Computational Materials Science 244 (2024) 113155

Contents lists available at ScienceDirect

Computational Materials Science

journal homepage: www.elsevier.com/locate/commatsci

Full length article

Adaptive loss weighting for machine learning interatomic potentials

Daniel Ocampo [a], Daniela Posso [b], Reza Namakian [a], Wei Gao [a,c,*]

[a] J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843, United States
[b] Department of Mechanical Engineering, The University of Texas at San Antonio, San Antonio, TX 78249, United States
[c] Department of Materials Science & Engineering, Texas A&M University, College Station, TX 77843, United States

ARTICLE INFO

Dataset link: https://gao-group.github.io/atomdnn/index.html

Keywords:
Machine learning
Interatomic potentials
Adaptive loss weighting
Loss function
Neural network

ABSTRACT

Training machine learning interatomic potentials often requires optimizing a loss function composed of three variables: potential energies, forces, and stress. The contribution of each variable to the total loss is typically weighted using fixed coefficients. Identifying these coefficients usually relies on iterative or heuristic methods, which may yield sub-optimal results. To address this issue, we propose an adaptive loss weighting algorithm that automatically adjusts the loss weights of these variables during the training of potentials, dynamically adapting to the characteristics of the training dataset. The comparative analysis of models trained with fixed and adaptive loss weights demonstrates that the adaptive method not only achieves more balanced predictions across the three variables but also improves overall prediction accuracy.

1. Introduction

Atomistic simulations employing empirical interatomic potentials have been widely used in materials modeling. Although faster than simulations using density functional theory (DFT), they often yield less accurate results. A key limitation of traditional interatomic potentials is their fixed functional forms and few fitting parameters. By contrast, machine learning interatomic potentials (ML-IAPs) offer greater flexibility by learning the energy surface shape from training datasets. Therefore, if the training datasets cover sufficient physics of interests, well-trained ML-IAPs can provide accuracy comparable to DFT. While ML-IAPs generally have higher computational costs than traditional interatomic potentials, they effectively balance accuracy and efficiency. There have seen many successful examples of ML-IAPs developed for various material systems [1–4].

ML-IAPs can be broadly split into two types. The first is descriptor-based ML-IAP, in which the descriptors (or fingerprints) are used to describe the environment of the atoms in a system. Various descriptors have been proposed in the literature, such as Atom-Centered Symmetry Functions (ACSF) [5], Smooth Overlap of Atomic Positions (SOAP), Atomic Cluster Expansion (ACE) [6], and Moment Tensor Potentials [7], among others. A comprehensive review of the descriptors can be found in Musil et al.'s work [8]. Representative descriptor-based ML-IAPs include: Behler and Parrinello Neural Network potential [5,9], Gaussian approximation potential (GAP) [10], Spectral Neighbor Analysis Potential (SNAP) [11], Moment Tensor Potential (MTP) [7], Performant implementation of the atomic cluster expansion (PACE) [12], and DeePMD [13] among others. The second type of ML-IAP is the end-to-end potential, which operate differently by learning directly from the types and positions of atoms, without the need for predefined descriptors. Representative ones include: Crystal Graph Convolutional Neural Networks (CGCNN) [14,15], SchNet [16], and MatErials Graph Network (MEGNet) [17] among others. Although the end-to-end ML-IAPs leverage more recent and advanced feature learning AI technology, there is currently no conclusive evidence to suggest that end-to-end ML-IAPs outperform the descriptor-based ML-IAP in terms of prediction accuracy. The study reported in this paper are performed using ACSF.

The training of most ML-IAPs involves minimizing a loss function, which measures the difference between the predicted outputs of the potential and the actual target value obtained from Ab initio simulations. Typically, a ML-IAP's loss function comprises three components: potential energy, atomic forces, and stress tensor, each weighted by a prefactor (i.e. loss weight). Most of current ML-IAPs assign a predefined loss weight to each component, which stays as a constant throughout the training process [7,9,11,12]. Approaches like DeepMD modulate these weights linearly during training, though the rationale and effectiveness of this approach are not fully clear [13,18]. In this paper, our study demonstrates that varying combinations of loss weights significantly impact model performance, raising a key question: what constitutes an effective combination of loss weights that balances energy, force, and stress predictions? Our hypothesis is that a universally optimal set of loss weights for all ML-IAPs may not exist, as the optimal weights are likely tied to the material system and the characteristics of

* Corresponding author at: J. Mike Walker '66 Department of Mechanical Engineering, Texas A&M University, College Station, TX 77843, United States.
E-mail address: wei.gao@tamu.edu (W. Gao).

https://doi.org/10.1016/j.commatsci.2024.113155
Received 26 March 2024; Received in revised form 24 May 2024; Accepted 3 June 2024
Available online 20 June 2024

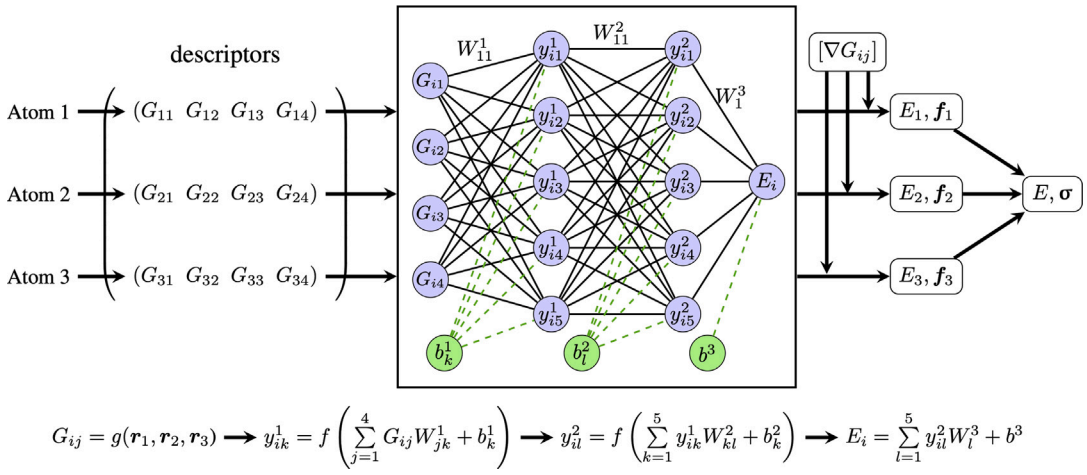

Fig. 1. Schematic of neural network interatomic potential where the potential energy, forces, and stresses are calculated using a feed-forward, descriptor-based neural network.

individual training datasets. Instead, we propose a new method that automatically adjusts the loss weights during the training of potentials. This approach dynamically adapts to the characteristics of the training dataset and optimizes ML-IAPs predictions.

The paper is structured as follows: first, we provide an overview of ML-IAPs based on neural networks, including the formulation of loss functions. Next, we introduce the principle and algorithm of adaptive loss weighting. Then, the results from models using adaptive methods are compared with those from models using fixed loss weights, in order to demonstrate the advantage of the adaptive method. Finally, the paper is concluded with a summary of main findings.

2. Computation methods

The adaptive loss weighting algorithm can be applied to a wide range of ML-IAPs that require minimizing a loss function. In this study, to demonstrate its applicability, we implemented it within the framework of the Behler and Parrinello type of neural network potential [5,9] inside an open-source package AtomDNN [19] developed by the authors.

2.1. Neural network potentials

In a descriptor-based neural network potential designed for a system with N atoms, the Cartesian coordinates of each atom i are denoted as $\boldsymbol{r}_i = \{r^{(1)}, \ldots, r^{(N)}\}$, which are transformed into a descriptor vector $\boldsymbol{G}_{ij}$ with M components, where $j = 1, \ldots, M$. While many descriptor types can be integrated with neural networks, in this study, atom-centered symmetry functions (ACSF) are utilized. The working principle of the neural network potential can be explained in Fig. 1, where a scenario with three atoms of the same chemical species is considered. These atoms are represented by 4 component descriptors, which are fed to the neural network as inputs. This network consists of two densely connected layers, each containing 5 neurons, and concludes with a linear concatenation layer. The network can be considered as a high dimensional non-linear function mapping, parametrized with weight matrices and bias vectors which are optimized during training. The equations in the figure constitute a forward-pass through the network, where W^1_{jk}, W^2_{kl} and W^3_l are the weight matrices of layers 1, 2 and, 3, respectively, b^1_k, b^2_l, and b^3 are the corresponding biases, and f is the activation function.

In this model, each atom of the same chemical species is processed through an identical neural network, yielding a per-atom energy E_i. In systems comprising multiple chemical species, separate parallel networks are employed for each species. The total potential energy of the system, $\mathcal{E}$, is the sum of per-atom energies, which can be written in terms of descriptors as

$$\mathcal{E} = \sum_i^N E_i = \sum_i^N E_i(G_{i1}, G_{i2}, \ldots, G_{iM}), \tag{1}$$

where i is the index for individual atom that is described by a feature vector with M components. The atomic forces can be obtained from the negative gradients of the total potential energy with respect to the atomic positions, which can be expressed in terms of the derivatives of descriptors using a chain rule

$$f_{j\alpha} = -\frac{\partial \mathcal{E}}{\partial r_{j\alpha}} = -\sum_{i=1}^{N}\sum_{m=1}^{M} \frac{\partial E_i}{\partial G_{im}} \frac{\partial G_{im}}{\partial r_{j\alpha}}, \tag{2}$$

where $r_{j\alpha} (\alpha = 1, 2, 3)$ are the Cartesian coordinates of the jth atom. The derivatives of descriptors, $\partial G_{im}/\partial r_{j\alpha}$, are fed to the network as additional inputs (represented by ∇G_{ij} in Fig. 1). The other derivative term, $\partial E_i/\partial G_{im}$, is readily available from the back-propagation algorithm used during the training process. In addition, for solid-state materials, Cauchy stress tensor can be used for training ML-IAPs, which can be written as

$$\sigma_{\alpha\beta} = \frac{1}{V}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{j\in \mathrm{NB}_i} \frac{\partial E_i}{\partial G_{im}} \frac{\partial G_{im}}{\partial r_{j\alpha}} r_{j\beta}, \tag{3}$$

where V is the current volume and NB_i represents the neighbor list of atom i withing the cutoff distance r_c. The derivation of Eq. (3) can be found in Appendix.

2.2. Loss function of ML-IAPs

The loss function for training neural network potential as well as many other types of ML-IAPs can be written as Eq. (4), which is composed of three components: potential energy, forces, and stresses.

$$\mathcal{L} = \frac{\alpha_1}{N}\sum_{i=1}^{N}\left(\hat{E}_i - E_i\right)^2 + \frac{\alpha_2}{3N}\sum_{i=1}^{N}\sum_{\alpha=0}^{3}\left(\hat{f}_{i\alpha} - f_{i\alpha}\right)^2 + \frac{\alpha_3}{6}\sum_{j=1}^{6}\left(\hat{\sigma}_j - \sigma_j\right)^2, \tag{4}$$

where α_1, α_2, and α_3 are the weight coefficients to balance each component towards the calculation of the total loss. The variables with hat refer to the true values from DFT calculations, and the Voigt notation is applied to stress components. Previous studies have indicated that training of ML-IAP with both energy and force is beneficial as it integrates data from potential energies and their gradients, leading to more stable predictions and a reduction in the quantity of data structures needed for effective training [20,21]. The loss term for stresses has also been shown to be beneficial in promoting transferability of the ML-IAP [22]. However, the specific impact of incorporating stress in the

loss has not been thoroughly investigated. It is also noteworthy that most prior studies have limited their focus to only the potential energy and force components in training, which could lead to unsatisfactory stress predictions, as our later computation results will show.

In previous studies, weight coefficients for ML-IAPs were typically kept constant during training. For instance, in a recent study, a series of ACE-based potentials were trained using various energy and force weight combinations on a copper dataset. The findings suggested that an $0.8:0.2$ ratio of potential energy to force loss weight was optimal for performance [23]. In anther study, the ACE-based potential was trained by assigning varied loss weight coefficients to different subsets of the training data, though the methodology for selecting these parameters was not detailed [12]. Moreover, instead of fixed weight coefficients, a different approach was taken in training a neural network potential using DeepMD-kit. This method uses a monotonic decrease in the force weight while increasing the energy weight during the training of a dataset containing pure Silica zeolite structures [24]. However, the effectiveness of this particular training strategy was not detailed.

2.3. Adaptive loss weighting

In this study, we propose to dynamically adjust loss weights during the potential training process. This concept is inspired by the work of Heydari et al. [25], who developed an adaptive loss weighting algorithm for a convolutional neural network for image reconstruction and synthetic data generation. Similar to their strategy, our method was designed to manage multipart loss functions by dynamically altering each loss weight. This is accomplished by continuously recalibrating the loss weights based on the change in loss values from one epoch to the next, ensuring a balanced optimization process without any single component becoming overly dominant. The advantages of this approach are twofold: it avoids the scenarios for certain loss components to disproportionately impact the training process and eliminates the need for manually determining the optimal fixed weight combination each time the material system or dataset changes.

The proposed adaptive method is presented in Algorithm 1. To illustrate the machinery of the algorithm, we first re-write Eq. (4) as

$$\mathcal{L}_i = \sum_{k=1}^{3} \alpha_k^i \ell_k, \tag{5}$$

where ℓ_1, ℓ_2, and ℓ_3 are the loss components for potential energy, forces, and stresses, respectively. The weight factor α_k is calculated at the ith epoch by

$$\alpha_k^i = \frac{e^{\beta \cdot s_k^i}}{\sum_{m=1}^{3} e^{\beta \cdot s_m^i}}, \tag{6}$$

where

$$s_k^i = \ell_k^i - \ell_k^{i-1} \tag{7}$$

represents the rate of change of k-th loss component, scaled by a hyperparameter β. Eq. (6) adopts the classic Softmax function, and therefore, the algorithm is referred to as *Softadapt*. This equation indicates that a positive value of β places a higher weight on the component with the most positive rate of change, which corresponds to the worst performing component in terms of learning. On the other hand, a negative value of β results in a higher relative weight for the term with the most negative rate of change or the best performing component. A zero value of β yields equal fixed weights for each loss component. The magnitude of β determines how sensitive the loss weights respond to the changes in individual loss components, with larger β values leading to more responsive adjustment. While the Softadapt method may require adjusting the hyperparameter β, it offers greater convenience compared to the iterative adjustment of the ratio among three loss weights required in the fixed loss weight approach.

Algorithm 1 Adaptive Loss Weighting for gradient descent based neural network training. The set of trainable variables of the model are represented by θ. Note, batching is omitted here for simplicity purposes. Normalization of **s** vector is considered as a default for reasons previously mentioned.

Require: *optimizer*
Require: *loss_fn* (loss function, which calculates the difference between target y_k value and prediction $h(x, \theta)$)
Require: n (update loss weights every n epochs)
Require: $\alpha_k^{(0)}$ (initial loss weights values)
Require: $\epsilon = 10^{-8}$ for numerical stability
loss_weights $\leftarrow$ *list*() empty list to store average loss weights for n epochs
$\alpha_k^{(i)} \leftarrow \alpha_k^{(0)}$
for $i = 1$ **in** *epochs* **do**
  $l_k^{(i)} \leftarrow loss\_fn(\mathbf{y}, h(\mathbf{x}, \boldsymbol{\theta}))$ Compute loss for current loss weights
  $l_T \leftarrow \alpha_k^{(i)}$ Compute total loss using current loss weights
  Perform back-propagation to update θ parameters
  if $(epoch\%n) == 0$ **then**
    epoch_loss_weights $\leftarrow$ *list*() empty list to store loss weights for current epoch
  end if
  $s_k^{(i)} \leftarrow l_k^{(i)} - l_k^{(i-1)} / \left((\sum_{m=1}^{M} |s|) + \epsilon \right)$
  $\alpha_k^{(i)} \leftarrow \exp(\beta s_k^{(i)}) / \left((\sum_{m=1}^{M} \exp(\beta s_m^{(i)})) + \epsilon \right)$
  Append $\alpha_k^{(i)}$ to *epoch_loss_weights*
  if $[(epoch + 1)\%n] == 0$ **then**
    loss_weights $\leftarrow$ Compute average of n entries in *epoch_loss_weights*
  end if
end for

2.4. Benchmark data generation

Due to lack of accessible datasets comprising high-fidelity stress data, we created a benchmark dataset using a two-dimensional $MoTe_2$ as a model material. Monolayer $MoTe_2$ is one of the members inside the 2D transition metal dichalcogenide (TMD) materials, which have shown great promise for many revolutionary applications in nanoelectronics, optoelectronics, and photonics [26]. It exhibits two stable structural phases: semiconducting 2H phase and metallic 1T′ [27]. The dataset consists of a total of 3146 structures, evenly distributed between the 2H and 1T′ phases of $MoTe_2$. It includes initially relaxed unit cells of 2H and 1T′, strained over a range of −10% to 10%, with increments of 2% for a total of 242 initial structures. In addition, 12 perturbed structures for each structure, with magnitudes of up to 1 Å drawn from a Gaussian distribution centered at the positions of each atom from the initial structure. Perturbing the atomic coordinates of a relaxed structure to generate additional training data has been shown to enhance the description of the phase space near the minima of the potential energy surface [28]. Structural defects and free surfaces were not included, as the dataset was created specifically for benchmarking rather than for training a general purpose interatomic potential.

All DFT calculations for data generation were performed using the plane-wave-based Vienna Ab-initio Simulation Package (VASP) [29,30]. Projector augmented wave (PAW) pseudopotentials [31,32] were used to represent ionic cores, and the electronic kinetic energy cutoff for the plane-wave basis describing the valence electrons was set to 293 eV. The Perdew–Burke–Ernzerhof (PBE) with the generalized gradient approximation (GGA) [33] was chosen for the exchange–correlation functional. The k-point selection was adapted to the dimension of each structure along the armchair and zigzag directions, keeping a k-point resolved value of $0.02\pi \times \text{Å}^{-1}$ in both directions, following

Table 1
RMSE for models trained with different loss weights combinations. The contribution of potential energy (meV/atom), force (meV/Å), and stress (MPa) are weighted by α_1, α_2, and α_3, respectively.

Model	Fixed loss weights			Training RMSE			Testing RMSE		
	α_1	α_2	α_3	Energy	Force	Stress	Energy	Force	Stress
1	1.0	0.00	0.00	2.37	1934.61	2832.32	19.28	2940.90	4890.23
2	0.90	0.10	0.00	1.98	15.87	489.98	2.30	24.28	520.72
3	**0.90**	0.05	0.05	2.72	21.39	22.02	**2.29**	28.42	34.10
4	0.05	**0.90**	0.05	7.61	16.00	30.17	6.12	**21.39**	37.20
5	0.05	0.05	**0.90**	7.79	42.69	15.48	7.42	48.43	**23.43**
6	0.33	0.33	0.33	3.58	20.23	18.11	5.41	25.50	22.44

the Monkhorst–Pack scheme [34] in VASPKIT [35]. As for the out-of-plane direction, a vacuum layer of 25 Å was used to separate the periodic images in the out-of-plane direction, thereby single k-point was maintained along this direction. The electronic energy and atomic forces were converged to 10^{-4} meV and 1 meV/Å, respectively, taking an average of 62.7 s of wall-time per structure on a cluster with 48 cores.

3. Results and discussion

The neural network models in this study are designed with a specific architecture, consisting of two hidden layers, each containing 30 neurons. A hyperbolic tangent function serves as the activation mechanism. The optimization is performed using the Adam optimizer, set at a learning rate of 0.001. Training of these models continued until negligible learning gains were observed. The dataset was divided into 70% training, 20% validation, and 10% testing.

3.1. Results of fixed loss weights

As discussed in Section 2.2, while various weighting schemes have been used in the literature, a detailed study that simultaneously considers potential energy, forces, and stresses in the loss function and investigates their weightings' impacts on the training performance is still missing. To address this, we first conducted a sequence of studies focusing on the effects of fixed loss weights. This involved systematically varying these weights and analyzing their impact on the performance of the trained model in terms of predictive capabilities in comparison with DFT calculations. The results are shown in Table 1.

In model 1, where the loss function incorporates only potential energy, the model's predictions for the forces and stresses are notably poor. Additionally, the testing error for energy (19.28 meV/atom) in this model is an order of magnitude higher than that of other models despite similar training error. This suggests that the model trained with energy alone lacks generalization capabilities. In the case of model 2, which is trained on both potential energy and atomic forces, we observe good performance in predicting energy and forces (2.30 meV/atom and 24.28 meV/Å, respectively). This significant improvement can be explained as follows. When the model is only trained with potential energy, we have 3146 potential energy values that can be used for training, each corresponding to a structure in the dataset. However, by incorporating the information of atomic forces (the gradients of potential energy), the number of training data points increases by $\sum_{i=1}^{3146} 3N_i$, where N_i is the number of atoms in ith structure. Therefore, by adding force information, the neural network receives much richer information about the physics of the material system, leading to more accurate predictions. One previous study has also found that including gradients benefits not only the accuracy of both energies and forces but also improves the stability of training [36]. Moreover, stress prediction from model 2 is improved compared to model 1, as stress is computed using the gradient of potential energy, as indicated by Eq. (A.7). However, the prediction still remains unsatisfactory compared to other models listed in the Table. Apparently, the absence of stress data during training notably deteriorates the model's ability to predict stress accurately. Although this was observed in this particular dataset, it is likely applicable to others as well. Model 3, which is trained on all three components delivers more balanced results across all metrics. The comparison between models 2 and 3 emphasizes also the importance of incorporating stress data in training ML-IAPs to more accurately predict stresses, i.e., 520.72 versus 34.10 MPa testing RMSE, in addition to energies and forces.

Models 3 to 5 were each trained with distinct weight combinations, specifically designed to prioritize one of three variables: potential energy, force, or stress. The results showed that the weight coefficient for one variable improves its corresponding result but at the cost of reducing accuracy in the other one or two variables. Specifically, model 3, which prioritized potential energy, exhibited the lowest testing loss for potential energy (2.29 meV/atom) but did so by sacrificing stress prediction accuracy (34.10 MPa). Model 4, focusing on force, achieved the lowest force testing loss (21.39 meV/Å), but compromising energy and stress accuracy. Similarly, model 5 achieved the best results for stress (23.43 MPa) at the expense of potential energy and force accuracy. This observed trade-off pattern suggests that previously reported approaches, which favor the forces during potential training (such as $\alpha_1 : \alpha_2 = 1 : 10$) [37], or completely overlook the energy term and focus exclusively on forces [38], might not be universally applicable across different datasets or material systems. Additionally, model 6, which distributed weights evenly across all three variables, managed to achieve satisfactory test results for force (25.50 meV/Å) and stress (22.44 MPa), yet it did not accurately predict the potential energy (5.41 meV/atom), more than twice that of model 3. Therefore, this fixed weight approach could require extensive experimentation with loss weight combinations, specific to the material dataset, in order to attain a balanced performance among all three variables. This suggests the potential benefits of employing automated methods to streamline the training process.

3.2. Results using adaptive loss weights

As discussed in the previous section, achieving optimal results for potential energy, force, and stress simultaneously may require iteratively adjusting the loss weight ratios based on material and dataset characteristics. By contrast, we propose the Softadapt method to dynamically balance each term's contribution to the loss function. In this section, the results of models trained using Softadapt are compared with those trained using fixed weights. For a fair comparison, all neural network models implementing the adaptive algorithm were configured with the same architecture and activation function to their fixed weight counterparts.

The results of models trained with Softadapt algorithm are presented in Table 2. The hyperparameter β controls how sensitively the weights respond to rate changes in loss components, with higher β values causing more rapid adjustments in loss weights during training. For the benchmark data, the optimal performance of Softadapt algorithm is achieved when β is around the order of 1.0, offering a better and more balanced testing errors across energy, force, and stress compared to the fixed loss weight models. When β is too low, the weight adjustments do not adequately keep up with loss rate changes. Conversely, when β is set too high, it leads to significant fluctuations in loss weights,

Table 2
RMSE for models trained using Softadapt algorithm with different β values. Units for potential energy, force, and stress are meV/atom, meV/Å, and MPa, respectively.

Model	Softadapt	Training RMSE			Testing RMSE		
	β	Energy	Force	Stress	Energy	Force	Stress
1	0.01	4.58	26.51	25.67	4.03	32.52	29.70
2	0.1	3.47	21.20	19.42	4.15	27.27	27.08
3	1.0	3.07	18.32	14.55	2.74	23.85	23.54
4	2.0	2.80	17.90	15.24	2.57	25.70	24.62

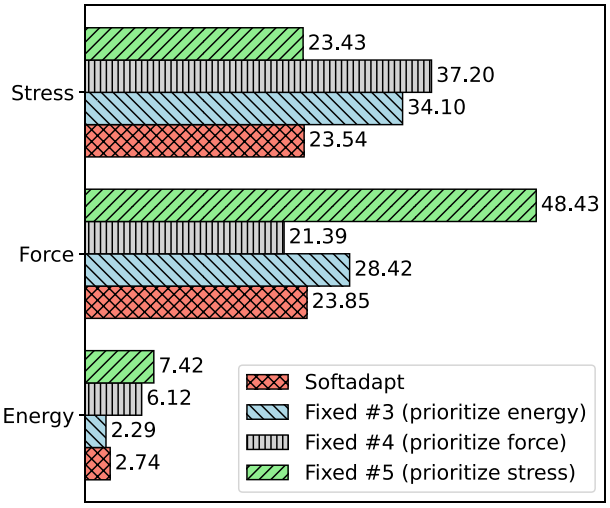

Fig. 2. RMSE comparison between the model trained using Softadpt algorithm and three models trained with fixed loss weights, each prioritizing potential energy, force, and stress. The RMSE values are taken from Tables 1 and 2. Units for potential energy, force, and stress are meV/atom, meV/Å, and MPa, respectively.

resulting in unstable training and increased errors. The comparison between models trained using Softadapt algorithm with $\beta = 1$ and those trained using fixed weights is illustrated in Fig. 2. Notably, for each term of potential energy, force, and stress, the Softadapt method achieves an accuracy level equivalent to that of the corresponding fixed model where that specific term is prioritized.

The results in Table 2 were trained using equal initial weights among potential energy, force, and stress, each equal to 0.33. It is important to note that the final results are not sensitive to the selection of the initial weights, due to the nature of adaptive algorithm. To examine the influence of adaptive algorithm on the changing of loss weights, we monitored the variation of loss weights throughout the training process. In this analysis, we intentionally varied the initial loss weights across three configurations: (0.9, 0.05, 0.05), (0.05, 0.9, 0.05), and (0.05, 0.05, 0.9). As shown in Fig. 3, despite starting with very different initial loss weights, the algorithm adjusts the contribution of each, converging to an optimum combination. Interestingly, for this particular dataset, noticeable adjustment in loss weights mainly occur within the first 1500 epochs.

Finally, we want to discuss the accuracy and efficiency of the trained potential. The RMSE of energy and force from our adaptive model is within a similar range compared to previous studies. For example, Zuo et al. [39] trained neural network potentials for six different metal elements, and their RMSE for energy and force ranged from 1.24~11.27 meV/atom and 50~200 meV/Å, respectively, although they did not report the RMSE of stress. In a separate study [40], Rosenbrock et al. compared GAP and MTP models trained on Ag-Pd alloy, with RMSE values ranging from 10~25 meV/atom for energy, 220~240 meV/Å for force, and 1300~2034 MPa for stress. Our RMSE values are lower, particularly for stress, which is two orders of magnitude lower. This difference may be attributed to the different complexity of the material system and the training dataset. In terms of efficiency, tested using a 1 ps MD simulation of a unit cell $MoTe_2$, our trained potential yields 0.001 s/(MD step·atom), whereas DFT takes 3.9 s/(MD step·atom) for comparison.

4. Summary

We proposed an adaptive loss weighting method to dynamically adjust the contribution of potential energy, force, and stress, based on their corresponding loss values during the training of machine learning interatomic potentials. Leveraging a benchmark dataset, we conducted a comparative analysis between models trained with fixed and adaptive loss weights. The key findings are summarized as follows.

- Stress data proves critical for the accurate prediction of stress values when training machine learning potentials.
- Models employing fixed loss weights yield imbalanced predictions for potential energy, force, and stress, as they enhance the accuracy of a prioritized variable at the expense of others.
- Models utilizing the adaptive algorithm demonstrate an ability to balance the contributions of the three variables, thereby yielding more balanced and accurate predictions.

CRediT authorship contribution statement

Daniel Ocampo: Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Daniela Posso:** Software, Investigation. **Reza Namakian:** Investigation, Writing – review & editing. **Wei Gao:** Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Wei Gao reports financial support was provided by National Science Foundation Division of Civil Mechanical and Manufacturing Innovation. If there are other authors, they 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

The data and code reported in this paper are available on Github: https://gao-group.github.io/atomdnn/index.html.

Acknowledgments

W.G. gratefully acknowledges financial support of this work by the National Science Foundation, United States through Grant no. CMMI-2308163 and CMMI-2305529. The authors acknowledge the Texas Advanced Computing Center (TACC) at the University of Texas at Austin and Texas A&M High Performance Research Computing for providing HPC resources that have contributed to the research results reported within this paper.

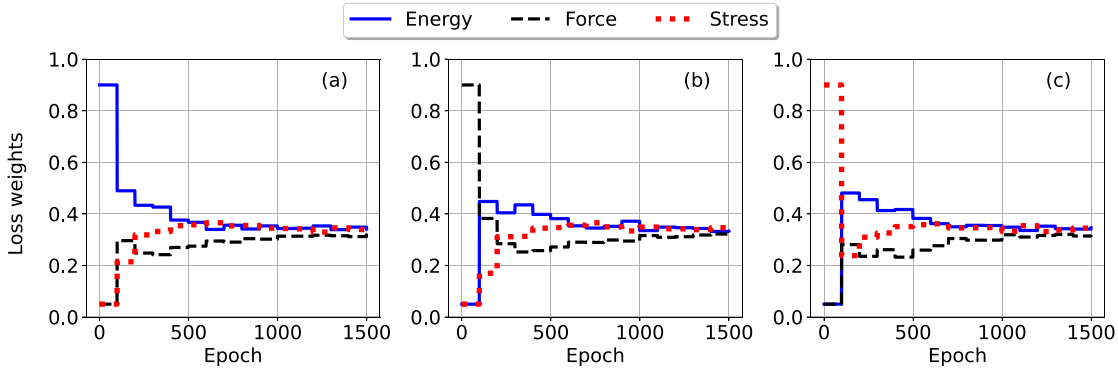

Fig. 3. Variation of loss weights during training using the Softadapt algorithm, with different initial weight ratios among potential energy, force and stress: (a) 0.9:0.05:0.05, (b) 0.05:0.9:0.05 and (c) 0.05:0.05:0.9.

Appendix. Cauchy stress derivation in ML-IAP

The first Piola–Kirchhoff (PK) stress tensor can be calculated as the work conjugate of deformation gradient tensor

$$P_{\alpha\beta} = \frac{1}{V_0}\frac{\partial \mathcal{E}}{\partial F_{\alpha\beta}}, \tag{A.1}$$

where V_0 is the volume of the reference configuration, and $F_{\alpha\beta}$ is the deformation gradient tensor. Similar to the atomic force calculation, the potential energy can be written as the sum of per-atom potential energies, so the stress can be written as

$$P_{\alpha\beta} = \frac{1}{V_0}\sum_{i=1}^{N}\sum_{m=1}^{M}\frac{\partial E_i}{\partial G_{im}}\frac{\partial G_{im}}{\partial F_{\alpha\beta}}. \tag{A.2}$$

The fingerprint G_{im} is determined by the coordinates of the atoms inside the neighbor list of atom i. Therefore, using the chain rule, we have

$$\frac{\partial G_{im}}{\partial F_{\alpha\beta}} = \sum_{j\in \mathrm{NB}_i}\sum_{\gamma=1}^{3}\frac{\partial G_{im}}{\partial r_{j\gamma}}\frac{\partial r_{j\gamma}}{\partial F_{\alpha\beta}}, \tag{A.3}$$

where atom j is inside the neighbor list of atom i (represented by NB_i). By definition, the deformation gradient maps the atom position from the reference configuration to the current configuration

$$r_{j\gamma} = \sum_{\beta=1}^{3} F_{\gamma\beta} R_{j\beta}, \tag{A.4}$$

where $R_{j\beta}$ is the coordinates of atom j in the reference configuration. Then, the stress in Eq. (A.2) can be written as

$$P_{\alpha\beta} = \frac{1}{V_0}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{j\in \mathrm{NB}_i}\frac{\partial E_i}{\partial G_{im}}\frac{\partial G_{im}}{\partial r_{j\alpha}}R_{j\beta}. \tag{A.5}$$

Cauchy stress can be further calculated by

$$\sigma_{\alpha\gamma} = \det(\boldsymbol{F})^{-1}\sum_{\beta=1}^{3} P_{\alpha\beta}F_{\gamma\beta}. \tag{A.6}$$

Substitute Eq. (A.5) into Eq. (A.6), and then apply Eq. (A.4) and $V = V_0\det(\boldsymbol{F})$, which is the volume in current configuration, we can get

$$\sigma_{\alpha\beta} = \frac{1}{V}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{j\in \mathrm{NB}_i}\frac{\partial E_i}{\partial G_{im}}\frac{\partial G_{im}}{\partial r_{j\alpha}}r_{j\beta} \tag{A.7}$$

after replacing γ with β.

References

[1] A.P. Bartók, J. Kermode, N. Bernstein, G. Csányi, Machine learning a general-purpose interatomic potential for silicon, Phys. Rev. X 8 (4) (2018) 041048.

[2] P. Rowe, G. Csányi, D. Alfe, A. Michaelides, Development of a machine learning potential for graphene, Phys. Rev. B 97 (5) (2018) 054303.

[3] M. Wen, E.B. Tadmor, Hybrid neural network potential for multilayer graphene, Phys. Rev. B 100 (19) (2019) 195419.

[4] A.C. Jain, D. Marchand, A. Glensk, M. Ceriotti, W. Curtin, Machine learning for metallurgy III: A neural network potential for Al-Mg-Si, Phys. Rev. Mater. 5 (5) (2021) 053805.

[5] J. Behler, M. Parrinello, Generalized neural-network representation of high-dimensional potential-energy surfaces, Phys. Rev. Lett. 98 (14) (2007) 146401.

[6] R. Drautz, Atomic cluster expansion for accurate and transferable interatomic potentials, Phys. Rev. B 99 (1) (2019) 014104.

[7] A.V. Shapeev, Moment tensor potentials: A class of systematically improvable interatomic potentials, Multiscale Model. Simul. 14 (3) (2016) 1153–1173.

[8] F. Musil, A. Grisafi, A.P. Bartók, C. Ortner, G. Csányi, M. Ceriotti, Physics-inspired structural representations for molecules and materials, Chem. Rev. 121 (16) (2021) 9759–9815.

[9] J. Behler, Atom-centered symmetry functions for constructing high-dimensional neural network potentials, J. Chem. Phys. 134 (7) (2011) 074106.

[10] A.P. Bartók, M.C. Payne, R. Kondor, G. Csányi, Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons, Phys. Rev. Lett. 104 (13) (2010) 136403.

[11] A.P. Thompson, L.P. Swiler, C.R. Trott, S.M. Foiles, G.J. Tucker, Spectral neighbor analysis method for automated generation of quantum-accurate interatomic potentials, J. Comput. Phys. 285 (2015) 316–330.

[12] Y. Lysogorskiy, C.v.d. Oord, A. Bochkarev, S. Menon, M. Rinaldi, T. Hammerschmidt, M. Mrovec, A. Thompson, G. Csányi, C. Ortner, et al., Performant implementation of the atomic cluster expansion (PACE) and application to copper and silicon, npj Comput. Mater. 7 (1) (2021) 97.

[13] H. Wang, L. Zhang, J. Han, E. Weinan, DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics, Comput. Phys. Comm. 228 (2018) 178–184.

[14] T. Xie, J.C. Grossman, Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties, Phys. Rev. Lett. 120 (14) (2018) 145301.

[15] C.W. Park, C. Wolverton, Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery, Phys. Rev. Mater. 4 (6) (2020) 063801.

[16] K.T. Schütt, H.E. Sauceda, P.-J. Kindermans, A. Tkatchenko, K.-R. Müller, Schnet–a deep learning architecture for molecules and materials, J. Chem. Phys. 148 (24) (2018) 241722.

[17] C. Chen, W. Ye, Y. Zuo, C. Zheng, S.P. Ong, Graph networks as a universal machine learning framework for molecules and crystals, Chem. Mater. 31 (9) (2019) 3564–3572.

[18] J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, et al., DeePMD-kit v2: A software package for deep potential models, J. Chem. Phys. 159 (5) (2023).

[19] W. Gao, D. Ocampo, D. Posso, Atomdnn, 2021, https://gao-group.github.io/atomdnn/index.html.

[20] A. Pukrittayakamee, M. Malshe, M. Hagan, L. Raff, R. Narulkar, S. Bukkapatnum, R. Komanduri, Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks, J. Chem. Phys. 130 (13) (2009).

[21] A.M. Cooper, J. Kästner, A. Urban, N. Artrith, Efficient training of ANN potentials by including atomic forces via taylor expansion and application to water and a transition-metal oxide, npj Comput. Mater. 6 (1) (2020) 54.

[22] H. Yanxon, Developments of Machine Learning Potentials for Atomistic Simulations (Ph.D. thesis), University of Nevada, Las Vegas, 2020.

[23] A. Bochkarev, Y. Lysogorskiy, S. Menon, M. Qamar, M. Mrovec, R. Drautz, Efficient parametrization of the atomic cluster expansion, Phys. Rev. Mater. 6 (1) (2022) 013804.

[24] T.G. Sours, A.R. Kulkarni, Predicting structural properties of pure silica zeolites using deep neural network potentials, J. Phys. Chem. C 127 (3) (2023) 1455–1463.
[25] A.A. Heydari, C.A. Thompson, A. Mehmood, Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions, 2019, arXiv preprint arXiv:1912.12355.
[26] M. Chhowalla, H.S. Shin, G. Eda, L.-J. Li, K.P. Loh, H. Zhang, The chemistry of two-dimensional layered transition metal dichalcogenide nanosheets, Nat. Chem. 5 (4) (2013) 263–275.
[27] A. Ghasemi, W. Gao, Atomistic mechanism of stress modulated phase transition in monolayer MoTe2, Extreme Mech. Lett. 40 (2020) 100946.
[28] J. Gibson, A. Hire, R.G. Hennig, Data-augmentation for graph neural network learning of the relaxed energies of unrelaxed structures, npj Comput. Mater. 8 (1) (2022) 211.
[29] G. Kresse, J. Furthmüller, Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set, Phys. Rev. B 54 (16) (1996) 11169.
[30] G. Kresse, J. Hafner, Ab initio molecular dynamics for liquid metals, Phys. Rev. B 47 (1) (1993) 558.
[31] G. Kresse, D. Joubert, From ultrasoft pseudopotentials to the projector augmented-wave method, Phys. Rev. b 59 (3) (1999) 1758.
[32] P.E. Blöchl, Projector augmented-wave method, Phys. Rev. B 50 (24) (1994) 17953.
[33] J.P. Perdew, K. Burke, M. Ernzerhof, Generalized gradient approximation made simple, Phys. Rev. Lett. 77 (18) (1996) 3865.
[34] H.J. Monkhorst, J.D. Pack, Special points for brillouin-zone integrations, Phys. Rev. B 13 (12) (1976) 5188.
[35] V. Wang, N. Xu, J.-C. Liu, G. Tang, W.-T. Geng, VASPKIT: A user-friendly interface facilitating high-throughput computing and analysis using VASP code, Comput. Phys. Comm. 267 (2021) 108033.
[36] A.S. Christensen, O.A. Von Lilienfeld, On the role of gradients for machine learning of molecular energies and forces, Mach. Learn.: Sci. Technol. 1 (4) (2020) 045018.
[37] A. Singraber, T. Morawietz, J. Behler, C. Dellago, Parallel multistream training of high-dimensional neural network potentials, J. Chem. Theory Comput. 15 (5) (2019) 3075–3092.
[38] S. Chmiela, H.E. Sauceda, K.-R. Müller, A. Tkatchenko, Towards exact molecular dynamics simulations with machine-learned force fields, Nat. Commun. 9 (1) (2018) 3887.
[39] Y. Zuo, C. Chen, X. Li, Z. Deng, Y. Chen, J. Behler, G. Csanyi, A.V. Shapeev, A.P. Thompson, M.A. Wood, et al., Performance and cost assessment of machine learning interatomic potentials, J. Phys. Chem. A 124 (4) (2020) 731–745.
[40] C.W. Rosenbrock, K. Gubaev, A.V. Shapeev, L.B. Pártay, N. Bernstein, G. Csányi, G.L. Hart, Machine-learned interatomic potentials for alloys and alloy phase diagrams, npj Comput. Mater. 7 (1) (2021) 24.