Evaluation scheme for self-adaptive methods of coefficients of loss components of multi-objective loss function

Spilios Dellis
National Centre for Scientific Research "Demokritos"
Athens, Greece
spiliospdellis@gmail.com

Eleonora Ricci
National Centre for Scientific Research "Demokritos"
Athens, Greece
e.ricci@inn.demokritos.gr

Dimitris-Paraskevas Gerakinis
National Centre for Scientific Research "Demokritos" & National Technical University of Athens
Athens, Greece
dp.gerakinis@inn.demokritos.gr

Niki Vergadou
National Centre for Scientific Research "Demokritos"
Athens, Greece
n.vergadou@inn.demokritos.gr

George Giannakopoulos
National Centre for Scientific Research "Demokritos"
Athens, Greece
ggianna@iit.demokritos.gr

ABSTRACT

This work addresses a challenge related to Multi-Objective Optimization in machine learning model training, specifically the problem of loss coefficients weight determination for physics grounded tasks. We propose a comprehensive comparative methodology for the analysis of balancing methods for loss function coefficients in deep learning models, to enhance replicability and comparisons across diverse applications, emphasizing the use of physical parameters as figures of merit. The proposed methodology is illustrated through the evaluation of self-adaptive methods for multicomponent loss coefficients in Graph Convolutional Neural Network (GCNN) models. The GCNN are trained to reproduce the forces acting on particles during coarse-grained molecular dynamics simulations. Criteria are outlined both for individual model assessment and for a statistical comparison between methods, highlight the differences in training-related characteristics, and performance metrics for the downstream task, across various self-balancing approaches.

1. Introduction

In the realm of machine learning (ML) model training, a prevalent challenge involves optimizing an objective function crafted as a weighted linear combination of multiple losses [1]. The efficacy of a model is intricately tied to the assignment of weights to each loss component during training. Traditionally, deciding the optimal set of weights entails designating them as hyperparameters and executing an exhaustive grid search, which is a computationally demanding process. However, recent years have witnessed the emergence of various weight adaptation methods designed to alleviate this computational burden during training of ML models.

Multi-Objective Optimization (MOO) targets simultaneous optimisation of a set of $k > 1$ objectives, that can potentially be conflicting and which can be turned into a single objective through linear scalarization [2]:

$$\mathcal{L}(\theta) = \sum_{i=1}^{k} \lambda_i \mathcal{L}_i(\theta) \ , \ \lambda_i \in R > 0 \qquad \text{Eq. 1}$$

In various domains such as engineering, natural sciences, and economics, numerous challenges can be cast as MOO problems, often necessitating trade-offs to concurrently fulfil multiple objectives [3]. The solutions to MOO models are typically expressed as sets of Pareto optima, representing optimal compromises among given criteria. While, theoretically, Pareto optimal solutions should be independent of scalarization [4], the application of neural networks in MOO introduces a highly non-convex solution space. Despite the universal approximation capabilities of neural networks [5], achieving global optimality through gradient-based optimization is not guaranteed. To address this, scaling the loss space offers a means to guide gradients with a predetermined desirable property. However, manual determination of optimal scaling factors λ_i becomes impractical, particularly as the number of objectives increases, or when dynamic evolution of λ_i during the training process is desired.

Strategies proposed to balance loss coefficients in deep learning models involve various methodologies, including manual tuning, grid search, random search, Bayesian optimization, and multi-task learning [3,6–8]. Each method presents unique advantages and drawbacks. For instance, manual tuning, though straightforward, can be labour-intensive and suboptimal due to its reliance on human intuition. Automated techniques, such as Bayesian optimization and multi-task learning, streamline the process but may require significant computational resources. In addition, despite these options, a critical gap

persists in establishing a comprehensive and application-agnostic comparative test to evaluate and contrast multiple methodologies. Existing methods often rely on model-specific metrics like accuracy or loss, which may lack direct relevance to physical outcomes in the downstream task for which the models are used. Consequently, there is a pressing need for an evaluation methodology that employs physical parameters as key figures of merit, allowing for a more direct assessment of a model's applicability to physical problems.

In addressing this gap, our research aims to contribute a robust, quantitative, and reproducible comparative methodology for assessing loss coefficient balancing techniques. This methodology should transcend application boundaries and offer transparent statistical figures of merit, grounded in physical parameters. The motivation behind this endeavour lies in the requirement to facilitate comparisons across diverse applications, ensuring replicability of outcomes. We argue for the adoption of physical parameters as figures of merit, particularly in domains like molecular dynamics, where metrics such as accuracy in potential energy and forces prediction, as well as system stability during simulations are more meaningful to determine the suitability of a trained model than training-related metrics [9]. To illustrate our proposed methodology, we employ self-balancing techniques for a multi-objective loss function during the training of Graph Convolutional Neural Network (GCNN) models, used for the prediction of particle interactions in coarse-grained molecular dynamics simulations [10,11]. Our loss function comprises two components, one targeting forces acting on coarse-grained particles and the other related to the prediction of the intermolecular potential energy in a bulk liquid system [11,12]. These examples serve as tangible demonstrations of an automated, rigorous method for model pre-screening and performance quantification, contributing valuable insights towards our overarching goal of proposing a broadly applicable approach for evaluating deep learning models, not only in molecular dynamics but for physics-grounded problems in general.

2. Methodology

2.1 Test case description and model architecture

In this work we tested different loss coefficient self-balancing methods and developed comparative metrics for a physics-grounded deep learning task. We trained Graph Convolutional Neural Networks (GCNN) to reproduce the forces acting on particles during Coarse-Grained (CG) molecular dynamics

(MD) simulations. Coarse Graining consists of averaging out a certain number of high-resolution degrees of freedoms, namely atoms, into a single interaction site, or coarse-grained bead (i.e. choosing a CG mapping) and represent the interactions between the CG beads consistently with the underlying atomistic system, namely developing a CG force field, or CG potential. Adopting a CG molecular description enables access to larger lengths and longer timescales, thereby expanding the reach of systems and phenomena that can be investigated with molecular simulations. In this work, we explored the possibility to train ML models, such as neural networks, to replace traditional force fields based on predefined functionals. Traditional CG models typically use predefined pairwise functionals to describe particle interactions, which inherently limits their ability to describe many-body interactions. To overcome the limitations of traditional methods, in this work we investigated the development of an ML-based CG simulation strategy [11]. In recent years, ML-based strategies have been applied to bridge the quantum-mechanical and atomistic levels of descriptions, mostly for the case of inorganic systems or isolated small organic molecules [6]. However, their application to CG simulations, necessary to study complex macromolecular systems at bulk conditions has been scarcely explored so far. The chemical system considered for this work is liquid benzene at bulk conditions, and the CG mapping chosen is 1 CG bead per molecule [11].

The ML architecture employed in this study is an adaptation of the SchNet model [13,14], specifically tailored for the development of Coarse-Grained (CG) Machine Learned force fields applied to bulk organic systems [11]. Our focus centres on utilizing liquid benzene as a test system, and the training strategy incorporates a force-matching scheme [12,15], described in detail elsewhere [11]. The main features of the SchNet architecture utilized are summarized hereafter. For a more detailed description, the original works can be consulted [13,14]. The GCNN architecture utilized adopts a graph representation of molecules, with nodes corresponding to particles and edges to bonds or interatomic distances. Each particle is represented through a feature vector, which is initialized to distinguish between the particle chemical identities (embedding layer). The feature is then updated to encode information on the surrounding environment, by performing continuous convolutions across the particle neighbourhood, optimizing the convolutional filter weights during the training (convolutional layers). By performing multiple convolution operations, each node can influence other increasingly distant nodes, thereby potentially encoding long-range information into the particle descriptor. Afterwards, a

fully connected section is included (readout layers). A sequence of continuous convolution layers and readout layers constitutes a SchNet "block". Multiple blocks can be utilized in series to define the full network architecture. The output found at the end of the last block can be interpreted as a learned feature representation, which encodes the information from the particle neighbourhood required to predict the target property. Finally, given the learned feature representation as input, a fully connected (dense) section is tasked with predicting the final scalar output, which is interpreted as a per-particle energy contribution. This local decomposition ensures invariance to the total number of particles. All energy contributions are summed to obtain the total energy which is then differentiated with respect to the positions of the particles to predict the force acting on each particle.

Overall, there are several hyperparameters to optimize in the SchNet architecture. Hyperparameters are user-defined settings of ML models that control various aspects of the model architecture and behaviour, such as (in this case):

- the model size & architecture (number of convolutional filters, filters size, number of SchNet blocks, activation function)
- the particle representation (feature vector size, cutoff radius of the local neighbourhood)
- the training process (learning rate, decay ratio, batch size, number of samples, number of epochs)
- the embedded prior terms (excluded volume diameter, excluded volume exponent)

The optimal values for these hyperparameters were adopted from a previous study, carried out on the same physical system [11] focusing here only on the optimization of the loss coefficients, as described in the following.

2.2 Loss Function definition

The neural network training involves minimizing a loss function comprising two terms. The first term corresponds to the mean squared difference between predicted and MD-derived forces, while the second term pertains to the intermolecular energy of the system. As detailed in a previous work, the inclusion of the second term proved essential, as solely training on forces led to non-meaningful ML models during a wide hyperparameter search, especially in systems with exclusively intermolecular interactions at the CG level [11]. The loss function is formulated as:

$$L = \lambda_1 \frac{1}{3N} \sum_{i}^{N} \left(-\nabla_{x_i} (\Sigma_i \widehat{U}_i + U_{ex}) - \mathcal{M} \left(\boldsymbol{F}^A (\boldsymbol{r_j}) \right) \right)^2 + \lambda_2 \left(\left(\Sigma_i \widehat{U}_i + U_{ex} \right) - U_A \right)^2 \qquad \text{Eq. 2}$$

where N is the total number of CG particles, λ_1 and λ_2 the loss components scaling coefficients, $\mathcal{M}\left(\mathbf{F}^{\mathrm{A}}(\mathbf{r}_j)\right)$ is the force acting on the CG particle at position $\mathbf{x}_i$, obtained from the atomistic MD simulation mapped to the CG space, while the term in square brackets is the force acting on the CG particle at position $\mathbf{x}_i$ predicted by the GCNN. $\widehat{U}_i$ are the per-particle energy values predicted by the CGNN, and U_A is the intermolecular energy component from the atomistic simulation. The loss function includes a regularization term U_{ex} linked to excluded volume repulsion effects. This prior term ensures that the system's energy tends towards infinity for nonphysical states where particles have close contacts between each other. Specifically, the excluded volume energy is added to the total energy predicted by the network, prior to differentiation. This prior term is expressed as:

$$U_{ex} = \sum_{i=1}^{N-1} \sum_{j=i}^{N} \left(\frac{\sigma}{\left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|} \right)^{n^{ex}} \qquad \text{Eq. 3}$$

where σ and n^{ex} are hyperparameters of the model, for which suitable values must be identified. The training dataset was generated through atomistic Molecular Dynamics (MD) simulations of a liquid benzene system containing 500 molecules, conducted at 340 K and atmospheric pressure, using LAMMPS [16]. After equilibration through a 1 ns NPT simulation, a subsequent 20 ns NVT run at the average equilibrium density was carried out, from which 10000 configurations were saved every 1 ps. 9000 configurations were utilized for model training, 1000 for the test set. The molecular configurations from the MD trajectory were randomly assigned to either set or shuffled during the training. Subsequent NVT CG simulations with trained ML potentials were conducted using the ASE integrator [17] to validate the ML-based predictions and optimize the hyperparameter set accordingly. Further details on the methodology and the full set of hyperparameters are available in Ref. [11].

2.3 Evaluation metrics

To comprehensively assess the performance of each model, various simulation-derived metrics were computed for the CG system and compared against the atomistic simulation reference. The evaluated parameters include potential energy, mean square displacement, pair correlation function, and

temperature. Statistical parameters extracted from these results were employed as metrics within the evaluation framework. These metrics are:

- ΔU (Absolute Percentage Difference in Potential Energy): This metric quantifies the absolute percentage difference between the mean value of the coarse-grained potential energy during a CG simulation and its average value in the training set, offering insights into the accuracy of energy predictions.

- $CV_{U_{CG}}$ (Coefficient of Variation of Coarse-Grained Potential Energy): The coefficient of variation of coarse-grained simulated potential energy assesses the stability of simulation results. A low coefficient of variation signifies greater stability, while a high value may indicate energy drifts or unphysically high oscillations.

- $\bar{T}$ (Mean Coarse-Grained Simulated Temperature): This parameter represents the mean temperature of the coarse-grained simulation with the trained model, and it is an indicator of the simulation stability.

- CV_T (Coefficient of Variation of Coarse-Grained Simulated Temperature): Similar to the coefficient of variation for potential energy, the coefficient of variation for temperature gauges the stability of the simulation, with higher values potentially indicative of model instability.

- d_{KS} and p_{KS} (Kolmogorov–Smirnov Test Parameters for Pair Correlation Functions): The statistical parameters d_{KS} and p_{KS} from the Kolmogorov–Smirnov test assess the similarity between coarse-grained and atomistic simulated pair correlation functions. A lower d_{KS} and a higher p_{KS} indicate better agreement between the two functions.

- s_{MSD} (slope of the last 30% of Mean Square Displacement): The slope of the last 30% of the mean square displacement curve is expression of a specific mode of failure: a slope approaching 0 suggests a frozen system with minimal particle mobility beyond small in-place oscillations.

These chosen parameters facilitate a quantifiable assessment of model performance, eliminating the need for exhaustive results visualization. A hierarchy between these metrics can be defined and followed for a pre-screening of individual models. Notably, the second and fourth parameters, based on the Coefficient of Variation (CV) for potential energy and temperature, respectively, serve as simulation stability indicators, which is a crucial first requisite for model success. Subsequently, if the initial stability criteria are met, the first and third parameters can be considered to verify the validity of results by comparing mean potential energy and temperature values from coarse-grain simulations with the counterparts derived from the reference atomistic simulation. If these are satisfactory, the fifth

parameter, employing the Kolmogorov-Smirnov (K-S) test, indicates the accuracy in the structural representation, by evaluating the similarity between pair correlation functions in coarse-grained and atomistic simulations.

Additionally, parameters reflecting the efficiency of the training process can be defined to contribute to the comparison between different model training methods. These include: the number of training epochs (N_{ep}), mean duration per epoch (t_{ep}), and the standard deviation (σ_{ep}) of the duration of each training epoch. These efficiency metrics provide valuable insights into the computational resources' requirements, enhancing the overall evaluation framework.

2.4 Loss coefficients self-adaptive methods

The self-adaptive methods for loss coefficients employed in this study are categorized into two groups. The first group encompasses value-based methods that dynamically adjust the loss coefficients based on their actual values, while the second group involves methods that leverage the gradient information of the loss components for adaptation.

2.4.1 Loss component value-based methods

The first set of methods for self-adaptive loss coefficients in this study comprises two approaches, the "constant value" and the "ratio" method. For the "constant value" method, the coefficients (λ_i) of each loss component, i, are considered hyperparameters of the ML model and are optimized through a detailed grid search (Eq. 4).

$$\lambda_i(t) = A_i \qquad i = 1, \dots, k-1 \ , \ A_i \in R \geq 0 \qquad \text{Eq. 4}$$

The "ratio" method calculates the coefficients at each epoch t using the normalized value of each loss component in the previous epoch $(t-1)$ relative to a reference loss component also in the previous epoch (Eq. 5). The choice of the reference loss component k remains arbitrary but consistent throughout training.

$$\lambda_i(t) = \frac{L_i(t-1)}{L_k(t-1)} \qquad i = 1, \dots, k-1 \qquad \text{Eq. 5}$$

2.4.1 Loss component gabased methods

The second set of self-adaptive methods comprises three variations within the SoftAdapt family, developed by A. Ali Heydari et al. [16] to address the challenges associated with weighting multi-component loss functions. These methods have found application in image reconstruction and synthetic data generation. Inspired by softmax, the SoftAdapt methods adaptively adjust the weights in the linear combination of individual objective functions based on their respective performances and the collective loss function. The methodology assesses performance by approximating the rate of change of each loss function over a brief history, discerning whether it has been increasing or decreasing. Subsequently, these individual rates of change are compared, determining the visibility of each objective function to the optimizer.

The SoftAdapt family consists of three variants: the original version, the weighted variant, and the normalized variant. Each variant introduces unique refinements to the adaptive weight adjustment process, providing a nuanced approach to addressing the dynamic nature of the loss functions during model training. Mathematically, the three methods are described as:

$$\text{Original} \qquad \lambda_i(t) = \frac{\exp\left(\beta s_i(t)\right)}{\sum_{j=1}^{k} \exp\left(\beta s_j(t)\right)} \qquad i = 1, \dots, k \qquad \text{Eq. 6}$$

$$\text{Weighted} \qquad \lambda_i(t) = \frac{L_i(t) \exp\left(\beta s_i(t)\right)}{\sum_{j=1}^{k} L_j(t) \exp\left(\beta s_j(t)\right)} \qquad i = 1, \dots, k \qquad \text{Eq. 7}$$

$$\text{Normalized} \qquad \lambda_i(t) = \frac{\exp\left(\beta \frac{s_i(t)}{\sum_{l=1}^{k} s_l(t)}\right)}{\sum_{j=1}^{k} \exp\left(\beta \frac{s_j(t)}{\sum_{l=1}^{k} s_l(t)}\right)} \qquad i = 1, \dots, k \qquad \text{Eq. 8}$$

In Eqs. 6, 7, 8 the term $s_i(t)$ is the recent rate of change of the i th loss term (e.g. $s_i(t) = L_i(t) - L_i(t-1)$ or a more accurate finite difference approximation), and β is a tunable hyper-parameter. If $\beta > 0$ is used, SoftAdapt will assign more weight to the worst performing component of the loss function (*i.e.* the component with most positive rate of change), $\beta < 0$ favours the best performing losses (most negative rate of change) and using $\beta = 0$ gives equal weights (classic SoftAdapt method). The default value of the β hyperparameter according to the original authors is 0.1. In the normalized version of the SoftAdapt $s_i(t)$ is normalized before using it.

2.5 Evaluation scheme

To systematically evaluate and compare the effectiveness of various self-adaptation approaches for balancing loss coefficient terms, a structured procedure was developed. Five models were trained for each self-adaptation approach, each initiated with different seed values to define diverse initial conditions. Consequently, distinct populations of models were generated for each self-adaptation approach. The evaluation of individual models was based on the metrics outlined in Section 2.3, serving as quantitative metrics to capture diverse behavioural characteristics.

Following the individual model evaluations, a comparative analysis of model populations was undertaken. This involved conducting a Kruskal-Wallis test on the values of the metrics of each model population. The Kruskal-Wallis test, a non-parametric statistical method, compares the medians of two or more independent groups. The Kruskal-Wallis test does not assume a normal distribution of data, offering flexibility for non-normally distributed datasets. However, it does require that samples within groups are independent and identically distributed. The null hypothesis posits that all samples originate from the same distribution or from distributions with the same median. Rejection of the null hypothesis, indicated by a p-value below the significance level, suggests a significant difference between at least two groups.

Upon identifying statistically significant differences using the Kruskal-Wallis test, a post hoc test was conducted to discern which models differed and to what extent. The post hoc test employed in this study is Tukey's range test, also known as Tukey's Honest Significant Difference (HSD) test. This statistical method identifies means that are significantly different from each other. Tukey's range test allow to specify which groups exhibit significant differences by comparing all possible pairs of means, proving particularly useful when assuming equal variances across groups.

The proposed evaluation scheme is illustrated using a block diagram in Figure 1.

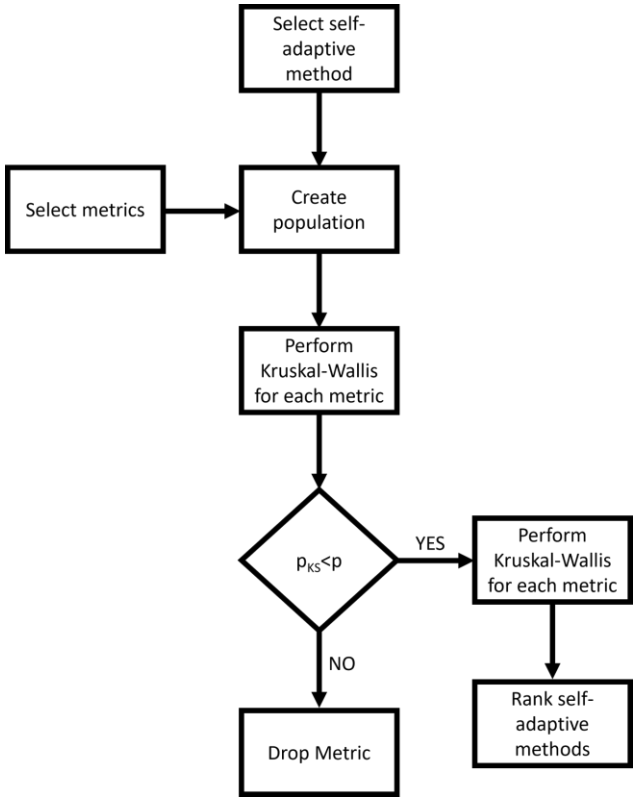


Figure 2. Block diagram describing the proposed evaluation scheme.

3. Results and Discussion

3.1 Methods performance and technical comparison

In the evaluation of self-balancing approaches, the utilization of box plots for different statistical parameters is a convenient choice. A box plot, also referred to as a box-and-whisker plot, serves as a graphical tool that presents a comprehensive five-number summary of a dataset, encompassing the minimum, the first quartile (Q1), the median, the third quartile (Q3), and the maximum. The "box" in a box plot represents the interquartile range, signifying the span between Q1 and Q3 and thus encapsulating the middle 50% of the data. The line within the box denotes the median of the dataset. Extending from the box, the "whiskers" illustrate the variability beyond the lower and upper quartiles, providing a holistic depiction of the data dispersion. Any outliers, if present, are typically depicted as individual points lying beyond the whiskers. Box plots can be drawn either vertically or horizontally and

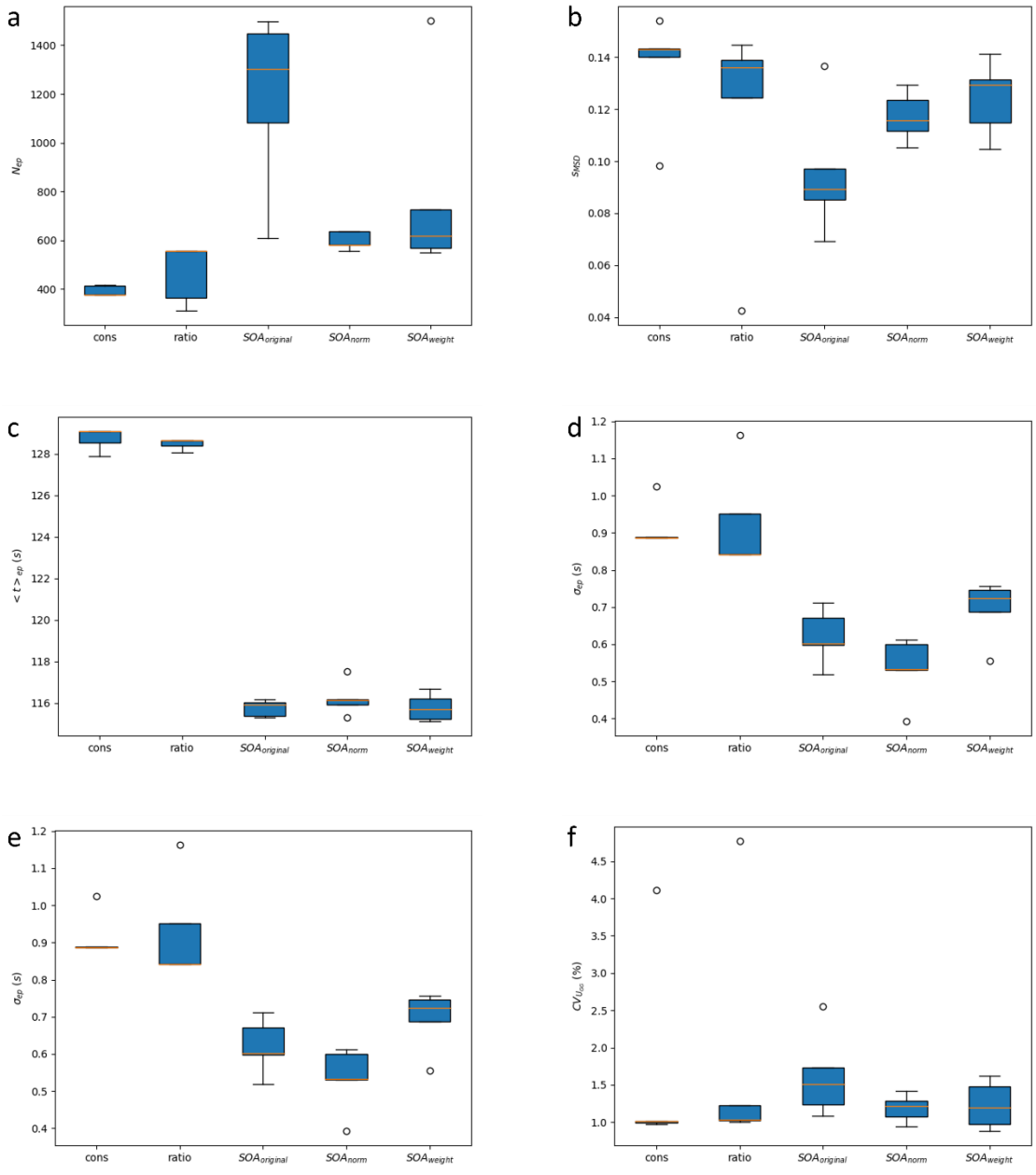
prove particularly valuable for comparing one or more datasets, identifying outliers, gauging variability, and detecting symmetry in the data.

Despite the limited population size within each approach, certain preliminary conclusions can be inferred. As depicted in Figure 2a, a noticeable difference emerges in the required number of epochs for model training across various self-balancing approaches. Notably, the constant weight approach demands the fewest number of epochs, whereas the normalized version of SoftAdapt necessitates the most. This visual representation through box plots offers a clear and intuitive means of comparing the training characteristics of different self-balancing methods, facilitating a quick grasp of the observed trends and variations. The integration of self-balancing methods within the model appears to exert an influence not only on the duration but also on the consistency of training epochs time, as depicted in Figure 2c and d. Notably, all SoftAdapt variants demand less time per epoch, and this duration is more consistent during training compared to the constant weight and ratio approaches.

When examining the absolute difference between the mean Coarse-Grained (CG) and atomistic simulated potential energies, no substantial differences emerge between the approaches (Figure 2e). The constant weight and ratio approaches seem to exhibit a lower mean value compared to the SoftAdapt methods, but the presence of outliers in both cases, coupled with a small population size, complicates the drawing of clear conclusions. This complexity is also apparent in the case of the coefficient of variation of the CG simulated potential energy ($CV_{U_{CG}}$), where, despite the lower mean value and dispersion for the constant weight and ratio approaches, outliers are present in both instances and cloud the interpretation (Figure 2f). All methods deliver a simulated temperature close to the expected value, with similar dispersion if outliers are accounted for (Figure 2g). However, the box plots of the coefficient of variation of the simulated temperature (CV_T) parameter indicate that the normalized version of SoftAdapt exhibits significantly higher oscillations in the simulated temperature relative to the other methods (Figure 2h).

In the case of the Kolmogorov-Smirnov (K-S) test results, no notable difference between the self-balancing methods is observed. Both the statistic and p-value parameters display similar behaviour across the various approaches (Figure 2i and j). Lastly, a significant difference between the self-balancing methods emerges in the Mean Square Displacement (MSD) slope parameter. The constant

weight approach results in the highest MSD slope, while the normalized version of SoftAdapt yields the lowest.



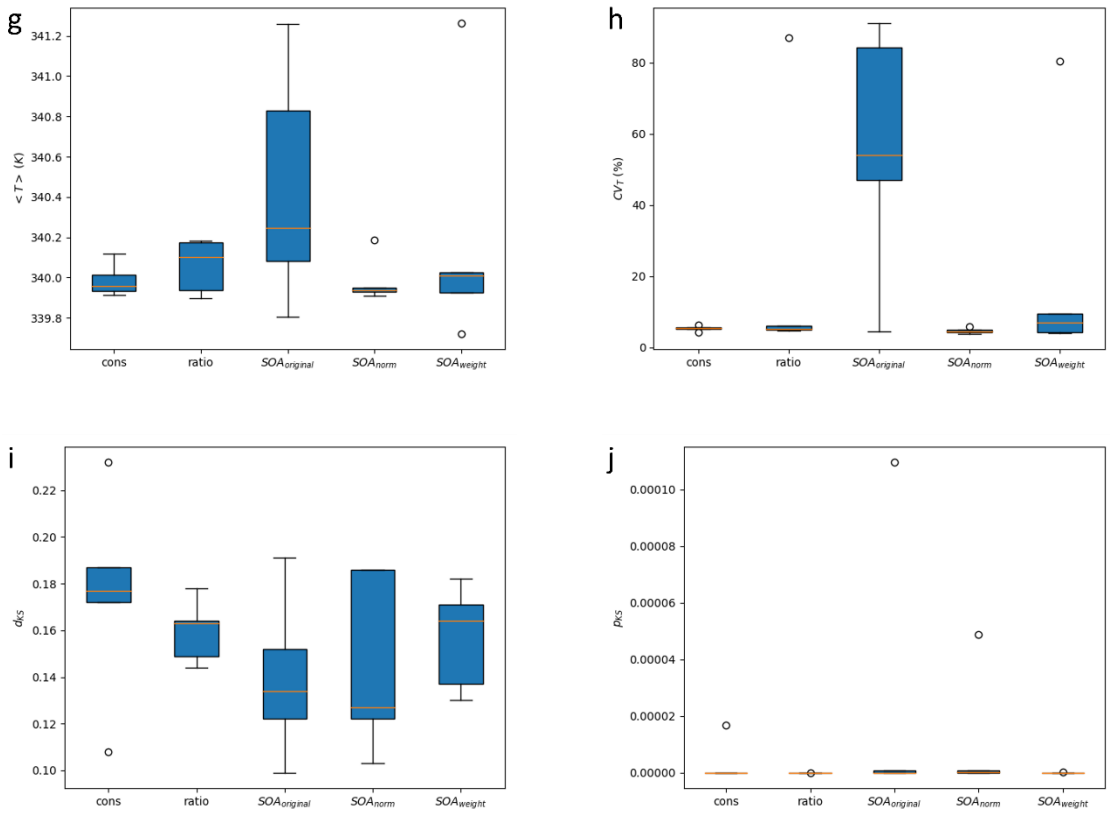


Figure 2. Box plots of the (a) number of epochs, (b) the last 30% of the MSD function, (c) mean and (d) standard deviation of the duration of training epoch, (e) absolute percentage difference between the CG and atomistic simulated potential energy and (f) its coefficient of variation, (g) mean CG simulated temperature and (h) its coefficient of variation, (i) statistic and (j) p-value of K-S test for the constant weights approach, the ratio approach, the original variant of SoftAdapt, the normalized SoftAdapt, and the weighted SoftAdapt.

3.2 Assessment of the loss coefficient self-adaptation methods through evaluation scheme

The methodology outlined in Section 2.5 facilitates a robust, repeatable, and quantifiable comparison of various self-balancing methods. This approach employs the Kruskal-Wallis test, a non-parametric statistical tool, to identify statistical parameters where at least one method exhibits distinct behaviour compared to others. Subsequently, Tukey's test, a post-hoc analysis, is utilized to pinpoint differing

methods and quantify the extent of these differences. Adopting a significance level of 0.1, the null hypothesis of the Kruskal-Wallis test is dismissed for several parameters, including the total number of epochs, mean iteration time, standard deviation of iteration time, and MSD slope parameters. This suggests that, for these specific parameters, the mean value of at least one method is statistically significantly different from the others. The results from the Kruskal-Wallis test are shown in Table 1.

Table 1. Results of the Kruskal-Wallis test for the different statistical parameters.

	N_{ep}	t_{ep}	σ_{ep}	ΔU	$CV_{U_{CG}}$	$\bar{T}$	CV_T	d_{KS}	p_{KS}	s_{MSD}
statistic	17.78	17.79	19.61	4.45	4.31	2.80	6.57	2.88	2.88	8.37
p-value	0.0014	0.0014	0.0006	0.35	0.35	0.59	0.16	0.58	0.58	0.079

Given the limited number of experiments for each approach, a more lenient p-value of 0.35 is employed for Tukey's test. This extends the scope of Tukey's test to parameters such as the absolute difference between mean CG and atomistic simulated potential energies, and the coefficient of variation for CG simulated potential energy and temperature. The results of the pairwise Tukey's tests for these parameters are compiled in Table 2, with the "rejected" column indicating null hypothesis rejection using a significance level of 0.35.

Leveraging the outcomes from Tukey's test, a ranking system was created for the different self-balancing approaches. A score was assigned to each method for each of the 10 training- or simulation-based metrics (same score if the performance of the methods was statistically equivalent, different score if they were statistically different). By summing all the assigned scores, a ranking is obtained. In terms of the total number of epochs parameter, the constant weight approach closely aligns with the ratio approach and the original variance of SoftAdapt, requiring fewer epochs compared to the normalized and weighted versions of SoftAdapt. The latter two also appear to demand fewer epochs. For the mean iteration time parameter, no significant difference is evident among the SoftAdapt methods or between the constant weight and ratio methods. Moreover, the SoftAdapt group requires less time per epoch than the constant weight and ratio group. The same trend is observed concerning the standard deviation of the time per epoch.

Table 2. Results of the pairwise Tukey's test for the parameters that present p-value equal or below 0.35 in the Kruskal-Wallis test.

		N_{ep}			t_{ep}			σ_{ep}			ΔU			CV_T			s_{MSD}		
Group A	**Group B**	Mean diff.	p-adj	Reject?	Mean diff.	p-adj	Reject?	Mean diff.	p-adj	Reject?	Mean diff.	p-adj	Reject?	Mean diff.	p-adj	Reject?	Mean diff.	p-adj	Reject?
constant	ratio	77.0	0.987	False	-0.250	0.955	False	0.014	0.999	False	16.237	0.873	False	16.237	0.873	False	-0.018	0.777	False
constant	Normalized SA	797.0	0.001	True	-12.972	0.0	True	-0.294	0.006	True	4.381	0.968	False	50.3830	0.052	True	-0.040	0.127	True
constant	Original SA	207.2	0.682	False	-12.523	0.0	True	-0.382	0.0	True	1.627	0.999	False	-0.638	1.0	False	-0.019	0.770	False
constant	Weighted SA	401.0	0.118	True	-12.943	0.0	True	-0.221	0.010	True	-0.414	1.0	False	15.701	0.886	False	-0.011	0.951	False
ratio	Normalized SA	720.0	0.002	True	-12.722	0.0	True	-0.308	0.0	True	-2.860	0.993	False	34.594	0.290	True	-0.022	0.657	False
ratio	Original SA	130.2	0.918	False	-12.273	0.0	True	-0.395	0.0	True	-5.613	0.924	False	-16.875	0.857	False	0.007	0.992	False
ratio	Weighted SA	324.0	0.273	True	-12.693	0.0	True	-0.235	0.006	True	-7.613	0.802	False	-0.536	1.0	False	0.007	0.992	False
Normalized SA	Original SA	-589.8	0.01	True	0.449	0.725	False	-0.086	0.593	False	-2.753	0.994	False	-51.468	0.048	True	0.022	0.665	False
Normalized SA	Weighted SA	-396.0	0.125	True	0.029	1.0	False	0.074	0.720	False	-4.794	0.956	False	-35.130	0.276	True	0.029	0.400	False
Original SA	Weighted SA	193.8	0.732	False	-0.420	0.770	False	0.16	0.086	True	-2.041	0.998	False	16.339	0.871	False	0.007	0.991	False

Table 3. Ranking of each self-balancing method based on individual metrics.

Method	N_{ep}	t_{ep}	σ_{ep}	ΔU	$CV_{U_{CG}}$	$\bar{T}$	CV_T	d_{KS}	p_{KS}	s_{MSD}	**Total score**
Constant	1	1	4	2	2	1	1	3	1	3	19
Ratio	2	1	3	5	3	1	2	2	1	1	21
Original SA	3	1	3	4	3	1	2	2	1	1	21
Weighted SA	4	1	2	1	1	1	2	2	1	2	17
Normalized SA	5	1	1	3	2	1	3	1	1	4	22

For the absolute difference between CG and atomistic simulated potential energy, all methods can be deemed equivalent. Regarding the coefficient of variance of the simulated temperature parameter, all methods, except the normalized SoftAdapt, are analogous, demonstrating a lower CV_T compared to the normalized SoftAdapt. Lastly, for the MSD slope parameters, all methods, except the constant weight method, can be considered equal. All these methods also have lower mean values compared to the constant weight method.

Considering these comparisons, the self-balancing methods can be ranked as follows: Weighted SoftAdapt > Constant Weights > Ratio/Original SoftAdapt > Normalized SoftAdapt

This ranking is based on the performance of each method across various parameters presented in Table 3. It is important to take though into account that all individual metrics were considered of equal significance for the extraction of the above ranking. Further refinement can be implemented by defining appropriate weightings depending on each metric importance towards an even more robust quantification of performance of each method. These findings highlight that, even without any optimization of the weighted version of SoftAdapt's hyperparameters, the SoftAdapt method is a strong candidate for further investigation. Its ability to deliver good performance without specific hyperparameter optimization suggests there might great untapped potential in the utilization of this methods.

4. Summary and Conclusions

This investigation set out to devise a robust and reproducible framework capable of automating the process of pre-screening ML models, quantifying the performance of more promising candidates, and compare different training strategies. This objective is tackled via the creation of an evaluation and ranking protocol that considers the behaviour of a model using metrics grounded on physical insights, and subsequently gauges the performance of a population of models employing statistical tests. By establishing a hierarchy between the evaluation metrics, it presents a powerful tool that can sieve out suboptimal models earlier in the process, hence, saving valuable computational resources and time. Concurrently, it quantifies the performance of promising models with a statistical rigour that adds a layer of reliability and objectivity to the selection process.

The developed methodology was implemented to evaluate the most effective among multiple self-balancing methods for a MOO problem. This problem was contextualized within the domain of self-adapting the weights associated with the two components of the loss function for a GCNN model. The underlying model used for this problem was rooted in the SchNet architecture, which was tasked here with acting as a force field for CG molecular dynamics simulations. The self-balancing methods employed in the study are the linear scalarization constant weight method, the ratio method, and three variations of SoftAdapt: original, weighted, and normalized. For the rigorous evaluation of the models under consideration, a comprehensive set of phtsics-grounded performance metrics was established.

The findings derived from the Kruskal-Wallis test reveal that the various self-balancing approaches diverge specifically in relation to five factors. These include the cumulative number of training epochs, the coefficient of variance of the simulated temperature, the average duration of each training epoch, the standard deviation of the duration of each epoch, as well as the slope of the MSD. Subsequent to the Kruskal-Wallis test, a pairwise Tukey's test was implemented to conduct a comparative analysis between the distinct approaches. Through this statistical method, it was possible to rank these different methods based on their overall performance. In accordance with the results from Tukey's test, the Weighted SoftAdapt method proved to be the most efficient approach. Following in the ranking was the Constant Weight approach. The Ratio Approach was ranked third, together with the Original SoftAdapt whereas the Normalized SoftAdapt approach ranked at the bottom. In light of these findings, it appears that the SoftAdapt method is the best candidate for further investigation.

Following the aforementioned results, several areas have been highlighted for further exploration. Firstly, the validation of these findings through an increased number of experiments would be of interest, to increase the robustness of the statistical analyses and strengthen the reliability of the conclusions drawn. Secondly, refining the evaluation parameters, by assigning varying levels of significance to each metric could reflect the real-world relevance or importance of each parameter more accurately and potentially improve the quality of conclusions drawn from the method, providing a more precise measure of the models' effectiveness. Lastly, an exploration into other self-adaptation approaches could be undertaken. Given the performance of the SoftAdapt methods in our study, further investigation into alternative approaches may reveal additional or even more effective strategies for optimizing the balance of the loss function.

Acknowledgments

E.R. gratefully acknowledges funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 101030668.

This work was supported by computational time granted from the National Infrastructures for Research and Technology S.A. (GRNET S.A.) in the National HPC facility - ARIS - under the project MULTIPOLS II (ID: 013019).

References

[1] Z. Xiang, W. Peng, X. Liu, W. Yao, Self-adaptive loss balanced Physics-informed neural networks, Neurocomputing. 496 (2022) 11–34. doi:10.1016/j.neucom.2022.05.015.

[2] D.. Jones, S. Mirrazavi, M. Tamiz, Multi-objective meta-heuristics: An overview of the current state-of-the-art, Eur. J. Oper. Res. 137 (2002) 1–9. doi:10.1016/S0377-2217(01)00123-0.

[3] O. Sener, V. Koltun, Multi-task learning as multi-objective optimization, Adv. Neural Inf. Process. Syst. 2018-Decem (2018) 527–538. http://arxiv.org/abs/1810.04650.

[4] M. Ruchte, J. Grabocka, Scalable Pareto Front Approximation for Deep Multi-Objective Learning, Proc. - IEEE Int. Conf. Data Mining, ICDM. 2021-Decem (2021) 1306–1311. doi:10.1109/ICDM51629.2021.00162.

[5] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Networks. 3 (1990) 551–560. doi:10.1016/0893-6080(90)90005-6.

[6] Y. Bengio, Practical Recommendations for Gradient-Based Training of Deep Architectures, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2012: pp. 437–478. doi:10.1007/978-3-642-35289-8_26.

[7] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.

[8] J. Snoek, H. Larochelle, R.P. Adams, Practical Bayesian optimization of machine learning algorithms, in: Adv. Neural Inf. Process. Syst., 2012: pp. 2951–2959. https://arxiv.org/abs/1206.2944v2

[9] S. Chmiela, H.E. Sauceda, K.-R. Müller, A. Tkatchenko, Towards exact molecular dynamics simulations with machine-learned force fields, Nat. Commun. 9 (2018) 3887. doi:10.1038/s41467-018-06169-2.

[10] G. Fitzgerald, J. DeJoannis, M. Meunier, Multiscale modeling of nanomaterials: Recent developments and future prospects, in: Model. Charact. Prod. Nanomater. Electron. Photonics Energy Appl., Woodhead Publishing, 2015: pp. 3–53. doi:10.1016/B978-1-78242-228-0.00001-6.

[11] E. Ricci, G. Giannakopoulos, V. Karkaletsis, D.N. Theodorou, N. Vergadou, Developing Machine-Learned Potentials for Coarse-Grained Molecular Simulations: Challenges and Pitfalls, in: Proc. 12th Hell. Conf. Artif. Intell., ACM, New York, NY, USA, 2022: pp. 1–6. doi:10.1145/3549737.3549793.

[12] E. Ricci, N. Vergadou, G.G. Vogiatzis, M.G. De Angelis, D.N. Theodorou, Molecular Simulations and Mechanistic Analysis of the Effect of CO2 Sorption on Thermodynamics, Structure, and Local Dynamics of Molten Atactic Polystyrene, Macromolecules. 53 (2020) 3669–3689. doi:10.1021/acs.macromol.0c00323.

[13] K.T. Schütt, P.J. Kindermans, H.E. Sauceda, S. Chmiela, A. Tkatchenko, K.R. Müller, SchNet: A continuous-filter convolutional neural network for modeling quantum interactions, Adv. Neural Inf. Process. Syst. 2017-Decem (2017) 992–1002.

[14] 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 (2018). doi:10.1063/1.5019779.

[15] F. Ercolessi, J.B. Adams, Interatomic Potentials from First-Principles Calculations: The Force-Matching Method, Europhys. Lett. 26 (1994) 583–588. doi:10.1209/0295-5075/26/8/005.

[16] S. Plimpton, Fast Parallel Algorithms for Short-Range Molecular Dynamics, J. Comput. Phys. 117 (1995) 1–19. doi:10.1006/jcph.1995.1039.

[17] A. Hjorth Larsen, J. Jørgen Mortensen, J. Blomqvist, I.E. Castelli, R. Christensen, M. Dułak, J. Friis, M.N. Groves, B. Hammer, C. Hargus, E.D. Hermes, P.C. Jennings, P. Bjerre Jensen, J. Kermode, J.R. Kitchin, E. Leonhard Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. Bergmann Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K.S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, K.W. Jacobsen, The atomic simulation environment—a Python library for working with atoms, J. Phys. Condens. Matter. 29 (2017) 273002. doi:10.1088/1361-648X/aa680e.