

Dynamic Training Enhances Machine Learning Potentials for Long-Lasting Molecular Dynamics

Ivan Žugec^{1,3*}, Tin Hadži Veljković⁴, Maite Alducin^{1,2*},
J. Iñaki Juaristi^{1,2,3*}

¹Centro de Física de Materiales CFM/MPC, CSIC-UPV/EHU, Paseo Manuel de Lardizabal 5, 20018, Donostia-San Sebastián, Spain.

²Donostia International Physics Center, Paseo Manuel de Lardizabal 4, Donostia-San Sebastián, 20018, Spain.

³Departamento de Polímeros y Materiales Avanzados: Física, Química y Tecnología, Facultad de Química (UPV/EHU), Apartado 1072, Donostia-San Sebastián, 20080, Spain.

⁴UvA-Bosch Delta Lab, University of Amsterdam, Amsterdam Science Park 904, Amsterdam, 1098 XH, Netherlands.

*Corresponding author(s). E-mail(s): zucec.ivan@gmail.com;
maite.alducin@ehu.eus; josebainaki.juaristi@ehu.eus;
Contributing authors: tin.hadzi@gmail.com;

Abstract

Molecular Dynamics (MD) simulations are vital for exploring complex systems in computational physics and chemistry. While machine learning methods dramatically reduce computational costs relative to ab initio methods, their accuracy in long-lasting simulations remains limited. Here we propose dynamic training (DT), a method designed to enhance model performance over extended MD simulations. Applying DT to an equivariant graph neural network (EGNN) on the challenging system of a hydrogen molecule interacting with a palladium cluster anchored to a graphene vacancy demonstrates a superior prediction accuracy compared to conventional approaches. Crucially, the DT architecture-independent design ensures its applicability across diverse machine learning potentials, making it a practical tool for advancing MD simulations.

Introduction

Molecular dynamics (MD) simulations have proven to be a powerful and versatile tool, providing valuable insights into the mechanisms behind complex phenomena [1–3]. Moreover, MD simulations also hold significant promise in optimizing and accelerating discovery of novel materials [4]. However, with current MD approaches, one often needs to choose between accuracy and simulation speed. Ab initio methods such as density functional theory (DFT) provide highly accurate predictions but are computationally expensive and scale poorly with system size. In contrast, classical force fields offer near-linear scaling with system size, but often lack accuracy and the transferability required for application to diverse systems. In recent years machine learning potentials (MLPs) have emerged as a powerful alternative to the aforementioned methods as they offer the possibility of achieving accuracy comparable to that of ab initio methods, while maintaining near-linear scaling with system size due to their predominantly local nature. MLPs come in various forms, including kernel methods [5–7], permutationally invariant polynomials [8, 9], and neural networks [10–18]. Among these, neural network potentials (NNPs) have emerged as a particularly promising avenue for creation of accurate and efficient multidimensional potential energy surfaces (PES) [19–28]. However, the application of neural network potentials is not without challenges. The requirement for extensive training data can be a substantial barrier, particularly for systems for which high-quality reference calculations are computationally expensive. Furthermore, the standard training paradigm, which focuses on minimizing the global error in single-step predictions, may not adequately capture local intricacies present in the PES. This disparity becomes apparent in applications such as MD simulations, where the cumulative effect of errors and exposure to varying temperatures can drive the system into regions where the potential is less accurately learned, frequently causing instabilities in the simulated dynamics [29].

In this work, we propose a dynamic training (DT) approach for enhancing the training of NNPs from ab-initio molecular dynamics (AIMD) simulations. We apply this strategy to an equivariant graph neural network (EGNN), resulting in what we term DT-EGNN. In contrast to conventional approaches that process data points in isolation, our method explicitly accounts for the sequential nature of MD simulations by including integration of equations of motion into the training process of a neural network. Thus, it enables direct comparison between predicted simulations and AIMD reference data, enhancing the ability of a model to capture the temporal evolution of the system. To exemplify this statement, a comprehensive and challenging dataset of AIMD simulations describing the dynamics of H_2 molecules interacting with Pd_6 clusters anchored in graphene vacancies ($\text{H}_2/\text{Pd}_6@\text{G}_{\text{vac}}$) [30] is used and we demonstrate that DT-EGNN achieves higher accuracy compared to conventional training methods while indicating promising data efficiency.

Results

Method Description

Development of machine learning models in computational chemistry is often hindered by the scarcity and high computational cost of accurate data. One way to speed up the process of creating a dataset is to use AIMD. Many widely-used datasets, including MD17 [6], Open Catalyst Project [31], and ANI-1x [32] are either partially or fully generated from AIMD. The common way of utilizing these datasets is to randomly select atomic structures for training, validation, and testing. While this approach simplifies data handling, it discards valuable temporal information present in the simulations. Given that many NNP applications revolve around performing MD simulations, it is reasonable to expect that incorporating temporal structure within the training process would enhance the NNP quality. Therefore, in this work, we propose treating each data point as a subsequence of an AIMD simulation rather than as an isolated atomic configuration. This in turn enables us to incorporate molecular dynamics directly into the training process of a NNP. In order to implement this approach, we have chosen an EGNN as our NNP architecture (see Fig. 1a). Detailed information about the EGNN architecture employed in this work is provided in the Methods section.

Starting with the data preprocessing step we extract the information about the unit cell, types of atoms, and their positions for each atomic structure in the dataset. With these quantities we can form a graph for each atomic structure. Node features of a graph are represented by a one-hot vector representing the atom element, whereas the edge features of the graph are represented by the distances between atoms. Which nodes are connected, i.e., the so-called node neighborhood, is determined by the radius graph method. Global graph features we are aiming to learn are represented by the DFT computed energy and atomic forces for a given atomic structure. However, barring practical considerations such as memory consumption, there are no limitations on how much information we can record into the data structure. It is therefore at this point that we leverage the sequential nature of the AIMD data. Each data point, be it in the training or validation set, has information on not only the DFT calculated energy and forces for the given atomic structure, but also the atomic forces for the ensuing $S_{\max} - 1$ atomic structures that follow it in the corresponding AIMD simulation. Here, S_{\max} is the predefined upper limit for the subsequence length and can, in general, be different for training and validation points. If the given atomic structure happens to be at the i -th timestep of an AIMD simulation, we take atomic force information of the next $[i + 1, \dots, i + S_{\max} - 1]$ atomic configurations calculated in AIMD. Finally, together with the integration timestep, we also store the atomic positions and velocities of the i -th structure because they provide initial conditions for the dynamics of the subsequence that takes place in the training process. This approach naturally extends to the points in the validation set as they also become subsequences of AIMD simulations. While forward passes require storing gradients for neural network optimization, validation passes do not have this memory constraint, enabling us to use subsequences that are an order of magnitude longer than those in the training set.

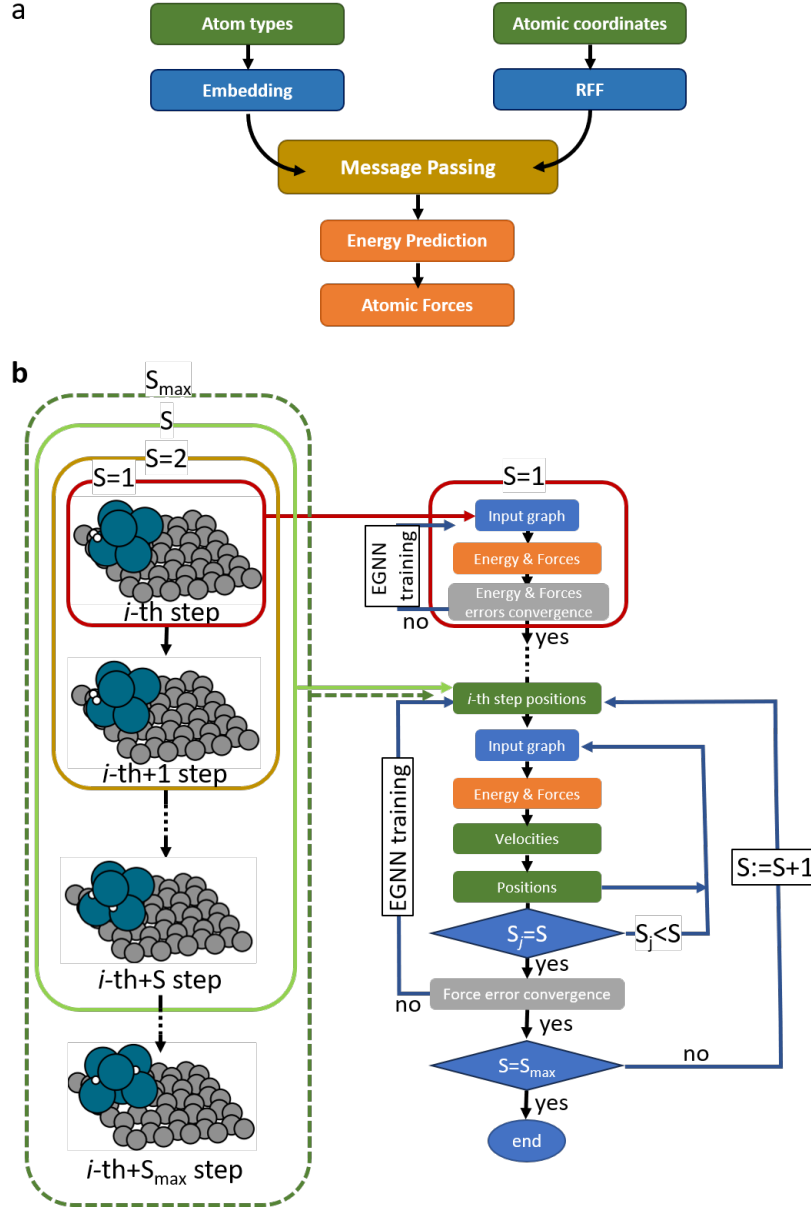


Fig. 1 DT-EGNN method. **a** Schematic representation of the EGNN architecture used in this work. Atom types undergo processing through the embedding layer while atomic coordinates are mapped to random Fourier features (RFF). These inputs are then updated via message passing layers, leading to energy prediction. Finally, atomic forces are computed as the negative gradient of the energy. **b** Schematic representation of the DT method. It starts by training a model on all the initial structures present in the training points ($S = 1$). Once the convergence criteria is met, the dynamics information is progressively expanded by increasing the subsequence length S by one. In general, training for a given S starts by predicting the atomic forces for the initial structures that, together with positions and velocities, determine the next-step atomic coordinates. These coordinates are mapped to a new input graph, enabling prediction of atomic forces and corresponding velocities. The loop continues until the desired subsequence length is reached.

In order to perform stable dynamics within the forward pass, we must ensure that the predictions on initial atomic structures are as accurate as possible. To achieve this, the training process starts with the standard practice of minimizing prediction errors of energies and forces on single atomic structures, which correspond to the initial atomic structure of each training point. It is useful to frame them as subsequences of length one ($S = 1$). Once the convergence criteria is met, instead of terminating the training process, we continue with the training by incrementing the subsequence length by one. Consequently, this makes the training iteration to consist of more parts as summarized in Fig. 1b. Similarly to $S = 1$, it starts with the prediction of the energy and forces for the initial atomic structures. These forces, in conjunction with velocities and positions, which were stored in the data preprocessing part, are utilized to derive the next-step atomic coordinates via the Euler method. These coordinates can now be used to build a graph in a similar way as described before. Upon generation of a new graph, the model predicts new atomic forces, which are then combined with the forces from the previous step to update the velocities using the velocity Verlet algorithm. This forms a loop that occurs once if the subsequence length is two, and more generally $S - 1$ times if the length of the subsequence is S . Once the model yields the predictions of energies and atomic forces for the whole subsequence, they are compared to those obtained in the corresponding AIMD calculations that were recorded during data preprocessing step. The loss function, as well as other important details regarding the training process are described in the Methods section.

Notice that the error between the model prediction and the corresponding DFT calculation of any given structure within the subsequence will depend on all the model predictions that came before it. This inevitably comes from the dynamic nature of the training process. Furthermore, all predictions are connected in a computational graph through which gradients can flow. This statement is not trivial as we show shortly, but it means that the network will be penalized for predictions that lead to high errors as simulation progresses. Another way to think of it is that subsequences act as a type of regularization that pushes the network weights to local minima more suitable for the task of performing long-lasting dynamic simulations.

Here, a remark regarding the calculation of the atomic neighborhoods in DT is in order. For subsequence lengths greater than one, each update of the atomic positions during the dynamical training can modify the underlying atomic neighborhood structures and, therefore, they must be recalculated. The typical method used to construct a neighborhood in an atomistic system is the radius graph, where neighbors comprise all atoms within a sphere of radius R from a given atom. However, this approach presents a challenge due to the nature of the greater-than-or-equal-to function, which acts as a step function in determining neighbor status. Since the step function is not differentiable, the computational graph would be disconnected leading to poor learning. While employing some kind of smoother function (e.g., sigmoid function) might seem like a natural solution, it remains unclear how to implement message passing on a continuous scale in this context without taking all atoms as neighbors. This in turn would be problematic both in terms of scaling the system and applying the methodology to periodic systems because the computational cost would be prohibitively expensive. To address this issue, we treat the atomic neighborhood as a parameter derived from

the corresponding AIMD calculations (see Supplementary Note 1). Importantly, this problem only exists during the training phase as we do not require differentiability during inference.

DT accuracy and efficiency

We validate the proposed DT method on a challenging dataset of AIMD simulations. The dataset contains 100 DFT-based microcanonical trajectories integrated with a timestep $\Delta t = 0.5$ fs, resulting in a total of 228,925 atomic configurations. In these simulations, a H_2 molecule with an initial translational energy of 0.125 eV interacts with a substrate equilibrated at 300 K. The substrate consists of a Pd_6 cluster anchored to a vacancy of a graphene layer. Different processes such as H_2 scattering after one or multiple bouncing events, as well as H_2 adsorption and H_2 dissociation on the Pd_6 cluster that often involve Pd_6 isomerization are observed [30]. Using these data, we have trained several DT-EGNN models that basically differ in the size of the employed training sets (number of training points and S_{\max} values). All these models served us to test and confirm the accuracy and efficiency of the DT method as follows. First we show that increasing the subsequence length during the training process increases the accuracy of the DT-EGNN models. Next, we demonstrate that increasing the subsequence length also reduces the errors in atomic configurations not seen during the training process, underscoring the data efficiency of the method.

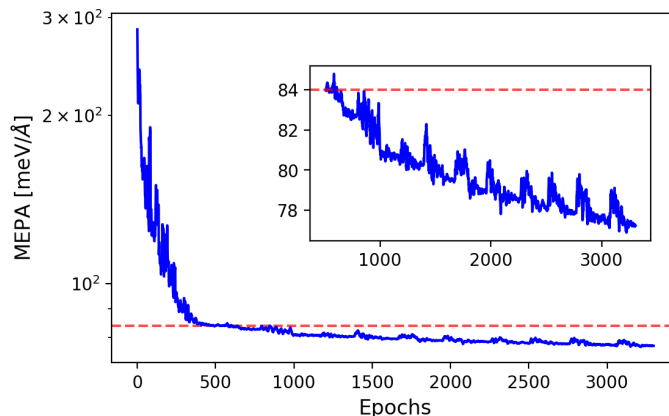


Fig. 2 Evolution of validation error. Mean error of atomic forces per atom type and per simulation step (MEPA) in logarithmic scale obtained in the validation set as a function of the training epoch (blue line). Red dashed line indicates the MEPA value in the validation set at first convergence ($S = 1$, epoch number 549). The model was trained on 226,825 points with maximum subsequence length $S_{\max} = 11$. The subsequence length in the validation set is $S_{\text{val}} = 60$. Inset: Detailed view of the validation error (linear scale) behaviour after the first convergence ($S = 1$).

Common practices often involve using validation points that closely resemble training points. In this work however, we adopt a different strategy. Our training set consists of 226,825 points and each training point has a maximum subsequence length $S_{\max} = 11$, whereas the validation set contains 2048 points, each having a subsequence

length $S_{\text{val}} = 60$. In other words, each time we evaluate our model against the validation set we perform 2048 MD simulations for 60 integration steps. It is important to point out that while the subsequence length of a training point changes during the training process every time a convergence is reached (see Fig. 1b), it stays fixed for a validation point. This provides us with a constant benchmark throughout the training process, while also giving us the opportunity to guide the training process towards a model that best aligns with the demands of a long-lasting molecular dynamics simulation. The validation error curve during the training process of one of the calculated DT-EGNN models for $\text{H}_2/\text{Pd}_6@G_{\text{vac}}$ is shown in Fig. 2. The employed error metric, defined in equation (13) in the Methods section and abbreviated as MEPA, represents the mean error of atomic forces per atom type and per simulation step. The validation curve exhibits the typical step decrease that is obtained at the beginning of the training process. The scheduler decreases the learning rate when the validation error shows no improvement over a patience period. When the patience period is exceeded at the minimum learning rate, we consider the model converged for the current subsequence length. First such convergence is marked by the red dashed line in Fig. 2. Once this criterium is met, the subsequence length is increased by one, and the learning rate reset. This reset value turned out to be a very important hyperparameter because a too small value might cause the neural network to stay trapped in the local minimum it found itself after the last convergence, whereas a too large value might destabilize the learning process. Details of all hyperparameters used in the training processes are provided in the Methods section. The novel ingredient in the DT method is the information on the system dynamics that is incorporated through the subsequence length hyperparameter. The benefit of adding such information is confirmed in Fig. 2, in which we observe that the validation error decreases with the increase in the subsequence length. However, this decrease is not strictly monotonic due to the fluctuations which largely align with the increase in learning rate following convergence. Since the validation error represents the average error per simulation step, these validation error reductions have an amplified significance. This is because the accumulation of errors at each step becomes increasingly important over longer time scales.

A key aspect of any machine learning method is its efficiency in utilizing training data. This is especially true for atomistic systems where ab-initio data are scarce and expensive to come by. We have investigated how well our approach generalizes to atomic structures unseen in the training set. Leaving the model architecture unchanged, we have selected for training a subset of 50 simulations from the original 100 AIMD simulations. This amounts to 108,019 training points with the maximum subsequence length set to $S_{\text{max}} = 15$. The validation set consists of 512 points with a subsequence length equal to $S_{\text{val}} = 120$. The 512 points used for validation are chosen from the same subset of 50 simulations used for training so that the other half of the AIMD simulations remain completely invisible to the model. In order to examine how the training subsequence length affects the model performance, we save the model with the lowest validation error at each subsequence length during the training process. Thus, as $S_{\text{max}} = 15$, we end up with fifteen models with each model representing the best performing model for its respective training subsequence length. Using a set of 50 previously unseen AIMD simulations, we randomly sampled 256 atomic structures.

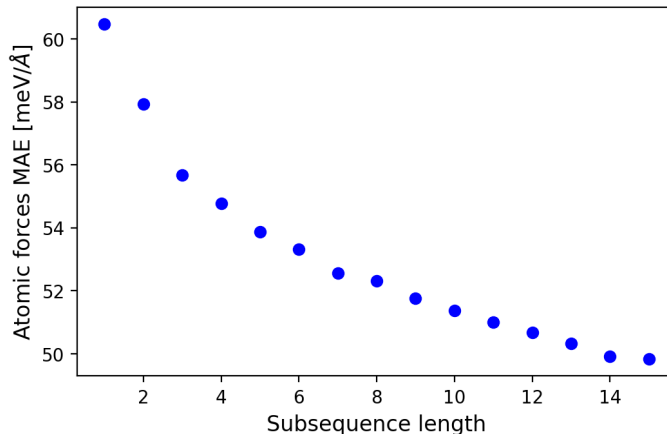


Fig. 3 DT-EGNN performance with training subsequence length. Mean absolute error of atomic forces on previously unseen atomic configurations as a function of training subsequence length S . At each S , the unseen configurations (256 points, each with subsequence length of 120 steps) are evaluated using the model with the lowest validation error at this S . The training set consists of 108,019 points, each with a maximum subsequence length $S_{\max} = 15$ and the validation set contains 512 points, each with a subsequence length $S_{\text{val}} = 120$.

Each structure served as a starting point for a 120-timesteps simulation, which is then compared to the reference AIMD data. The resulting mean absolute errors (MAE) in atomic forces for each of the fifteen models are shown in Fig. 3. We observe a consistent decrease in the prediction error with increasing subsequence length, despite the number of unique atomic structures in the training set staying constant throughout the training process.

Comparison to conventional training methods

So far we have examined the properties and performance of the DT approach in isolation. However, to establish its practical value, we compare it to conventional NNP training methods. We evaluate the DT-EGNN model trained using the complete AIMD data set with $S_{\max} = 11$ and $S_{\text{val}} = 60$, against the state-of-the-art MACE NNP and two additional NNPs (EGNN-MSE and EGNN-MEPA) trained using traditional single-point configurations for both training and validation. The latter two NNPs share identical architecture and number of parameters with our DT-EGNN model. However, EGNN-MSE has the mean squared error (MSE) of energies and forces as a loss function (see equation (16)), whereas EGNN-MEPA has the same loss function as DT-EGNN at subsequence length $S = 1$ (equations (9) to (11)). All NNPs were trained on the complete dataset of 100 AIMD simulations as detailed in the Methods section. To test the performance beyond the training simulation length of the DT-EGNN models, we selected one random atomic structure from each of the 100 trajectories, resulting in 100 test configurations. From each test configuration and for each NNP, we performed 300-timesteps MD simulations using the AIMD timestep of 0.5 fs. The atomic forces obtained at each integration step are compared to their AIMD counterparts. The corresponding MEPA values obtained with the different NNPs are summarized in

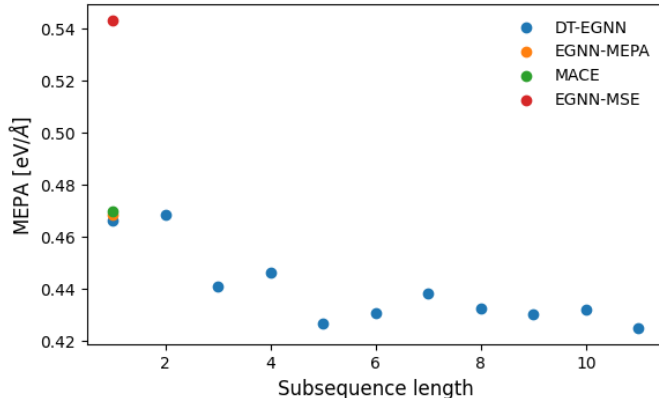


Fig. 4 DT-EGNN compared to conventionally trained NNPs. Mean error of atomic forces per atom type and per simulation step (MEPA) as a function of training subsequence length S , obtained with the DT-EGNN showcased in Fig. 2. At each S 100 test MD simulations of 300 integration steps are evaluated using the model with the lowest validation error at this S . For comparison, the MEPA obtained in equivalent MD simulations performed with MACE, EGNN-MEPA, and EGNN-MSE NNPs, which were trained in the complete data set of 100 AIMD simulations, are shown at subsequence length equal to one.

Table 1 and also plotted in Fig. 4 to highlight the DT-EGNN performance with the training subsequence length. With all EGNN models being structurally equal, it is interesting to compare how different training design choices impact the performance. Since EGNN-MEPA has the same loss function as DT-EGNN at training subsequence length equal to one, we would expect it to perform the same as the DT-EGNN model at $S = 1$. However, there is a difference in the MEPA value of 2 meV/Å that comes from the DT-EGNN validation scheme using simulations of length 60, which makes it more optimized for this task. Notably, EGNN-MSE performs the poorest. This is a consequence of having an unbalanced number of atoms between the different atoms types in this system, namely, 49 carbon atoms, six palladium atoms and only two hydrogen atoms. Minimizing the mean squared error or the mean absolute error metrics pushes the network to minimize the error on the carbon atoms, as that is the best strategy to minimize the overall mean error of a given configuration (see Supplementary Note 2). This problem is further exacerbated by the fact that hydrogen atoms cover the biggest phase space volume in our system. However, if we first take the mean of the error per each atom type and take the sum of these errors, we have a much more realistic measure of how the network is performing. Surprisingly, DT-EGNN is outperforming the state-of-the-art MACE model already at subsequence length equal to one and, as shown in Fig. 4, it continues to improve its performance as the training subsequence length increases.

In conclusion, we have introduced dynamic training, a unique methodology to train NNPs optimized for long-lasting molecular dynamics. Our approach extends beyond the conventional single-point training paradigm by incorporating sequences of consecutive atomic configurations that provide information on the system dynamics. We

NNP	MEPA [eV/Å]
EGNN-MEPA	0.468
DT-EGNN ($S = 1$)	0.466
EGNN-MSE	0.543
MACE	0.470
DT-EGNN ($S = 11$)	0.424

Table 1 Mean error of atomic forces per atom type and per simulation step (MEPA) between NNPs and DFT calculated on 300-timesteps simulations.

have validated the method on a challenging dataset of 100 ab-initio molecular dynamics simulations of H_2 impinging on a substrate consisting of a Pd_6 cluster anchored to a graphene vacancy. With the help of this dataset we have explored the relation between the accuracy of a model and the training subsequence length. We found that increasing the subsequence length during the training process does indeed improve the accuracy of the DT-EGNN. Furthermore, we have demonstrated the ability of the DT-EGNN method to generalize on atomic structures not present in the training set. Finally, comparing the DT-EGNN to conventionally trained NNPs on simulation lengths much larger than those present in the training set, we show that DT-EGNN outperforms the EGNNs with identical architecture but trained in conventional fashion. Moreover, we also show that it performs better than the state-of-the-art model MACE.

A key feature of the DT method is that it is agnostic with respect to the architecture of the model. This means that it can be applied to any other already existing NNP. Looking at the relation between the EGNN models trained by the conventional methods and DT-EGNN, it is reasonable to expect that applying the DT method to MACE would also cause an improvement in the performance relative to MACE trained with the conventional method. Another promising direction for future research lies in exploring benefits of incorporating multiple different systems in the training set with varying integration times.

We expect that the proposed method will enable researchers to conduct more accurate and efficient molecular dynamics simulations, particularly in systems in which generating training data is computationally expensive. The demonstrated improvement in data efficiency could significantly impact applications in computational chemistry and materials science, where access to high-quality training data often constitutes a bottleneck in model development.

Methods

Equivariant Graph Neural Networks

Graph Neural Networks (GNNs) are a class of deep learning models designed to process data represented as graphs. A graph is a pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where \mathcal{V} represents a set of nodes, and \mathcal{E} set of edges. This structure maps naturally to atomistic systems, where atoms and their interactions can be directly represented within the graph framework. In this work nodes are represented by a one-hot vector denoting the chemical element of an atom, while interatomic distances represent edges between the nodes. Each

prediction of a model starts with the embedding layer acting on node and edge features. Let $\mathbf{x}_0 \in \mathbb{R}^a$ be a one-hot vector representing the atom species, where a is the number of unique atom species in the system, and d_{ij} a scalar representing the interatomic distance between atoms i and j . Then embedding mappings are

$$\phi_n : \mathbb{R}^a \rightarrow \mathbb{R}^{d_n} \quad (1)$$

$$\phi_e : \mathbb{R}^1 \rightarrow \mathbb{R}^{2d_e}, \quad (2)$$

where d_n and d_e are embedding dimensions for node and edge, respectively. ϕ_n is represented by a multilayer perceptron, and ϕ_e is a random Fourier feature mapping [33] of the form,

$$\phi_e(d_{ij}) = [\sin(b_1 d_{ij}), \cos(b_1 d_{ij}), \dots, \sin(b_{d_e} d_{ij}), \cos(b_{d_e} d_{ij})], \quad (3)$$

where each parameter b_i is sampled from the normal distribution $\mathcal{N}(0, \sigma^2)$. Embedding layers are followed by message passing layers defined by the following transformations

$$\mathbf{m}_{ij}^l = \Phi_l(\mathbf{h}_i^l, \mathbf{h}_j^l, \phi_e(d_{ij})) \quad (4)$$

$$\mathbf{m}_i^l = \sum_{j \in N(i)} \mathbf{m}_{ij}^l \quad (5)$$

$$\mathbf{h}_i^{l+1} = \Phi'_l(\mathbf{h}_i^l, \mathbf{m}_i^l), \quad (6)$$

where $\mathbf{h}_i^l, \mathbf{m}_i^l \in \mathbb{R}^{d_n}$ are the i -th atom node and edge vectors at layer l . The neighborhood of an atom i denoted as $N(i)$ is calculated by a radius graph. At each layer l , update functions Φ_l , and Φ'_l are represented by multilayer perceptrons. After K message passing layers and global pooling, the final vector \mathbf{h}^K is passed through a final multilayer perceptron ψ to obtain a prediction of the potential energy of the system

$$E_{\text{pred}} = \psi(\mathbf{h}^K). \quad (7)$$

Finally, adiabatic atomic forces are obtained by taking the gradient of the predicted system energy

$$\mathbf{F}_i = -\nabla_i E_{\text{pred}}. \quad (8)$$

Training details

For the initial subsequence length $S = 1$, the loss function in DT-EGNN is defined as

$$L(S = 1) = L_{\text{energy}} + L_{\text{force}}, \quad (9)$$

where

$$L_{\text{energy}} = \frac{1}{B} \sum_b \frac{1}{N_b} |E_{\text{pred},b} - E_{\text{DFT},b}| \quad (10)$$

and

$$L_{\text{force}} = \frac{1}{B} \sum_b \sum_{a \in A_b} \left(\frac{1}{3N_{a,b}} \sum_{i=1}^{N_{a,b}} \sum_{\alpha=1}^3 |F_{\text{pred},\alpha,b}^{a,i} - F_{\text{DFT},\alpha,b}^{a,i}| \right). \quad (11)$$

Here B is the batch size; $E_{\text{pred},b}$ and $E_{\text{DFT},b}$ are the potential energies of the atomic configuration b calculated by NNP and DFT, respectively; N_b is the total number of atoms in the atomic configuration b ; A_b is the set of the different atomic species present in b , with $N_{a,b}$ being the number of atoms of the atomic species a from A_b in atomic configuration b .

For subsequence lengths larger than one, the loss function includes only force terms

$$L(S > 1) = \sum_{k=1}^S \lambda_k L_{\text{force}}^k, \quad (12)$$

where the force term for each subsequence length L_{force}^k is of the same form as in equation (11) and λ_k is equal to 50 for k equal to one, and unity for any k greater than one. Such scaling scheme proved to be crucial for the successful implementation of the method. This importance stems from the fact that during training process, initial structures are the only structures for which atomic positions exactly match those from the corresponding AIMD simulations.

Mean error of atomic forces per atom type and per simulation step (MEPA) is defined as

$$\text{MEPA} = \frac{1}{T} \sum_{i=1}^T L_{\text{force}}^i, \quad (13)$$

where T is the total number of simulation steps and L_{force}^i has the same form as in equation (11).

The loss function of the EGNN-MSE model has the following form

$$L_{\text{MSE}} = \frac{1}{B} \sum_b \frac{1}{N_b} (E_{\text{pred},b} - E_{\text{DFT},b})^2 + \frac{1}{B} \sum_b \frac{1}{3N_b} \sum_{i=1}^{N_b} \sum_{\alpha=1}^3 (F_{\text{pred},\alpha,b} - F_{\text{DFT},\alpha,b})^2. \quad (14)$$

All EGNN NNPs used in this work have identical architecture with the following model hyperparameters. Embedded node feature vectors have dimension of 128, whereas embedded edge feature vectors have dimension of 512. The standard deviation of the normal distribution from which random Fourier features were computed is equal to four. There are three message passing layers and the cutoff radius determining the neighborhood structure is equal to 5 Å. In total, each EGNN in this work has 513,281 learnable parameters.

The MACE NNP used in this work has the following model hyperparameters. Number of invariant and equivariant messages was set to 128. Cutoff radius was set to 5 Å. The rest of the MACE hyperparameters were set to default values as per mace-torch version 0.3.6. In total, the MACE model in this work has 751,888 learnable parameters.

All NNPs were trained on NVIDIA A100 GPUs in Python under version 3.12, Pytorch under version 2.4.0, and Pytorch Geometric under version 2.5.3. For the models trained with conventional methods (MACE, EGNN-MEPA, EGNN-MSE) we used 95%/5% splits resulting in 217,478 atomic configuration in the training set and 11,447 in the validation set. The learning rate, initially set to 10^{-3} , was controlled by the Pytorch’s ReduceLRonPlateau scheduler. All NNPs were trained until the minimum learning rate of 2×10^{-6} was reached except for the MACE NNP that was trained for 450 epochs. This constitutes nearly four times as many gradient updates than what is recommended in the official MACE documentation as a heuristic. Average epoch training times for the different NNPs employed in this study are summarized in Table 2. The reset value of the learning rate for DT-EGNN was set to 10^{-4} . Training batch size was equal to 128 for the NNPs trained in a single-point fashion and 32 for DT-EGNN. Validation batch size was equal to 256 and test batch size was equal to one for all NNPs used in this work. All NNPs used Adam [34] as an optimizer.

NNP	Epoch time [s]
MACE	3573
DT-EGNN (S=11)	1566
DT-EGNN (S=1)	196
EGNN-MEPA	625
EGNN-MSE	675

Table 2 Average epoch duration (in seconds) for the NNPs used in this work. DT-EGNN models were trained using 4 GPUs, whereas MACE, EGNN-MSE and EGNN-MEPA were trained using a single GPU.

Data availability

AIMD simulations and related informations used in the work have been deposited in the Figshare database under accession code <https://doi.org/10.6084/m9.figshare.28498778.v1>

Code availability

An open-source software implementation of DT-EGNN approach is available at <https://github.com/IZugec/DTEGNN>

Author contributions

Ivan Žugec conceptualization, data curation, formal analysis, investigation, methodology, software, validation, writing-original draft, writing-review & editing; **Tin Hadži Veljković** methodology, validation, writing-review & editing; **Maite Alducin** methodology, validation, supervision, writing-review & editing, funding acquisition; **J. Iñaki Juaristi** methodology, validation, supervision, writing-review & editing, funding acquisition.

Materials & Correspondence

Correspondance and requests for materials should be addressed to [Ivan Žugec](#), [Maite Alducin](#) or [J. Iñaki Juaristi](#).

Acknowledgements. Financial support was provided by the Spanish MCIN/AEI/10.13039/501100011033/, FEDER Una manera de hacer Europa (Grant No. PID2022-140163NB-I00), Gobierno Vasco-UPV/EHU (Project No. IT1569-22), and the Basque Government Education Departments' IKUR program, also co-funded by the European NextGenerationEU action through the Spanish Plan de Recuperación, Transformación y Resiliencia (PRTR). Computer resources were provided by the Donostia International Physics Center (DIPC) Supercomputing Center.

References

- [1] Rahman, A.: Correlations in the motion of atoms in liquid argon. *Phys. Rev.* **136**(2A), 405 (1964)
- [2] Karplus, M., McCammon, J.A.: Molecular dynamics simulations of biomolecules. *Nat. Struct. Mol. Biol.* **9**(9), 646–652 (2002)
- [3] Schlick, T.: *Molecular Modeling and Simulation: an Interdisciplinary Guide* vol. 2. Springer, New York (2010)
- [4] Pilania, G., Goldsmith, B.R., Yoon, M., Dongare, A.M.: Recent advances in computational materials design: methods, applications, algorithms, and informatics. *J. Mater. Sci.* **57**(23), 10471–10474 (2022)
- [5] Rupp, M., Tkatchenko, A., Müller, K.-R., Von Lilienfeld, O.A.: Fast and accurate modeling of molecular atomization energies with machine learning. *Phys. Rev. Lett.* **108**(5), 058301 (2012)
- [6] Chmiela, S., Tkatchenko, A., Sauceda, H.E., Poltavsky, I., Schütt, K.T., Müller, K.-R.: Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.* **3**(5), 1603015 (2017)
- [7] Dral, P.O.: Mlatom: A program package for quantum chemical research assisted by machine learning. *J. Comput. Chem.* **40**(26), 2339–2347 (2019)
- [8] Qu, C., Yu, Q., Bowman, J.M.: Permutationally invariant potential energy surfaces. *Annu. Rev. Phys. Chem.* **69**(1), 151–175 (2018)
- [9] Houston, P.L., Qu, C., Yu, Q., Conte, R., Nandi, A., Li, J.K., Bowman, J.M.: Pespip: Software to fit complex molecular and many-body potential energy surfaces with permutationally invariant polynomials. *J. Chem. Phys.* **158**(4) (2023)

- [10] Behler, J., Parrinello, M.: Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.* **98**(14), 146401 (2007)
- [11] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: *International Conference on Machine Learning*, pp. 1263–1272 (2017). PMLR
- [12] Smith, J.S., Isayev, O., Roitberg, A.E.: Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chem. Sci.* **8**(4), 3192–3203 (2017)
- [13] Schütt, K.T., Sauceda, H.E., Kindermans, P.-J., Tkatchenko, A., Müller, K.-R.: Schnet—a deep learning architecture for molecules and materials. *J. Chem. Phys.* **148**(24) (2018)
- [14] Zhang, Y., Xia, J., Jiang, B.: Physically motivated recursively embedded atom neural networks: Incorporating local completeness and nonlocality. *Phys. Rev. Lett.* **127**, 156002 (2021) <https://doi.org/10.1103/PhysRevLett.127.156002>
- [15] Zhang, Y., Xia, J., Jiang, B.: REANN: A PyTorch-based end-to-end multi-functional deep neural network package for molecular, reactive, and periodic systems. *J. Chem. Phys.* **156**(11), 114801 (2022) <https://doi.org/10.1063/5.0080766> https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0080766/18280834/114801_1.5.0080766.pdf
- [16] Gasteiger, J., Becker, F., Günnemann, S.: Gemnet: Universal directional graph neural networks for molecules. *Adv. Neural. Inf. Process. Syst.* **34**, 6790–6802 (2021)
- [17] Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J.P., Kornbluth, M., Molinari, N., Smidt, T.E., Kozinsky, B.: E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nat. Commun.* **13**(1), 2453 (2022)
- [18] Batatia, I., Benner, P., Chiang, Y., Elena, A.M., Kovács, D.P., Riebesell, J., Advincula, X.R., Asta, M., Baldwin, W.J., Bernstein, N., et al.: A foundation model for atomistic materials chemistry. *arXiv preprint arXiv:2401.00096* (2023)
- [19] Shakouri, K., Behler, J., Meyer, J., Kroes, G.-J.: Accurate neural network description of surface phonons in reactive gas–surface dynamics: N₂ + ru(0001). *J. Phys. Chem. Lett.* **8**(10), 2131–2136 (2017) <https://doi.org/10.1021/acs.jpcclett.7b00784> <https://doi.org/10.1021/acs.jpcclett.7b00784>. PMID: 28441867
- [20] Jiang, B., Li, J., Guo, H.: High-fidelity potential energy surfaces for gas-phase and gas–surface scattering processes from machine learning. *J. Phys. Chem. Lett.* **11**(13), 5120–5131 (2020) <https://doi.org/10.1021/acs.jpcclett.0c00989> <https://doi.org/10.1021/acs.jpcclett.0c00989>. PMID: 32517472

- [21] Zeng, Z., Wodaczek, F., Liu, K., Stein, F., Hutter, J., Chen, J., Cheng, B.: Mechanistic insight on water dissociation on pristine low-index TiO₂ surfaces from machine learning molecular dynamics simulations. *Nat. Commun.* **14**(1), 6131 (2023)
- [22] Deng, B., Zhong, P., Jun, K., Riebesell, J., Han, K., Bartel, C.J., Ceder, G.: Chgnet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nat. Mach. Intell.* **5**(9), 1031–1041 (2023)
- [23] Musaelian, A., Batzner, S., Johansson, A., Sun, L., Owen, C.J., Kornbluth, M., Kozinsky, B.: Learning local equivariant representations for large-scale atomistic dynamics. *Nat. Commun.* **14**(1), 579 (2023)
- [24] Žugec, I., Tetenoire, A., Muzas, A.S., Zhang, Y., Jiang, B., Alducin, M., Juaristi, J.I.: Understanding the photoinduced desorption and oxidation of CO on Ru(0001) using a neural network potential energy surface. *JACS Au* (2024)
- [25] Muzas, A.P.S., Serrano-Jiménez, A., Zhang, Y., Jiang, B., Juaristi, J.I., Alducin, M.: Multicoverage Study of Femtosecond Laser Induced Desorption of CO from Pd(111). *J. Phys. Chem. Lett.* **0**, 2587–2594 (2024) <https://doi.org/10.1021/acs.jpcclett.4c00026> <https://doi.org/10.1021/acs.jpcclett.4c00026>. PMID: 38416783
- [26] Žugec, I., Geilhufe, R.M., Lončarić, I.: Global machine learning potentials for molecular crystals. *J. Chem. Phys.* **160**(15) (2024)
- [27] Bochkarev, A., Lysogorskiy, Y., Drautz, R.: Graph atomic cluster expansion for semilocal interactions beyond equivariant message passing. *Phys. Rev. X* **14**(2), 021036 (2024)
- [28] Omranpour, A., Elsner, J., Lausch, K.N., Behler, J.: Machine learning potentials for heterogeneous catalysis. *ACS Catal.* **15**(3), 1616–1634 (2025) <https://doi.org/10.1021/acscatal.4c06717>
- [29] Fu, X., Wu, Z., Wang, W., Xie, T., Ketten, S., Gomez-Bombarelli, R., Jaakkola, T.: Forces are not enough: Benchmark and critical evaluation for machine learning force fields with molecular simulations. *arXiv preprint arXiv:2210.07237* (2022)
- [30] Alducin, M., Juaristi, J.I., Granja-DelRío, A., López, M.J., Alonso, J.A.: Dynamics of cluster isomerization induced by hydrogen adsorption. *J. Phys. Chem. C* **123**(24), 15236–15243 (2019)
- [31] Chanussot, L., Das, A., Goyal, S., Lavril, T., Shuaibi, M., Riviere, M., Tran, K., Heras-Domingo, J., Ho, C., Hu, W., *et al.*: Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catal.* **11**(10), 6059–6072 (2021)
- [32] Smith, J.S., Zubatyuk, R., Nebgen, B., Lubbers, N., Barros, K., Roitberg, A.E., Isayev, O., Tretiak, S.: The ani-1ccx and ani-1x data sets, coupled-cluster and

- density functional theory properties for molecules. *Sci. Data* **7**(1), 134 (2020)
- [33] Tancik, M., Srinivasan, P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. *Adv. Neural Inf. Process. Syst.* **33**, 7537–7547 (2020)
- [34] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

Supplementary Note 1

In order to benefit from performing dynamics during the training process, every operation from the first model prediction up to weight adjustments has to be differentiable. This presents a problem when building the atomic neighborhoods within the S -loop shown in Fig. 1b in the main text that stems from the binary nature of operations typically used to construct it. One way to deal with this problem is to extract neighborhood information of each atomic structure from the underlying AIMD simulations and record it during the data preprocessing step. Given sufficiently accurate predictions of atomic forces, the neighborhood structure of updated atomic positions will match the one from corresponding AIMD simulations. This is why it is very important to converge the model on subsequence length equal to one before introducing larger lengths in the training process.

Storing neighborhood structures at each step did not exceed our computational limits. However, applying the DT method to large datasets and systems with large amount of atoms could increase the computational cost. This is why we propose the following idea to alleviate the computational burden. If we assume that the changes in the neighborhood structure occur gradually along the dynamics, we can reuse the same neighborhood structure for multiple simulation steps. To examine the validity of this approximation for the system used in this work, we measure the rate of change of the neighborhood structure for atomic configurations offset by a different amount of simulation steps Δt . As a measure of a difference between sets of neighbors we use the Jaccard similarity index

$$J(N(i, t), N(i, t + \Delta t)) = \frac{|N(i, t) \cap N(i, t + \Delta t)|}{|N(i, t) \cup N(i, t + \Delta t)|}, \quad (15)$$

where $N(i, t)$ is the set of neighbors of atom i at simulation time t . The closer Δt is to zero, the closer the Jaccard index is to unity. The averaged Jaccard index for each atom over all 100 AIMD simulations of the training set is shown in Fig. 5. Indices 0 and 1 correspond to hydrogen atoms forming the H_2 molecule, indices from 2 to 50 correspond to carbon atoms, and indices 51 to 56 correspond to palladium atoms. As we would expect, hydrogen atoms have the lowest similarity index as they move the most. However, note that after nine simulation steps the averaged similarity is still above 93% for hydrogen atoms and above 97% for the rest of atoms in the system.

Moreover, training models with neighborhood updates taking place every five steps yielded comparable performance. This framework can therefore serve as a systematic method to determine appropriate neighborhood-structure-update frequencies across different systems in the dataset.

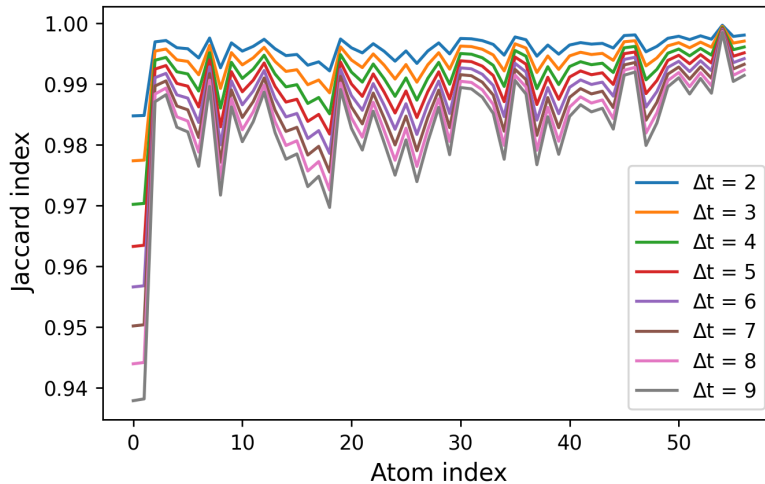


Fig. 5 Jaccard index of neighborhood structure as a function of atom index in the $\text{H}_2\text{Pd}_6@G_{\text{vac}}$ for various offsets Δt . Indices 0 and 1 correspond to hydrogen atoms forming the H_2 molecule, indices from 2 to 49 correspond to carbon atoms, and indices 50 to 56 correspond to palladium atoms.

Supplementary Note 2

Here we show how the choice of the loss function directly impacts the NNP performance during extended molecular dynamics simulations. The comparison involves three models: EGNN-MEPA, EGNN-MAE, and EGNN-MSE named after the loss functions used during training process. MEPA and MSE loss functions are already defined in the Method section of the main text and the mean absolute error (MAE) loss function is defined as

$$L_{\text{MAE}} = \frac{1}{B} \sum_b \frac{1}{N_b} |E_{\text{pred},b} - E_{\text{DFT},b}| + \frac{1}{B} \sum_b \frac{1}{3N_b} \sum_{i=1}^{N_b} \sum_{\alpha=1}^3 |F_{\text{pred},\alpha,b} - F_{\text{DFT},\alpha,b}|. \quad (16)$$

To ensure a controlled comparison, all NNPs have identical architectures and equal number of trainable parameters. Moreover, all of them were trained on the same training and validation sets. Figs. 6–8 show the components of MEPA during the training process on structures present in the validation set for each atomic species (carbon, palladium, and hydrogen) present in the system. Even though the validation error given to the scheduler during the training process was consistent with the loss function for each respective model, we calculate the MEPA for each model in order to compare

them. Models trained with MAE and MSE loss functions show better performance on carbon and palladium atoms, but perform significantly worse for hydrogen atoms. This is because MAE and MSE focus on minimizing the errors of the whole structure, while MEPA focuses on minimizing the error of each atom species.

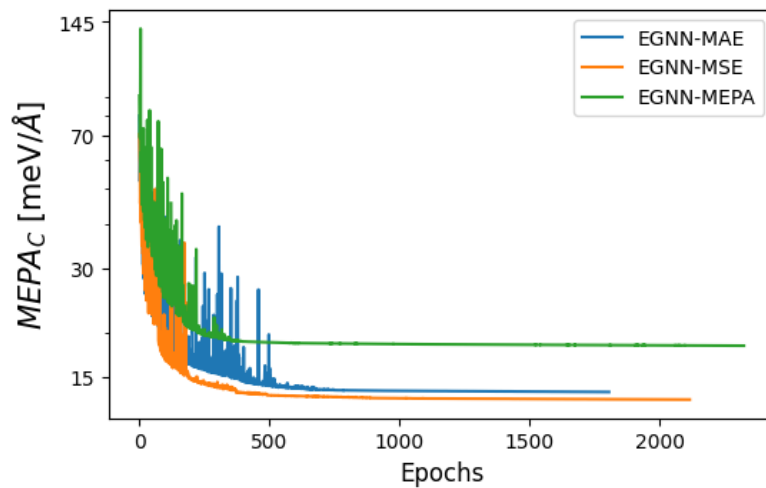


Fig. 6 Contribution to MEPA coming from carbon atoms as a function of epochs, calculated from atomic structures present in the validation set. All models, namely, EGNN-MAE (blue), EGNN-MSE (orange), and EGNN-MEPA (green), were trained on 217,478 training structures and validated on 11,447 structures.

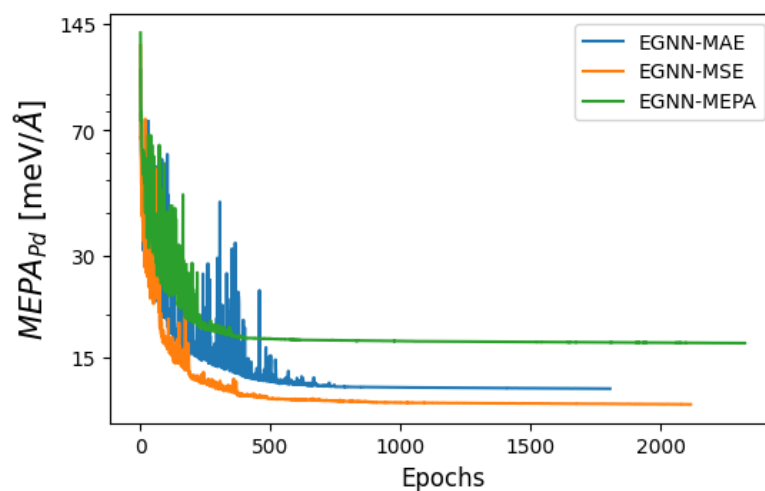


Fig. 7 Contribution to MEPA coming from palladium atoms as a function of epochs, calculated from atomic structures present in the validation set. All models, namely, EGNN-MAE (blue), EGNN-MSE (orange), and EGNN-MEPA (green), were trained on 217,478 training structures and validated on 11,447 structures.

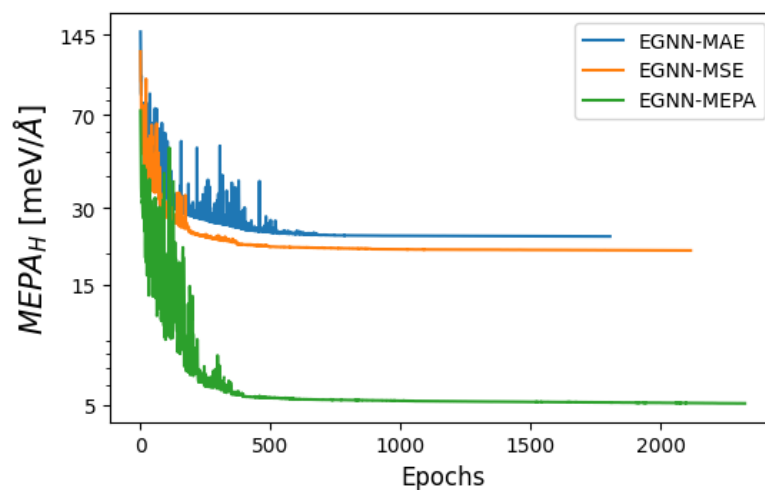


Fig. 8 Contribution to MEPA coming from hydrogen atoms as a function of epochs, calculated from atomic structures present in the validation set. All models, namely, EGNN-MAE (blue), EGNN-MSE (orange), and EGNN-MEPA (green), were trained on 217,478 training structures and validated on 11,447 structures.