RLEKF: An Optimizer for Deep Potential with *Ab Initio* Accuracy

Siyu Hu[1,2*], Wentao Zhang[3*], Qiuchen Sha[1,2], Feng Pan[3], Lin-Wang Wang[4], Weile Jia[1], Guangming Tan[1], Tong Zhao[1†]

[1] State Key Lab of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]School of Advanced Materials, Shenzhen Graduate School, Peking University, Shenzhen, China
[4]Institute of Semiconductors, Chinese Academy of Sciences, Beijing, China
husiyu20b@ict.ac.cn, zhang_wt@pku.edu.cn, shaqiuchen22@mails.ucas.ac.cn, panfeng@pkusz.edu.cn, lwwang@semi.ac.cn, {jiaweile,tgm,zhaotong}@ict.ac.cn

Abstract

It is imperative to accelerate the training of neural network force field such as Deep Potential, which usually requires thousands of images based on first-principles calculation and a couple of days to generate an accurate potential energy surface. To this end, we propose a novel optimizer named reorganized layer extended Kalman filtering (RLEKF), an optimized version of global extended Kalman filtering (GEKF) with a strategy of splitting big and gathering small layers to overcome the $O(N^2)$ computational cost of GEKF. This strategy provides an approximation of the dense weights error covariance matrix with a sparse diagonal block matrix for GEKF. We implement both RLEKF and the baseline Adam in our αDynamics package and numerical experiments are performed on 13 unbiased datasets. Overall, RLEKF converges faster with slightly better accuracy. For example, a test on a typical system, bulk copper, shows that RLEKF converges faster by both the number of training epochs ($\times$11.67) and wall-clock time ($\times$1.19). Besides, we theoretically prove that the updates of weights converge and thus are against the gradient exploding problem. Experimental results verify that RLEKF is not sensitive to the initialization of weights. The RLEKF sheds light on other AI-for-science applications where training a large neural network (with tons of thousands parameters) is a bottleneck.

Introduction

Ab initio molecular dynamics (AIMD) has been the method of choice in modeling physical phenomena from a microscopic scale, for example, water (Rahman and Stillinger 1971; Stillinger and Rahman 1974), alloy (Wang et al. 2009), nanotube (Raty, Gygi, and Galli 2005), and even protein (Karplus and Kuriyan 2005). However, the cubic scaling of the first-principles methods (Meier, Laino, and Curioni 2014) has hindered both spatial and temporal scales of AIMD packages within thousands of atoms and picoseconds on modern supercomputers. To overcome the "scaling wall" of AIMD, two types of machine-learned MD (MLMD) methods are adopted. The first one is based on classical ML methods. In 1992, Ercolessi and Adam first introduced ML for describing potentials with an accuracy comparable to that obtained by *ab initio* methods (Ercolessi and Adams 1992, 1994), and to date, methods like ACE (Drautz 2019; Lysogorskiy et al. 2021), SNAP (Thompson et al. 2015), and GPR (Bartók et al. 2010) are developed and widely used in physical problems such as copper and silicon, tantalum, bulk crystals. The second approach is based on neural network (NN) and was first introduced in 2007 by Behler and Parrinello (Behler and Parrinello 2007). The neural network MD (NNMD) method approximates both atomic energy (E_i) and force (F_i) with a local neighboring environment, and the atomic potential energy surface is trained through tons of data generated from first-principles calculations. One current state-of-the-art is the Deep Potential (DP) model, which combines large NN and physical symmetries (translation, rotation, and permutation invariance) for accurately describing the high-dimensional configuration space of interatomic potential. Although the corresponding package DeePMD-kit can reach 10 billion atoms when scaling to the top supercomputers (Jia et al. 2020; Guo et al. 2022) in model inference , training procedure of an individual model can still take from hours to days and is the bottleneck.

The two most commonly used training methods are Adam (Kingma and Ba 2014) and stochastic gradient descent (SGD) (Saad 1998) in NNMD packages due to their integration in NN framework such as TensorFlow and PyTorch. For example, many NNMD packages such as HDNNP (Behler 2014), SIMPLE-NN (Lee et al. 2019) adopt Adam in the training of interatomic potential. Yet these optimizers have a slow convergence rate in searching for the optimal solution on the landscape and can take up to hundreds of epochs in training one NNMD model with thousands of training data. Moreover, SGD suffers from gradient exploding without proper control of the learning rate.

Global extended Kalman filtering (Chui, Chen et al. 2017) (GEKF) is a good choice in both convergence and robustness. For example, RuNNer (Behler 2011) adopts GEKF as an optimizer in its training of a simple three-layer fully connected NN with 1000 parameters or so. As shown in (Singraber et al. 2019), RuNNer can achieve 0.69 meV/atom in Energy RMSE and 35.5 meV/Å in Force RMSE for H_2O

[*]These authors contributed equally.
[†]Corresponding author

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

physical system. However, since the error covariance matrix $\mathbf{P}$ is updated globally (Fig. 2), GEKF can be computationally expensive when NNs with tens of thousands of parameters are applied.

Our **main contribution** is a reorganized layer extended Kalman filtering (RLEKF) method, which approximates the dense weights error covariance matrix of GEKF with a sparse diagonal block matrix to reduce the computational cost. Technically, these layers are reorganized by splitting big and gathering adjacent small layers. To have a fair comparison, both RLEKF and Adam methods are implemented in our NNMD package αDynamics. Compared to the Adam method, our testing results show that RLEKF can reach the same or higher accuracy for both force (better than Adam in 7 out of 13 testing cases) and energy (better than Adam in 11 out of 13 testing cases). For a typical copper system, the time-to-solution of RLEKF can be $\times 11.67$ and $\times 1.19$ faster in terms of the number of epochs and wall-clock time, respectively. Especially, RLEKF can significantly reduce the number of epochs to 2-3 for reaching a reasonable accuracy ($1.2\times$ the RMSE of the best accuracy possible). We theoretically prove the weights updating convergence and therefore this protects RLEKF from gradient exploding. Our work also sheds light on other NN-based applications where training NNs with a relatively large number of parameters is a bottleneck.

Related Work

NNMD Packages. Fully connected NNs are the most widely used in NNMD packages. For example, HDNNP (Behler and Parrinello 2007; Behler 2017), which is a three-layered fully-connected NN is introduced in RuNNer (Behler 2011). Other packages such as BIM-NN (Yao et al. 2017), SimpleNN (Lee et al. 2019), CabanaMD-NNP (Desai, Reeve, and Belak 2022), SPONGE (Huang et al. 2022), DeepMD-kit (Wang et al. 2018) are also implemented via fully-connected NN. Many physical phenomena such as a diagram of water and Cu_2S (Singraber et al. 2019), chemical molecule (Yao et al. 2017), SiO_2 (Lee et al. 2019), organic molecules ($C_{10}H_2$, $C_{10}H_3^+$) (Lysogorskiy et al. 2021) are studied with the packages above.

Recent progress of NNMD packages implemented with graph NN (GNN) is gaining momentum. DimeNet++ (Gasteiger, Groß, and Günnemann 2020; Gasteiger et al. 2020), NequIP (Batzner et al. 2022), has shown great potential in describing organic molecules (QM7 and QM9 dataset) at high accuracy. We remark that NNMD packages also differ in employing physical symmetries for NN inputs ("features"), which are not discussed due to it is out of the scope of this paper.

Training Methods in NNMD Packages. Nearly all NNMD packages mentioned above use Adam and SGD as the training procedure for their ease of use in NN frameworks. One challenge of these methods is their time-consuming model training. For example, it usually takes more than 100 epochs to systematically train an individual DP model in the DeePMD-kit software.

As an alternative, Kalman filtering (Kalman et al. 1960) (KF) aims to estimate states of a linear process theoretically based on a state-measurement model with Gaussian noise by an estimator optimal in the sense of minimizing the mean square error of predicted states with the noisy measurement as input. It is widely used in autonomous, navigation, and interactive computer graphics due to its fast convergence and noise filtering. However, the formulation of KF confines itself to a linear optimization problem. Extended Kalman filtering (Smith, Schmidt, and McGee 1962) (EKF) is introduced to solve nonlinear optimization problems through Taylor expansion, and then the linearized problem is solved by KF. In the implementation, EKF has many variants such as NDEKF (Murtuza and Chorian 1994), ONDEKF, LDEKF, and FDEKF. Note that the performance of EKF and its variants are compared in Ref. (Heimes 1998).

We remark that in training an NNMD model with more than tens of thousands of parameters, such as the DP mentioned in this paper, both Adam and EKF (and its currently known variants) are either not effective or not efficient.

Problem Setup and Formulation

DP. The key steps of the DP training are shown in Fig. 1(a). For each atom i, the physical symmetries such as translational, rotational, and permutational invariances are integrated into the descriptor $\mathcal{D}_i$ through a three-layer fully connected network named embedding net. Then $\mathcal{D}_i$ is trained via a three-layered fitting net.

1. Every snapshot of the molecule system consists of each atom's 3D Cartesian coordinates $\mathbf{r}_i = (x_i, y_i, z_i) \in R^3, i = 1, 2, ..., N_a$ which is then translated into neighbor list of atom i, $\mathcal{R}_i = \{\mathbf{r}_{ij} \in \mathbb{R}^3 | \ | \mathbf{r}_j - \mathbf{r}_i | < r_c\}$ as the input of DP NN. Then, we gather it into its smooth version $\tilde{\mathcal{R}}_i \in \mathbb{R}^{N_m \times 4}$, $(\tilde{\mathcal{R}}_i)_j = s(|\mathbf{r}_{ij}|)(1, \mathbf{r}_{ij}/|\mathbf{r}_{ij}|) \in \mathbb{R}^4$, where $s(x) = 1/x$ when $x < r_{cs}$, $s(x) = 0$ when $x > r_c$, decaying smoothly between the two thresholds, N_m is the maximum length of all neighbor lists, j is the neighbor index of atom i.
2. Define the embedding net $\mathcal{G}_i \in \mathbb{R}^{N_m \times M}$, $\mathcal{G}_i = \mathcal{G}(s(|\mathbf{r}_{i\cdot}|))$, where M is called symmetry order, $\mathcal{G} = \mathcal{E}_2 \circ \mathcal{E}_1 \circ \mathcal{E}_0$, $\mathcal{E}_l(X) = X + \tanh\left(XW_l + \mathbf{1} \otimes w_l\right), l \in \{1, 2\}$, $\mathcal{E}_0(\mathbf{x}) = \tanh\left(\mathbf{x} \otimes W_0 + \mathbf{1} \otimes w_0\right)$, $|\mathbf{r}_{i\cdot}|$ and $\mathbf{1} \in \mathbb{R}^{N_m \times 1}, w_l \in \mathbb{R}^{1 \times M}, W_l \in \mathbb{R}^{N_m \times M}, l \in \{0, 1, 2\}$ and the functions s and $\tanh$ are element-wise.
3. $\tilde{\mathcal{R}}_i$ timing $\mathcal{G}_i$ yields the so-called descriptor $\mathcal{D}_i := \mathcal{G}_i^\intercal \tilde{\mathcal{R}}_i \tilde{\mathcal{R}}_i^\intercal \mathcal{G}_i^< \in \mathbb{R}^{M \times M^<}$, i.e. $\mathcal{G}_i^<$ is the several columns of $\mathcal{G}_i$ and $M^< < M$.
4. The fitting net is $E_i = \mathcal{F}(\mathcal{D}_i) = \mathcal{F}_3 \circ \mathcal{F}_2 \circ \mathcal{F}_1 \circ \mathcal{F}_0(\mathcal{D}_i)$, where $\mathcal{D}_i$ is reshaped into a vector of form $\mathbb{R}^{MM^<}$, $\mathcal{F}_l(\mathbf{x}) = \mathbf{x} + \tanh\left(\tilde{W}_l \mathbf{x} + \tilde{w}_l\right), \tilde{w}_l \in \mathbb{R}^{d \times 1}, \tilde{W}_l \in \mathbb{R}^{d \times d}, l \in \{1, 2\}$, $\mathcal{F}_3(\mathbf{x}) = \tilde{W}_3 \mathbf{x} + \tilde{w}_3$, $\tilde{w}_3 \in \mathbb{R}$, $\tilde{W}_3 \in \mathbb{R}^{1 \times d}$, $\mathcal{F}_0(\mathbf{x}) = \tanh\left(\tilde{W}_0 \mathbf{x} + \tilde{w}_0\right), \tilde{w}_0 \in \mathbb{R}^{d \times 1}$, $\tilde{W}_0 \in \mathbb{R}^{d \times MM^<}$.
5. The output $E := \sum_i E_i$, $F_i := -\nabla_{\mathbf{r}_i} E$.

EKF with Memory Factor for NNs. For neural networks,

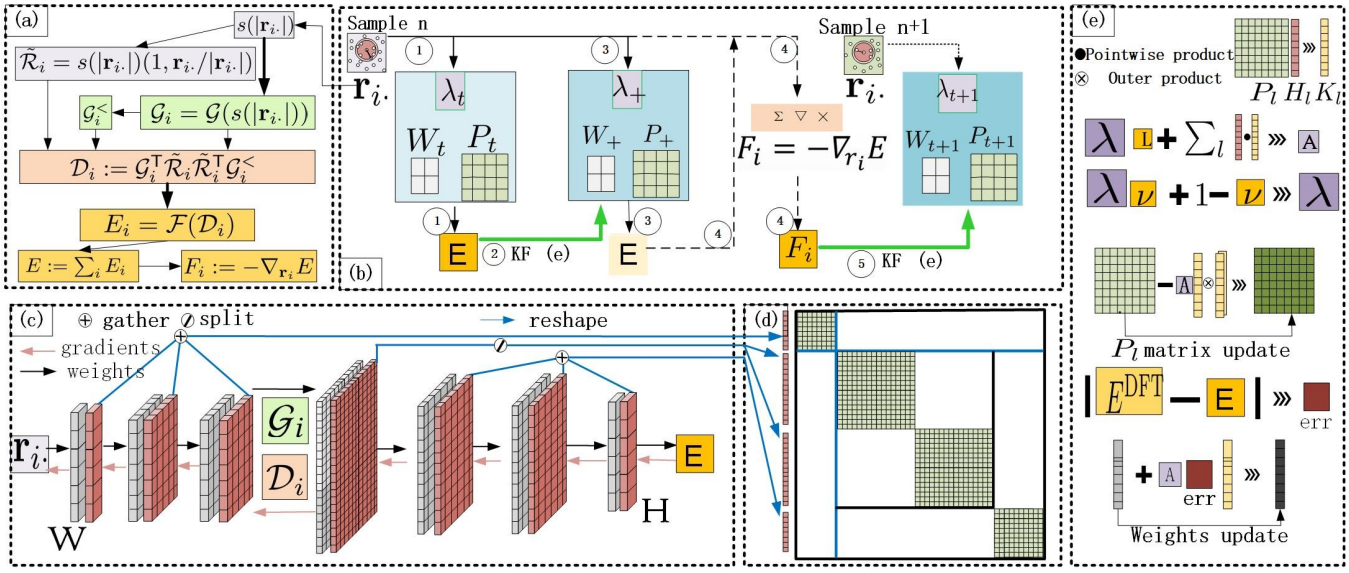

Figure 1: The overview of DP NN with RLEKF. (a) Structure of DP NN. The two arrows in bold correspond to the embedding net and the fitting net. (b) Macro-structure of training, ① input data into the network, ② calculate the gradient of energy with respect to weights through backpropagation and update w, P, λ with Kalman filter ③ feed data into the network for energy as an intermediate, ④ obtain force through the energy obtained in ③ based on a formula the arrow directs to, ⑤ derive the gradient of force with respect to weights through backpropagation and use Kalman filtering to update w, P, λ, and then the above progress for the next sample starts. (c) Weights and gradients flow of DP NN. The gather OP is implemented on layers when the total number of accumulated parameters is less than N_b, whereas the split OP is activated on a layer if the number of its parameters exceeds N_b. (The gather works on the first three layers in embedding net and the last two layers in fitting net, meanwhile the split works on the first layer in fitting net in the following experiment.) (d) Error covariance matrix and splitting strategy. This picture shows our splitting strategy which is applied to most of the following experiments. Here is an example: with the default block size set as N_b =5120 and 10240 for efficiency, except for the first layer of the fitting net, the number of weights of each layer, less than the default block size, so does not need to be split. The first fitting net layer containing 20000 weights will be split into 2 parts ($20000 = 1 \times 10240 + 1 \times 9760$) if $N_b = 10240$. (e) Updating strategy (Alg. 1).

the model of interest

$$\begin{cases} \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} = \boldsymbol{w}, \\ y_t = h(\boldsymbol{\theta}_t, x_t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, R_t), \end{cases} \tag{1}$$

is formulated in stochastic language as an EKF problem targeting on $\hat{\boldsymbol{\theta}}_t$, where $\boldsymbol{w}$ is the vector of all trainable parameters in the network $h(\cdot,\cdot)$, $\{(x_t, y_t)\}_{t\in\mathbb{N}}$ are pairs of feature and label, $\{y_t\}_{t\in\mathbb{N}}$ can also be seen as measurements of EKF, $\{\eta_t\}_{t\in\mathbb{N}}$ are noise terms subject to normal distribution with mean 0 and variances $\{R_t\}_{t\in\mathbb{N}}$ correspondingly, and $\forall t \in \mathbb{N}, \hat{\boldsymbol{\theta}}_{t|t-1} := \mathbb{E}[\boldsymbol{\theta}_t | y_{t-1}, y_{k-2}, \ldots, y_1]$. With fixed $\boldsymbol{\theta}_t$ and bounded x_t, $h(\boldsymbol{\theta}_t, x_t)$ will be approximated well by its linearization at $\hat{\boldsymbol{\theta}}_{t|t-1}$, just omitting a term $\mathcal{O}((\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_{t|t-1})^2)$.

$$y_t \approx h(\hat{\boldsymbol{\theta}}_{t|t-1}, x_t) + \mathbf{H}_t^\mathsf{T}(\boldsymbol{\theta}_t - \hat{\boldsymbol{\theta}}_{t|t-1}) + \eta_t \tag{2}$$

$$\mathbf{H}_t = \left.\frac{\partial h(\boldsymbol{\theta}, x_t)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{t|t-1}}$$

If set $m_t = y_t - h(\hat{\boldsymbol{\theta}}_{t|t-1}, x_t) + \mathbf{H}_t^\mathsf{T}\hat{\boldsymbol{\theta}}_{t|t-1}$ and rewrite (2) the following KF problem

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} = \boldsymbol{w}, \tag{3}$$

$$m_t \approx \mathbf{H}_t^\mathsf{T}\boldsymbol{\theta}_t + \eta_t. \tag{4}$$

At the beginning of training, the estimator $\hat{\boldsymbol{\theta}}_{t|t-1}$ is far away from $\boldsymbol{w}$, so less attention should be paid to those data fed to the network at an earlier stage of training than those at later stage. Through timing a factor $\alpha_t := \Pi_{i=1}^t \lambda_i^{-1/2}$ and $\alpha_0 := 1$, where $0 < \lambda_i \leq 1$ and $\lambda_i \to 1$, the last problem enjoys the better variant as below

$$\begin{cases} \boldsymbol{\theta}_t = \lambda_t^{-1/2}\boldsymbol{\theta}_{t-1}, \ \boldsymbol{\theta}_1 = \boldsymbol{w} \\ \tilde{m}_t = \alpha_t m_t \approx \mathbf{H}_t^\mathsf{T}\boldsymbol{\theta}_t + \alpha_t\eta_t = \mathbf{H}_t^\mathsf{T}\boldsymbol{\theta}_t + \tilde{\eta}_t, \end{cases} \tag{5}$$

where λ_t is called memory factor. The greater λ_t is, the more weight, or say attention, is paid to previous data. According to basic KF theory (Haykin and Haykin 2001) , we obtain

$$\begin{aligned} \mathbf{a}_t &= \lambda_t^{-1}\mathbf{H}_t^\mathsf{T}\mathbf{P}_{t-1}\mathbf{H}_t + \alpha_t^2 R_t, \\ \mathbf{K}_t &= \lambda_t^{-1}\mathbf{P}_{t-1}\mathbf{H}_t\mathbf{a}_t^{-1}, \\ \mathbf{P}_t &= \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}_t^\mathsf{T}\right)\lambda_t^{-1}\mathbf{P}_{t-1}, \\ \hat{\boldsymbol{\theta}}_t &= \hat{\boldsymbol{\theta}}_{t|t-1} + \mathbf{K}_t\tilde{\epsilon}_t, \\ \tilde{\epsilon}_t &= \tilde{m}_t - \mathbf{H}_t^\mathsf{T}\hat{\boldsymbol{\theta}}_{t|t-1} = \alpha_t(y_t - h(\alpha_t^{-1}\hat{\boldsymbol{\theta}}_{t|t-1}, x_t)). \end{aligned}$$

Finally, we recover the estimator of w via that of θ divided by the factor α_t, define $\forall t \in \mathbb{N}, \hat{\boldsymbol{w}}_{t|t-1} :=$

Algorithm 1: RLEKF($\hat{Y}, Y^{\text{DFT}}, \boldsymbol{w}^{in}, \mathbf{P}^{in}, \lambda^{in}$)

for $i = 1, 2, \ldots, length(Y^{\text{DFT}})$ **do**
  if $\hat{Y}_i \geq Y_i^{\text{DFT}}$ **then**
    $\hat{Y}_i$ = -$\hat{Y}_i$
  end if
end for
$\hat{Y} = \frac{\sum_i \hat{Y}_i}{length(Y^{\text{DFT}})}, errY = \frac{\sum_i |Y_i^{\text{DFT}} - \hat{Y}_i|}{length(Y^{\text{DFT}})}$
$H = \nabla_w \hat{Y} \mid_{w=\boldsymbol{w}^{in}}$
$a = 1/(L\lambda^{in} + H^{\mathsf{T}}\mathbf{P}^{in}H)$
for $l = 1, 2, \ldots, L$ **do**
  $K = \mathbf{P}_l^{in} \times H_l$
  $\mathbf{P}_l^{out} = (1/\lambda^{in}) \times (\mathbf{P}_l^{in} - aKK^{\mathsf{T}})$
  $\boldsymbol{w}_l^{out} = \boldsymbol{w}_l^{in} + aKerrY$
end for
$\lambda^{out} = \lambda^{in}\nu + 1 - \nu$
$\boldsymbol{w}^{out} = \boldsymbol{w}_l^{out} | l = 1, \ldots, L$
$\mathbf{P}^{out} = \mathbf{P}_l^{out} | l = 1, \ldots, L$
Output: $\boldsymbol{w}^{out}, \mathbf{P}^{out}, \lambda^{out}$

Algorithm 2: High-level Structure of Training with RLEKF

Input: $\{\boldsymbol{w}_0\}, \{\mathbf{P}_0\}, \{\lambda_1\}$
for $t = 0, 1, 2, \ldots, T-1$ **do**
  $\hat{E} = h_E(\boldsymbol{w}_t, x_t)$
  $\boldsymbol{w}_+, \mathbf{P}_+, \lambda_+ = \text{RLEKF}(\hat{E}, E_{DFT}, \boldsymbol{w}_t, \mathbf{P}_t, \lambda_t)$
  $\hat{F} = h_F(\boldsymbol{w}_+)$
  $\boldsymbol{w}_{t+1}, \mathbf{P}_{t+1}, \lambda_{t+1} = \text{RLEKF}(\hat{F}, F_{DFT}, \boldsymbol{w}_+, \mathbf{P}_+, \lambda_+)$
end for
Output: $\{\boldsymbol{w}_T\}, \{\mathbf{P}_T\}, \{\lambda_T\}$

$\alpha_t^{-1}\hat{\boldsymbol{\theta}}_{t|t-1}, \boldsymbol{w}_t := \alpha_t^{-1}\hat{\boldsymbol{\theta}}_t$, find $\hat{\boldsymbol{w}}_{t|t-1} = \boldsymbol{w}_{t-1}$, and then get our weights updating strategy

$$\epsilon_t = y_t - h(\boldsymbol{w}_{t-1}, x_t),$$
$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \mathbf{K}_t\epsilon_t.$$

Method

In this section, we introduce RLEKF and its splitting strategy (Fig. 1) and then compare RLEKF with GEKF.

The Workflow of Training with RLEKF. The overview of DP NN with RLEKF is shown in Fig. 1, and the corresponding training procedures are detailed in Alg. 2. Specifically, Alg. 2 and Fig. 1.b show the training of RLEKF for both energy (Alg. 2, line 2 & 3) and force (Alg. 2, line 4 & 5). The forward prediction(shown in Alg. 2 and Fig. 1.c), backward propagation (shown in Alg. 1 and Fig. 1.c), and updating of weights based on Kalman filtering with forgetting factor are shown in Alg. 1 (line 8 to line 14 and Fig. 1.e). Alg. 2 shows the iterative update of $\boldsymbol{w}_t$ with $\mathbf{P}_t, \lambda_t$ (also shown in Fig. 1.b).

After the initialization of $\boldsymbol{w}_0$, $\mathbf{P}_0$, and λ_1, the energy process starts and tries to fit the predicted energy to the energy label of a sample, and then the force process updates weights under the supervision of the samples' force label.

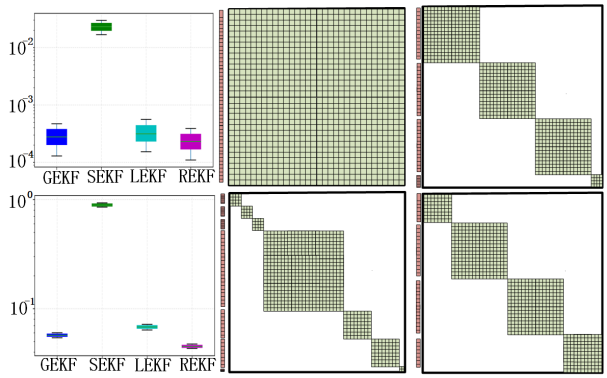

Figure 2: The two subfigures on the left are Energy (eV) (above) and Force (eV/Å) (below) RMSE boxplots of test dataset under different weights splitting strategy. The four subfigures on the right are weights error covariance matrixs corresponding to GEKF, SEKF, LEKF, and RLEKF in text order.

In the force process for each image, we randomly choose x atoms and concatenate their network-predicted force vectors as an effective predicted force $\hat{F}$, which is then used to update weights. This process (i.e. line 4 & 5 of Alg. 2) is repeated y times in a loop. Here, we recommend a relatively universal setting $x = 6, y = 4$. When these two kinds of alternative weights update in the direction of minimizing the trace of $\mathbf{P}_t$ for this sample, the same process for the next one repeats until time step T. The algorithm RLEKF in line 3 & 5 of Alg. 2 unfolds below.

RLEKF. Alg. 1 shows the layerwise weights-updating strategy of RLEKF, which is an optimized version of EKF in training the DP model. From line 1 to line 7, we transform any vector $\hat{Y}$ into a number, which enjoys two advantages. The first is it averages all the updating information in $Y_i^{\text{DFT}} - \hat{Y}_i$ and transforms a vector into a number which avoids an impregnable problem, solving inversion of overwhelmingly many big matrices introduced later. The second is once Alg. 1 invoked we only need to calculate the gradient for one time. More specifically, from line 1 to line 7, we just want to adjust the gradient of $\hat{Y}_i$ with respect to weights in the direction of "decreasing the difference between $\hat{Y}_i$ and Y_i^{DFT}". In order to keep $\mathbf{P}^{out}$ strictly symmetric if $\mathbf{P}^{in}$ is, the definition of Kalman gain in Alg. 1 is a little different from traditional Kalman filtering. In addition, $\mathbf{a}$ is usually a matrix, but in line 8 it degenerates into a number, which heavily reduces the computational complexity, where L is the number of blocks in $\mathbf{P}^{in}$ and take $\alpha_t^2 R_t = L\mathbf{I}$ as a suitable hyperparameter. RLEKF independently updates weights and $\mathbf{P}_t$ of different layers by KF theory and its splitting strategy introduced in the next subsection. Finally, all updated parameters are **collect**ed for the prediction of the next turn (Fig. 1.e, Alg. 1 from line 8 to line 16, where ν is forgetting rate, a hyperparameter describing the varying rate of λ_t and **split** is a function for obtaining the l-th part after splitting).

The Splitting Strategy of RLEKF. According to the split-

Systems	Structure	Time Step (fs)	# Snapshots	Energy(trn)	Energy(tst)	Force(trn)	Force(tst)
Cu	FCC	1	1646	**0.250**/0.451	**0.327**/0.442	40.6/**39.7**	45.2/**44.4**
Si	DC	2.5-3.5	3000	**0.148**/0.165	0.186/**0.181**	22.3/**21.9**	24.1/**23.4**
C	Graphene	2-3.5	4000	**0.0856**/0.133	**0.267**/0.278	**24.2**/27.6	**34.7**/35.5
Ag	FCC	2.5-3	2015	**0.142**/0.159	**0.243**/0.265	11.3/**11.0**	**12.4**/12.5
Mg	HCP	0.5-2	4000	**0.111**/0.160	**0.169**/0.189	**13.0**/14.8	**16.1**/17.1
NaCl	FCC	2-3.5	3193	**0.0403**/0.0631	**0.0435**/0.0514	6.06/**5.78**	6.69/**6.47**
Li	BCC,FCC,HCP	0.5	2494	**0.0482**/0.126	**0.312**/0.332	22.9/**14.0**	**24.8**/26.4
Al	FCC	2-3.5	4000	**0.359**/0.534	**0.473**/1.24	**54.1**/56.3	**58.6**/55.8
MgAlCu	Alloy	3	2530	**0.165**/0.227	**0.218**/0.233	44.1/**37.6**	45.2/**42.6**
H_2O	Liquid	0.5	4000	**0.297**/0.584	**0.545**/1.00	60.8/**20.6**	**68.1**/87.6
S	S_8	3-5	7000	**0.210**/0.401	0.628/0.628	51.5/**45.9**	60.4/**58.5**
CuO	FCC	3	1000	**2.203**/2.47	**2.04**/3.78	424/**414**	442/**438**
Cu+C	SA	3-3.2	2000	**0.458**/1.27	**1.07**/2.52	176/**143**	**190**/195

Table 1: Structures, the quantities of snapshots, time steps (frequency of yielding snapshots), and the root mean square errors (RMSE) of the training and the testing of RLEKF (before slashes) and Adam (after slashes) in terms of energy (meV) and forces (meV/Å) after 30 epochs for various systems. The RMSEs of the energies are normalized by the number of atoms in the system. For all sub-systems, the former 80% and the latter 20% of the snapshot dataset are used for training and testing, respectively. Better results are in bold. Acronyms: FCC (face-centered cubic), BCC (body-centered cubic), HCP (hexagonal close-packed), DC (diamond cubic), SA (surface adsorption).

ting strategy of RLEKF, GEKF can be approximated by RLEKF with a reduction in weights error covariance matrix considering the efficiency and stability of a large-scale application. Unlike GEKF, the weights error covariance matrix of whichever time step, up to a scalar factor, $\mathbf{P} = \{P_1, \ldots, P_L\}$ is a $N \times N$ block diagonal matrix, whose shape is $\{n_1 \times n_1, n_2 \times n_2, \ldots, n_L \times n_L\}$, where n_i and $N := \sum_i n_i$ are the number of weights of the ith block and that of the whole NN respectively. This means some weights error covariances are forced into 0 and $\mathbf{P}$ thus is no longer the real weights error covariance matrix at most an approximation, but fortunately there are indeed such good enough approximations that they carry most of the "big values" and information in these diagonal blocks, on which $\mathbf{P}$ concentrates. As shown in Fig. 2, GEKF concerns correlations between all the parameters. However, it is computationally expensive when adapted to large NNs. As a seemingly good choice, SEKF is a load balanced approximation version of GEKF, but the heavy correlations between parameters in a layer are ignored inappropriately. Overcoming the drawback of SEKF, LEKF can fully consider the internal correlation between parameters in the same layer, whereas unbearable computation still confronts us if the layer contains tens of thousands of parameters and decoupling every small neighboring layer deviates from the load balance idea. Here, we induce two heuristic principles for choosing a suitable strategy:

1. Put weights with high correlation (in the same layer) in the same block if possible.
2. The block must not be too large since that burdens computers with much computation (splitting the layers if the number of parameters is larger than the threshold N_b) or too small since that wastes computational power and loses massive information (gathering the near neighboring small layers). The threshold N_b aims at splitting the matrix $\mathbf{P}$ as evenly as possible for load balance and linear computational complexity.

Therefore, following these principles, we induce the splitting strategy of RLEKF (Fig. 1.d) and reorganize the weights in different layers, named parameter parts, into several more appropriate layers. Trainable parameters could be decomposed into a series of l the parts of size $p_1, p_2, \ldots, p_l$. Starting from the first part of size p_1, we execute the commands below repeatedly until $z = l$:

1. For a given part with p_s weights, we split them into $\lfloor \frac{p_s}{N_b} \rfloor$ layers of size N_b and a new part of size $p_s \leftarrow p_s - N_b \lfloor \frac{p_s}{N_b} \rfloor$.
2. Then consider subsequent parts with $p_s, p_{s+1}, \ldots, p_z$ weights, gather them together into a layer of size $\sum_{i=s}^{z} p_i$ and then set $s \leftarrow z+1$, if $\sum_{i=s}^{z} p_i \leq N_b$ and $(\sum_{i=s}^{z+1} p_i > N_b$ or $z = l)$.

Empirically, Fig. 2 validates the efficiency of RLEKF, comparably or even more accurate than GEKF.

Theoretical Analysis

In this section, we prove the convergence of weights updating and the stability of the training process (avoid gradient exploding). For simplicity, we analyze GEFK case which has the same asymptotic feature as RLEKF (Witkoskie and Doren 2005).

Theorem 1 *In EKF problem* (5)*, setting* $\alpha_t^2 R_t = L\mathbf{I}$*, assuming components of* $\mathbf{H}_i$ *are independent and subject to identical distribution with mean 0 and variance* σ^2*, we have*

$$\epsilon_t \mathbf{K}_t \sim \mathcal{O}(\frac{1}{t})$$

with the probability arbitrarily close to 1 when $t \to \infty$*, which means the convergence of weights updating and thus the algorithm avoidance of gradient exploding.*

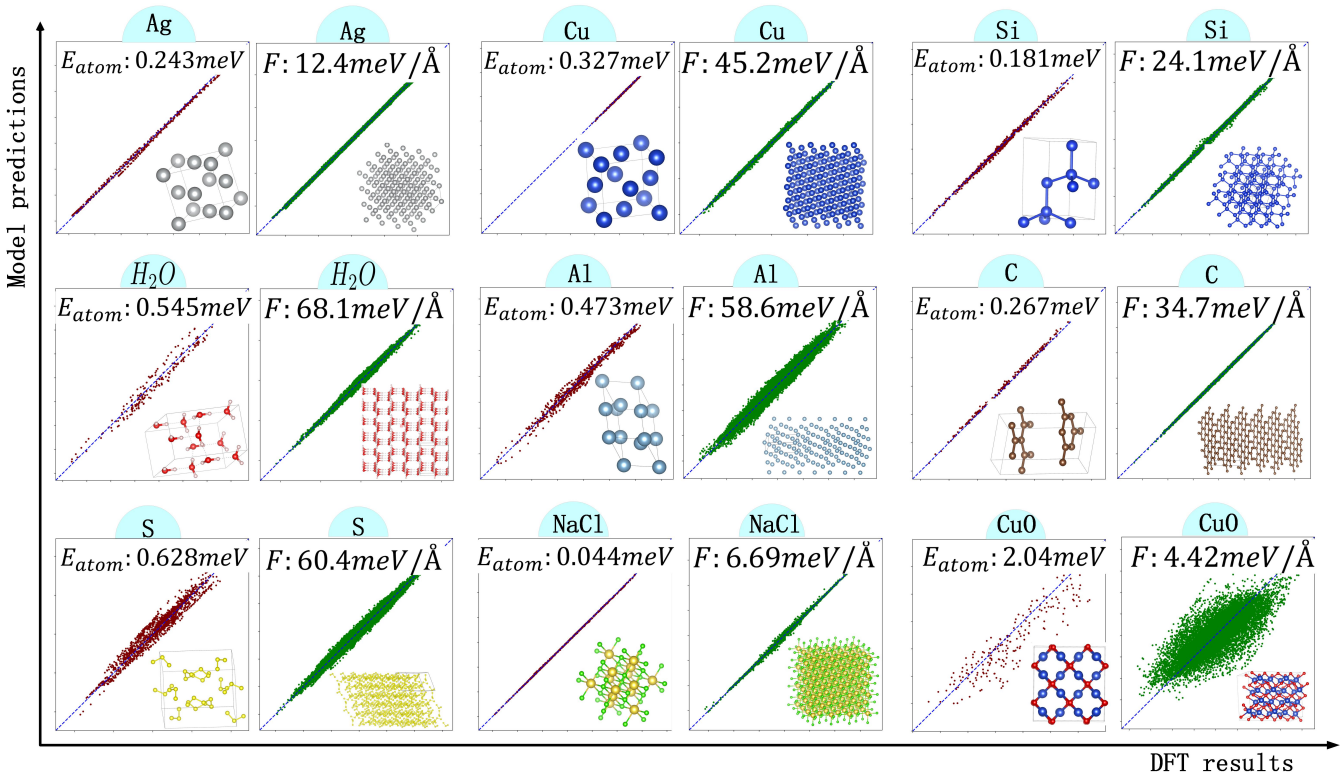

Figure 3: RLEKF performance contrasted with DFT (ground truth) on every snapshot in terms of energy (meV) and force (meV/Å) for various systems. That is to say, the closer the dots approach diagonals, the better the performance is. RMSE of E_{tot}, E_{atom}, and F on the test set, the structure of a unit cell, and extensive structure are shown in the corners of each subfigure.

The following is a brief proof of Theorem 1. Based on basic KF theory, we obtain

$$\begin{aligned}
\mathbf{P}_t &= \lambda_t^{-1}\mathbf{P}_{t-1} - \lambda_t^{-2}\mathbf{P}_{t-1}\mathbf{H}_t^{\intercal}\mathbf{a}_t^{-1}\mathbf{H}_t\mathbf{P}_{t-1}L^{-1},\\
\mathbf{a}_t &= \lambda_t^{-1}\mathbf{H}_t^{\intercal}\mathbf{P}_{t-1}\mathbf{H}_t L^{-1} + 1 = \mathbb{E}[\epsilon_t^2]\alpha_t^2,\\
\lambda_t &= 1-(1-\lambda_1)\nu^{t-1},\\
\mathbf{P}_t &= \mathbb{E}[\tilde{\boldsymbol{w}}_t\tilde{\boldsymbol{w}}_t^{\intercal}]\alpha_t^2,
\end{aligned}$$

and

$$\mathbf{P}_t^{-1} = \lambda_t\mathbf{P}_{t-1}^{-1} + \mathbf{H}_t\mathbf{H}_t^{\intercal}L^{-1}$$

by using Woodbury matrix identity, where $\tilde{\boldsymbol{w}}_t = \boldsymbol{w} - \boldsymbol{w}_t$, weights error covariance matrix $\mathbb{E}[\tilde{\boldsymbol{w}}_t\tilde{\boldsymbol{w}}_t^{\intercal}] = (\mathbf{P}_0^{-1} + \sum_{i=1}^{t}\alpha_i^2\mathbf{H}_i\mathbf{H}_i^{\intercal}L^{-1})^{-1}$. In the following experiments, we assume components of $\mathbf{H}_i$ are independent and subject to identical distribution with mean 0 and variance σ^2. So, the covariance matrix of $\mathbf{H}_i$ is $\sigma^2\mathbf{I}$ and $\mathbf{P}_0 = \mathbf{I}$. Hence

$$\mathbb{E}[\mathbf{P}_t^{-1}] = \mathbf{I} + \sum_{k=1}^{t}\alpha_k^2\alpha_t^{-2}L^{-1}\sigma^2\mathbf{I}.$$

Using q-Pochhammer symbol, we find

$$\lim_{t\to\infty}\alpha_t^2 = \Pi_{i=0}^{\infty}(1-(1-\lambda_1)\nu^i) = ((1-\lambda_1);\nu)_{\infty} = \alpha$$

exists. Therefore, $S(t) := \sum_{k=1}^{t}\alpha_k^2\alpha_t^{-2}$ of order $\mathcal{O}(t)$. According to the law of large numbers, we get

$$\lim_{t\to\infty}\frac{\mathbf{P}_t^{-1}}{S(t)} \xrightarrow{a.s.} \sigma^2 L^{-1}\mathbf{I},$$

i.e.

$$\lim_{t\to\infty}\mathbf{P}_t \overset{a.s.}{=\!=} \lim_{t\to\infty}\frac{L\mathbf{I}}{\sigma^2 S(t)}.$$

Hence,

$$\mathbf{K}_t \overset{a.s.}{\sim} \mathcal{O}(\frac{1}{t}),$$

if t is large enough. Further, ϵ_t is bounded with a probability

$$\mathbb{P}(|\epsilon_t| \le B) \ge 1 - \frac{\mathbb{E}[\epsilon_t^2]}{B^2} = 1 - \frac{\alpha_t^{-2}(\lambda_t^{-1}\mathbf{H}_t^{\intercal}\mathbf{P}_{t-1}\mathbf{H}_t + 1)}{B^2},$$

arbitrarily close to 1, where Markov's inequality is used, if initialization $\boldsymbol{w}_0$ is close enough to some local minimum $\boldsymbol{w}^*$ of the landscape for Taylor approximation (1). Therefore,

$$\epsilon_t\mathbf{K}_t \sim \mathcal{O}(\frac{1}{t})$$

with the probability arbitrarily close to 1.

Experiments and Results

Bulk systems are challenging AIMD tasks due to their extensiveness (periodic boundary condition) and complexity

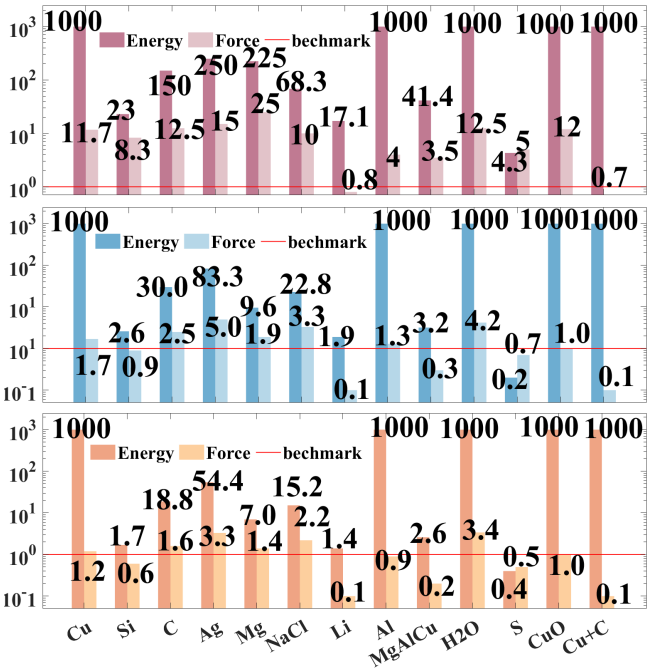

Figure 4: Speed ratio of RLEKF's reaching an accuracy baseline to Adam's in terms of Energy and Force according to epoch, iteration(weights updating), and wall-clock time, where 1000 means Adam can not reach the baseline. The baseline is 1.2× the lower RMSE between Adam and RLEKF on the test datasets in Tab. 1.

(many different phases and atomic components). Our experiment is conducted on several representative problematic bulk systems. They are simple metal (Cu, Ag, Mg, Al, Li), alloy (MgAlCu), nonmetal (S, C), semiconductor(Si), simple compound (H_2O), electrolyte (NaCl), and some challenging systems (CuO, Cu+C). The effectiveness of RLEKF is shown by a comparison with the SOTA benchmark Adam in terms of the RMSEs of predicted total energy of the whole system E_{tot}, and forces $\{(F_{x,i}, F_{y,i}, F_{z,i})|i = 1, \ldots, n\}$, where x, y, z correspond to different directions of Cartesian coordinate system and i is the index of atom (Tab. 1).

Experiment Setting. Set $\lambda_1 = 0.98$, $\mathbf{P}_0 = \mathbf{I}$, $\nu = 0.9987$, $\boldsymbol{w}_0$ consistent with DeePMD-kit (Wang et al. 2018). The network configuration is [1, 25, 25, 25] (embedding net), [400, 50, 50, 50,1] (fitting net).

Data Description. We choose 13 unbiased and representative datasets of the aforementioned systems with certain specific structures (second column of Tab. 1). For each dataset, snapshots are yielded based on solving *ab initio* molecular trajectories via PWmat (Jia et al. 2013). During this process, to enlarge the sampling span of configuration space, we fast generate a long sequence of the snapshot by small time step (the third column of Tab. 1) and choose one for every fixed number in the temperature 300K.

Main Results. Both RLEKF and Adam yield good results except for CuO and Cu+C systems (Tab. 1). From an **accuracy** perspective, RLEKF reaches a higher energy accuracy in 12 cases except for Si, little less accurate than Adam. As for force, RLEKF is more accurate than Adam in 7 cases while the reminder achieves a very comparable precision. Furthermore, Fig. 3 shows how far the fitting results of RLEKF deviate from those of DFT (ground truth). From a **speed** perspective, generally, RLEKF converges to a reasonable RMSE much faster than Adam in most of the 13 cases in terms of energy (Fig. 4). We take bulk copper as an example to demonstrate how to understand information in Fig. 4. The lower Energy and Force RMSE between Adam and RLEKF is 0.327 meV and 44.4 meV/$\mathring{A}$.

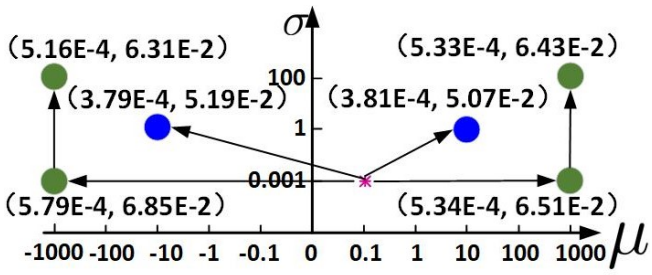

Figure 5: Accuracy (energy and force RMSE on a test dataset) of Cu models with different skip connection weights initialization $\mathcal{N}(\mu, \sigma^2)$. The red star represents the setup adopted in the above experiments, to whose result Fig. 1 refers. The 3 blue dots and the red star show similar accuracy, the 4 green dots show lower precision but are still very reasonable.

Robustness Analysis: Distribution of Hyperparameter of Weights Initialization. RLEKF is very stable as an optimizer, which can keep NNs from gradients exploding and consequently endow them with very loose initialization constraints almost up to none as shown in Fig. 5.

Computational Complexity. RLEKF also benefits from computing through reducing the computation compared to GEKF. There are 3 computational intensive parts in Alg. 1, calculating a (line 8), K (line 10), and P^{out} (line 11). Due to the even splitting strategy, the order of float operation for each block is $\mathcal{O}(N_b^2)$, and the number of the block is of order $\mathcal{O}(N/N_b)$. Hence the total computational complexity of RLEKF is of order $\mathcal{O}((N/N_b)N_b^2) = \mathcal{O}(NN_b)$.

Conclusions

We proposed an optimizer RLEKF on DP NN and tested RLEKF on several typical bulk systems (simple metals, insulators, and semiconductors) of diverse structure (FCC, BCC, HCP). RLEKF defeated SOTA benchmark Adam by 11-1 (7-6) in precision and 12-1 (7-6) in wall-clock time speed in terms of energy (force). Besides, the convergence of weights updating is proved theoretically and RLEKF presents robustness on weights initialization. To sum up, RLEKF is an accurate, efficient, and stable optimizer, which paves another path to training general large NNs.

Acknowledgments

This work is supported by the following funding: National Key Research and Development Program of China (2021YFB0300600), National Science Foundation of China (92270206, T2125013, 62032023, 61972377), CAS Project for Young Scientists in Basic Research (YSBR-005) and Network Information Project of Chinese Academy of Sciences (CASWX2021SF-0103), the Key Research Program of the Chinese Academy of Sciences grant No. ZDBS-SSW-WHC002, Soft Science Research Project of Guangdong Province (No. 2017B030301013), and Huawei Technologies Co., Ltd.. We thank Dr. Haibo Li for helpful discussions.

References

Bartók, A. P.; Payne, M. C.; Kondor, R.; and Csányi, G. 2010. Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons. *Physical review letters*, 104(13): 136403.

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

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

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

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

Behler, J.; and Parrinello, M. 2007. Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces. *Phys. Rev. Lett.*, 98: 146401.

Chui, C. K.; Chen, G.; et al. 2017. *Kalman filtering*. Springer.

Desai, S.; Reeve, S. T.; and Belak, J. F. 2022. Implementing a neural network interatomic model with performance portability for emerging exascale architectures. *Computer Physics Communications*, 270: 108156.

Drautz, R. 2019. Atomic cluster expansion for accurate and transferable interatomic potentials. *Phys. Rev. B*, 99: 014104.

Ercolessi, F.; and Adams, J. B. 1992. Interatomic potentials from first-principles calculations. *MRS Online Proceedings Library (OPL)*, 291.

Ercolessi, F.; and Adams, J. B. 1994. Interatomic potentials from first-principles calculations: the force-matching method. *EPL (Europhysics Letters)*, 26(8): 583.

Gasteiger, J.; Giri, S.; Margraf, J. T.; and Günnemann, S. 2020. Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules. In *Machine Learning for Molecules Workshop, NeurIPS*.

Gasteiger, J.; Groß, J.; and Günnemann, S. 2020. Directional Message Passing for Molecular Graphs. In *International Conference on Learning Representations (ICLR)*.

Guo, Z.; Lu, D.; Yan, Y.; Hu, S.; Liu, R.; Tan, G.; Sun, N.; Jiang, W.; Liu, L.; Chen, Y.; Zhang, L.; Chen, M.; Wang, H.; and Jia, W. 2022. Extending the Limit of Molecular Dynamics with Ab Initio Accuracy to 10 Billion Atoms. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '22, 205–218. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392044.

Haykin, S. S.; and Haykin, S. S. 2001. *Kalman filtering and neural networks*, volume 284. Wiley Online Library.

Heimes, F. 1998. 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, 1639–1644 vol.2.

Huang, Y.-P.; Xia, Y.; Yang, L.; Wei, J.; Yang, Y. I.; and Gao, Y. Q. 2022. SPONGE: A GPU-Accelerated Molecular Dynamics Package with Enhanced Sampling and AI-Driven Algorithms. *Chinese Journal of Chemistry*, 40(1): 160–168.

Jia, W.; Fu, J.; Cao, Z.; Wang, L.; Chi, X.; Gao, W.; and Wang, L.-W. 2013. Fast plane wave density functional theory molecular dynamics calculations on multi-GPU machines. *Journal of Computational Physics*, 251: 102–115.

Jia, W.; Wang, H.; Chen, M.; Lu, D.; Lin, L.; Car, R.; Weinan, E.; and Zhang, L. 2020. 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*, 1–14.

Kalman, R. E.; et al. 1960. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2): 102–119.

Karplus, M.; and Kuriyan, J. 2005. Molecular dynamics and protein function. *Proceedings of the National Academy of Sciences*, 102(19): 6679–6685.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lee, K.; Yoo, D.; Jeong, W.; and Han, S. 2019. SIMPLE-NN: An efficient package for training and executing neural-network interatomic potentials. *Computer Physics Communications*, 242: 95–103.

Lysogorskiy, Y.; Oord, C. v. d.; Bochkarev, A.; Menon, S.; Rinaldi, M.; Hammerschmidt, T.; Mrovec, M.; Thompson, A.; Csányi, G.; Ortner, C.; and Drautz, R. 2021. Performant implementation of the atomic cluster expansion (PACE) and application to copper and silicon. *npj Computational Materials*, 7(1): 97.

Meier, K.; Laino, T.; and Curioni, A. 2014. Solid-state electrolytes: revealing the mechanisms of Li-ion conduction in tetragonal and cubic LLZO by first-principles calculations. *The Journal of Physical Chemistry C*, 118(13): 6668–6679.

Murtuza, S.; and Chorian, S. 1994. Node decoupled extended Kalman filter based learning algorithm for neural networks. In *Proceedings of 1994 9th IEEE International Symposium on Intelligent Control*, 364–369.

Rahman, A.; and Stillinger, F. H. 1971. Molecular dynamics study of liquid water. *The Journal of Chemical Physics*, 55(7): 3336–3359.

Raty, J.-Y.; Gygi, F.; and Galli, G. 2005. Growth of carbon nanotubes on metal nanoparticles: a microscopic mechanism from ab initio molecular dynamics simulations. *Physical review letters*, 95(9): 096103.

Saad, D. 1998. Online algorithms and stochastic approximations. *Online Learning*, 5(3): 6.

Singraber, A.; Morawietz, T.; Behler, J.; and Dellago, C. 2019. Parallel multistream training of high-dimensional neural network potentials. *Journal of chemical theory and computation*, 15(5): 3075–3092.

Smith, G. L.; Schmidt, S. F.; and McGee, L. A. 1962. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration.

Stillinger, F. H.; and Rahman, A. 1974. Improved simulation of liquid water by molecular dynamics. *The Journal of Chemical Physics*, 60(4): 1545–1557.

Thompson, A.; Swiler, L.; Trott, C.; Foiles, S.; and Tucker, G. 2015. Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials. *Journal of Computational Physics*, 285: 316–330.

Wang, H.; Zhang, L.; Han, J.; and Weinan, E. 2018. DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, 228: 178–184.

Wang, S.; Kramer, M.; Xu, M.; Wu, S.; Hao, S.; Sordelet, D.; Ho, K.; and Wang, C. 2009. Experimental and ab initio molecular dynamics simulation studies of liquid Al 60 Cu 40 alloy. *Physical Review B*, 79(14): 144205.

Witkoskie, J. B.; and Doren, D. J. 2005. Neural Network Models of Potential Energy Surfaces:Prototypical Examples. *Journal of Chemical Theory and Computation*, 1(1): 14–23. PMID: 26641111.

Yao, K.; Herr, J. E.; Brown, S. N.; and Parkhill, J. 2017. Intrinsic Bond Energies from a Bonds-in-Molecules Neural Network. *The Journal of Physical Chemistry Letters*, 8(12): 2689–2694. PMID: 28573865.