 56(42):12828–12840, 2017.

[10] J. Behler. Four generations of high-dimensional neural network potentials. *Chemical Reviews*, 121(16):10037–10072, 2021.
[11] J. Behler and M. Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98:146401, Apr 2007.
[12] T. B. Blank, S. D. Brown, A. W. Calhoun, and D. J. Doren. Neural network models of potential energy surfaces. *The Journal of Chemical Physics*, 103(10):4129–4137, 1995.
[13] V. Botu and R. Ramprasad. Adaptive machine learning framework to accelerate ab initio molecular dynamics. *International Journal of Quantum Chemistry*, 115(16):1074–1083, 2015.
[14] R. Car and M. Parrinello. Unified approach for molecular dynamics and density-functional theory. *Physical Review Letters*, 55:2471–2474, Nov 1985.
[15] C. K. Chui and G. Chen. *Kalman filtering*. Springer, 2017.
[16] D. Cohn. Neural network exploration using optimal experiment design. *Advances in neural information processing systems*, 6, 1993.
[17] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.
[18] H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, Princeton, 1946.
[19] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi. Gaussian process regression for materials and molecules. *Chemical Reviews*, 121(16):10073–10141, 2021.
[20] S. Desai, S. T. Reeve, and J. F. Belak. Implementing a neural network interatomic model with performance portability for emerging exascale architectures. *Computer Physics Communications*, 270:108156, 2022.
[21] B. Faria, D. Perdigão, J. Brás, and L. Macedo. The joint role of batch size and query strategy in active learning-based prediction – a case study in the heart attack domain. In *Progress in Artificial Intelligence*, pages 464–475, Cham, 2022. Springer International Publishing.
[22] P. Friederich, F. Häse, J. Proppe, and A. Aspuru-Guzik. Machine-learned potentials for next-generation matter simulations. *Nature Materials*, 20(6):750–761, 2021.
[23] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059. PMLR, 2016.
[24] Y. Gal, J. Hron, and A. Kendall. Concrete dropout. *Advances in neural information processing systems*, 30, 2017.
[25] J. Gasteiger, S. Giri, J. T. Margraf, and S. Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. In *Machine Learning for Molecules Workshop, NeurIPS*, 2020.
[26] J. Gasteiger, J. Groß, and S. Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations (ICLR)*, 2020.
[27] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.
[28] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.
[29] M. Haghighatlari, J. Li, X. Guan, O. Zhang, A. Das, C. J. Stein, F. Heidar-Zadeh, M. Liu, M. Head-Gordon, L. Bertels, et al. Newtonnet: A newtonian message passing network for deep learning of interatomic potentials and forces. *Digital Discovery*, 1(3):333–343, 2022.
[30] S. S. Haykin. *Kalman filtering and neural networks*, volume 284. Wiley Online Library, 2001.
[31] F. Heimes. Extended Kalman filter neural network training: experimental results and algorithm improvements. In *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, volume 2, pages 1639–1644, 1998.
[32] J. M. Hernández-Lobato, J. Requeima, E. O. Pyzer-Knapp, and A. Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. In *International conference on machine learning*, pages 1470–1479. PMLR, 2017.
[33] S. Hu, W. Zhang, Q. Sha, F. Pan, L.-W. Wang, W. Jia, G. Tan, and T. Zhao. RLEKF: An optimizer for deep potential with ab initio accuracy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7910–7918, 2023.
[34] M. Hutchinson and M. Widom. VASP on a GPU: Application to exact-exchange calculations of the stability of elemental boron. *Computer Physics Communications*, 183(7):1422–1426, 2012.
[35] A. Jain, S. P. Ong, G. Hautier, W. Chen, W. D. Richards, S. Dacek, S. Cholia, D. Gunter, D. Skinner, G. Ceder, and K. A. Persson. Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1(1):011002, July 2013.
[36] W. Jia, Z. Cao, L. Wang, J. Fu, X. Chi, W. Gao, and L. Wang. The analysis of a plane wave pseudopotential density functional theory code on a GPU machine - sciencedirect. *Computer Physics Communications*, 184(1):9–18, 2013.
[37] W. Jia, J. Fu, Z. Cao, W. Long, and L. W. Wang. Fast plane wave density functional theory molecular dynamics calculations on multi-GPU machines. *Journal of Computational Physics*, 251(23):102–115, 2013.
[38] W. Jia, H. Wang, M. Chen, D. Lu, L. Lin, R. Car, W. E, and L. Zhang. Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In *SC20: International conference for high performance computing, networking, storage and analysis*, pages 1–14. IEEE, 2020.
[39] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. Development and testing of the opls all-atom force field on conformational energetics and properties of organic liquids. *Journal of the American Chemical Society*, 118(45):11225–11236, 1996.
[40] R. E. Kalman. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960.
[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.
[42] K. Lee, D. Yoo, W. Jeong, and S. Han. Simple-nn: An efficient package for training and executing neural-network interatomic potentials. *Computer Physics Communications*, 242:95–103, 2019.
[43] Y. Li and Y. Gal. Dropout inference in Bayesian neural networks with alpha-divergences. In *International conference on machine learning*, pages 2052–2061. PMLR, 2017.
[44] S. Lorenz, A. Gro, and M. Scheffler. Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks.

Chemical Physics Letters, 395(4–6):210–215, 2004.
[45] D. J. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.
[46] D. Marx and J. Hutter. *Ab Initio Molecular Dynamics: Basic Theory and Advanced Methods*. Cambridge University Press, 2009.
[47] S. Nosé. A molecular dynamics method for simulations in the canonical ensemble. *Molecular Physics*, 52(2):255–268, 1984.
[48] E. V. Podryabinkin and A. V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Computational Materials Science*, 140:171–180, 2017.
[49] J. Proppe, S. Gugler, and M. Reiher. Gaussian process-based refinement of dispersion corrections. *Journal of Chemical Theory and Computation*, 15(11):6046–6060, 2019.
[50] G. Puskorius and L. Feldkamp. Decoupled extended Kalman filter training of feedforward layered networks. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume 1, pages 771–777, 1991.
[51] C. R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society.*, 37:81–89, 1945.
[52] C. E. Rasmussen, C. K. Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.
[53] J.-Y. Raty, F. Gygi, and G. Galli. Growth of carbon nanotubes on metal nanoparticles: a microscopic mechanism from ab initio molecular dynamics simulations. *Physical Review Letters*, 95(9):096103, 2005.
[54] N. Roy and A. McCallum. Toward optimal active learning through Monte Carlo estimation of error reduction. *ICML*, 2:441–448, 2001.
[55] K. Schütt, O. Unke, and M. Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.
[56] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8(1):13890, Jan 2017.
[57] B. Settles. Active learning literature survey. *University of Wisconsin Madison*, 2010.
[58] B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. *Advances in neural information processing systems*, 20, 2007.
[59] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.
[60] G. N. Simm and M. Reiher. Error-controlled exploration of chemical reaction networks with Gaussian processes. *Journal of Chemical Theory and Computation*, 14(10):5238–5248, 2018.
[61] J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev, and A. E. Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of Chemical Physics*, 148(24):241733, 2018.
[62] F. H. Stillinger and T. A. Weber. Computer simulation of local order in condensed phases of silicon. *Physical Review B*, 1985.
[63] O. T. Unke, S. Chmiela, M. Gastegger, K. T. Schütt, H. E. Sauceda, and K.-R. Müller. Spookynet: Learning force fields with electronic degrees of freedom and nonlocal effects. *Nature Communications*, 12(1):7273, Dec 2021.
[64] H. Wang, L. Zhang, J. Han, and W. E. DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, 228:178–184, 2018.
[65] K. Yao, J. E. Herr, S. N. Brown, and J. Parkhill. Intrinsic bond energies from a bonds-in-molecules neural network. *The Journal of Physical Chemistry Letters*, 8(12):2689–2694, 2017. PMID: 28573865.
[66] J. Zeng, L. Cao, M. Xu, T. Zhu, and J. Z. Zhang. Neural network based in silico simulation of combustion reactions. *arXiv:1911.12252*, 2019.
[67] J. Zeng, L. Zhang, H. Wang, and T. Zhu. Exploring the chemical space of linear alkane pyrolysis via deep potential generator. *Energy & Fuels*, 35(1):762–769, 2020.
[68] L. Zhang. *Machine learning for multi-scale molecular modeling: theories, algorithms, and applications*. PhD thesis, Princeton University, 2020.
[69] L. Zhang, J. Han, H. Wang, and R. Car. Deep potential molecular dynamics: a scalable model with the accuracy of quantum mechanics. *Physical Review Letters*, 120(14):143001, 2018.
[70] L. Zhang, J. Han, H. Wang, W. Saidi, R. Car, et al. End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems. *Advances in neural information processing systems*, 31, 2018.
[71] L. Zhang, D.-Y. Lin, H. Wang, R. Car, and W. E. Active learning of uniformly accurate interatomic potentials for materials simulation. *Physical Review Materials*, 3(2):023804, 2019.
[72] Y. Zhang, H. Wang, W. Chen, J. Zeng, L. Zhang, H. Wang, and W. E. DP-GEN: A concurrent learning platform for the generation of reliable deep learning based potential energy models. *Computer Physics Communications*, 253:107206, 2020.