PANNA 2.0: Efficient neural network interatomic potentials and new architectures

Franco Pellegrini,[1] Ruggero Lot,[1] Yusuf Shaidu,[2, 3, 1] and Emine Küçükbenli[4, 5]
[1])*Scuola Internazionale Superiore di Studi Avanzati, Trieste, Italy.*
[2])*Department of Physics, University of California Berkeley, Berkeley, California 94720, United States*
[3])*Materials Sciences Division, Lawrence Berkeley National Laboratory, Berkeley, California 94720, United States*
[4])*Nvidia Corporation, Santa Clara, CA, USA*
[5])*John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts 02138, USA*

(*Electronic mail: ekucukbenli@nvidia.com)

(*Electronic mail: pellefra@sissa.it)

(Dated: 7 November 2023)

We present the latest release of PANNA 2.0 (Properties from Artificial Neural Network Architectures), a code for the generation of neural network interatomic potentials based on local atomic descriptors and multilayer perceptrons. Built on a new back end, this new release of PANNA features improved tools for customizing and monitoring network training, better GPU support including a fast descriptor calculator, new plugins for external codes and a new architecture for the inclusion of long-range electrostatic interactions through a variational charge equilibration scheme. We present an overview of the main features of the new code, and several benchmarks comparing the accuracy of PANNA models to the state of the art, on commonly used benchmarks as well as richer datasets.

I. INTRODUCTION

In recent years, machine learning (ML) based approaches have been successfully applied to numerous problems, spanning from image and natural language processing to many areas of physics. Within the field of atomistic simulations, several approaches have been presented to leverage ML for the accurate prediction of molecular and material properties. In particular, one of the main goals has been the fast computation of energies and forces, leading to the creation of ML-based interatomic potentials (MLIPs), able to achieve the accuracy of *ab initio* methods on selected systems, for a fraction of the cost. While we refer the reader to the many available reviews and seminal works [1–5] for an exhaustive presentation of ML approaches in material science in general, we will briefly present here the main flavors of popular MLIPs present in literature, to provide a context for implementations within PANNA.

Most MLIPs are based on two approximations: i) the possibility to write the total energy of a system as a sum of atomic contributions, ii) spatial locality. This allows to roughly break down the problem into two parts: defining—or learning—a description of a local atomic environment, and learning a function to map the descriptor to the local energy. The requirement for invariance with respect to translations, rotations, and permutations of the atoms is enforced exactly by either invariant descriptors or equivariant network architectures.

The earlier methods in the field to describe the local environment relied on fixed descriptors, e.g. Behler-Parrinello[6,7] (BP) type descriptors sample the two- and three-body distribution function with local sampling functions; while the Smooth Overlap of Atomic Positions[8] (SOAP) relies on spherical harmonics to obtain a rotationally invariant description of a power of the smoothed atomic density. In these cases, ML was limited to the mapping of descriptors to atomic quantities, which relied, for example, on multilayer perceptrons[6,7] (MLPs), or kernel methods, as in the case of the Gaussian approximation potential[9,10] (GAP). These and similar approaches have been shown to achieve chemical accuracy in a host of different systems[11–14], typically given a ground truth of a few thousands configurations to train on. A limit to the generalization capacity for a given number of training points, however, is related to the architectural bias of the approach, depending both on the ML model and the descriptors. Indeed, more advanced descriptors like the Atomic Cluster Expansion[15] (ACE) were shown to obtain lower generalization errors with the same amount of data, even when the fitting was done through a simple linear model[16].

In search for a better architectural bias, more advanced message passing[17] (MP), interaction layers[18], continuous filter convolution[19], or graph neural networks (GNN) were introduced: some using vectors, angles and other geometric information to define the node functions[20–22] and some promoting the states of the networks themselves to equivariant entities based on vectors[23] or a basis of spherical irreducible representations[24–26]. While the distinction between (learned) descriptor and ML model becomes blurred in these cases[27], it has been clearly shown that the bias imposed by these architectures can lead to better generalization accuracy with the same amount of data. However, several layers of message passing can lead to very large effective receptive fields for each atom, and even when this can be avoided, each layer typically involves the use of several MLPs, leading to a larger overall computational cost. While the scaling with respect to *ab initio* is still favorable, this increased cost renders the earlier MLP approaches still valuable, especially for applications where sufficient data can be generated.

In this work, we present a new implementation of PANNA[28] (Properties from Artificial Neural Network Architectures), version 2.0, a package for the creation and deployment of MLIPs based on local descriptors and Multilayer perceptrons. Several packages have been proposed to train this type of networks, e.g. DeepMD[29,30], AENet[31], AMP[32], TorchANI[33], and SIMPLE-NN[34]. These packages are written in different languages (FORTRAN, C/C++, Python), over different back ends (TensorFlow, PyTorch), and while some of them are based on input files, others provide an API to be called from user-written code. They are all based on atomic MLPs, but they support different descriptors, from the original BP[6] to ones with modifications[7,35]; supporting different training features, network customization, learning schedules, ensemble approaches and so on. With this latest version of PANNA we hope to enrich this landscape, where variety allows more options to be explored and more needs to be met.

With respect to the previous version, the PANNA suite has been entirely rewritten to be compatible with the newest versions (2.x) of the TensorFlow[36] framework. It supports all the features of the earlier versions, and can produce models of the same accuracy with similar training performances. In addition, PANNA 2.0 supports new features, such as the computation of descriptors during training, and a new architecture to handle long range electrostatic interactions. PANNA is written in Python, and it can be simply run by supplying appropriate input configuration files. It includes several tools to customize and monitor the training, both through a graphical interface and from command line, as well as tools to import and export data from and to different external codes. Finally, PANNA models can be exported to run molecular dynamics (MD) directly in popular packages such as LAMMPS[37] and ASE[38], or to even more codes through an interface with OpenKIM[39]. The PANNA code is released under an MIT license, and it can be downloaded at Ref. 40. A thorough documentation, including a list of all input file keywords and several tutorials on how to run different example cases, is available at Ref. 41. In the following, we will present the main features of the code and the underlying theory in Sec. II, and we will report benchmarks on accuracy on different systems, speed and data scaling in Sec. III.

II. THE IMPLEMENTATION

The core of PANNA 2.0 is based on the creation of fixed-size atomic descriptors as inputs to MLPs for the computation of atomic energies, summing to the total energy of a system. Distinct architectures can be defined for each atomic species, and weights are shared between all atoms of the same species. The training procedure consists in optimizing the MLP parameters to match the energy, and forces as its derivatives, on known configurations. This optimization is performed by minimizing a *loss function* of the error, through stochastic gradient descent on small sets of examples known as *mini-batches*. In the next sections, we will highlight the options available in PANNA 2.0 for each step of this training procedure, and we will discuss specifically a new architecture that models long-range electrostatic interactions.

A. General structure

A typical MLIP training pipeline starts with the reference energies and forces being computed with density functional theory (DFT) or some other reference approach. In PANNA, we offer tools to convert the output of codes such as Quantum ESPRESSO[42], VASP[43], USPEX[44] and LAMMPS[37] to a simple human readable format. This format is completely documented, so that a user can easily create a new converter from a different code.

In the next step of the pipeline, features or descriptors need to be computed for each atom. In the previous version of PANNA, descriptors had to be precomputed and stored. This is computationally advantageous as examples are typically reused multiple times throughout the training, and since the descriptor is fixed it is possible to create it once and for all. However, this can lead to the problem of storing a large amount of data (especially when the derivatives of the descriptor are needed to compute the forces), and reading the data multiple times from storage (if they do not fit in the working memory). Depending on the hardware and hyperparameters, this can become a limiting factor for the training. For this reason, in this new version of PANNA we have introduced the option of computing the descriptors *on-the-fly* during the training itself. While more computationally expensive, this allows training on large datasets, as we show e.g. in Sec. III B, where the largest dataset would require about 500Gb of disk space to store the descriptors and their derivatives, and would bring a significant I/O cost to the training. This option is also convenient for performing quick training cycles with various descriptor types and shapes for testing purposes without having to precompute and then read/write large files. Lastly, computing the descriptors in the training loop allows to optimize their parameters during training, an option that is planned for future release. With on-th-fly computation, the descriptor becomes the most expensive part of the training, hence we have optimized this part of the code to run efficiently on GPU, leading to affordable training on a single node.

PANNA natively includes routines to compute the standard BP descriptor[6] and a modified version (mBP) as detailed in the previous version of PANNA[28]. For precomputed descriptors, the binary format used for storage is carefully documented so that descriptors computed with external routines can be adapted to the PANNA pipeline. The data loading process is handled through asynchronous parallelization over cores. In the case of on-the-fly descriptor computation, the CPU handles the example loading and nearest neighbor computation, preparing the batches in a format easily handled by the GPU, if available.

PANNA currently implements MLP type networks. The general equation for the architecture is as follows:

$$a_i^l = \sigma\left(\sum_{j=1}^{n_{l-1}} w_{ij}^l a_j^{l-1} + b_i^l\right), \tag{1}$$

where a_i is a node of layer l, w and b are weights and biases—the parameters of the network—σ is a nonlinear function, and n_l is the number of nodes in layer l (input is considered as layer 0). In PANNA, users can easily define the desired architecture, on a per-species basis, by specifying the size of the layers. The last layer is typically of size one for a single output for energy, but see Sec. II B for a different case. The activation function σ can be chosen for each layer as Gaussian, rectified linear unit (ReLU), or hyperbolic tangent, besides the linear function which is typically used for the output. Since PANNA is built on TensorFlow, the supported activation functions can be easily extended if desired, to the vast list supported by this framework.

The remaining elements of a training pipeline are the loss and learning schedule. The loss function in PANNA is made up of contributions coming from the energy and the forces errors, with an adjustable relative factor. For both, users can choose between a quadratic and exponential function of the difference between the computed and expected values, and whether to consider per configuration or per atom quantities. A further regularization term can be added to the loss function as the sum of the absolute value (L1) or square (L2) of all weights, with a chosen prefactor.

The optimization of weights and biases, i.e. training, is finally performed on mini-batches of chosen size, modifying w and b according to the gradient of the loss through the Adam[45] algorithm, with a learning rate that can be chosen as constant or exponentially decaying. As in the case of activation functions, TensorFlow backend provides PANNA with multiple options for optimizers. Additionally, freezing weights of selected layers for selected species is allowed to facilitate fine tuning or transfer learning studies. The main training loop takes advantage of native TensorFlow operations and can handle very large batches or networks with efficient linear algebra both on GPU and multiple CPUs. Although it is available in the TensorFlow back-end, distributed training on multiple nodes is not supported in PANNA at the moment.

During training, several tools within PANNA can be used to monitor the progress of the optimization. As in the previous version of the code, we support visualization of several quantities during training through TensorBoard, a browser based visualization tool. Besides loss, its components, and other figures of merit, we provide the evolution of the distribution of weights and biases for each species and layer. Based on user feedback, we have also improved the tools to monitor the training directly from the command line: a progress bar shows the current step of the training along with the loss components and figures of merit such as the root mean square error (RMSE) or mean absolute error (MAE). More importantly, we have introduced the possibility to automatically evaluate the model on a validation set on-the-fly once per epoch or at a chosen frequency, to keep track of its generalization capacity. Quickly understanding whether a model is overfitting the training set is important to stop training with bad hyperparameters, or early-stop a model that is no more improving.

After the model is trained, it can be stored as a checkpoint, and PANNA's inference tool can be used to assess its performance on a testset. The model can also be exported to a format usable in external MD codes, like LAMMPS[37] or the many other MD packages supported via OpenKIM[39]. The LAMMPS plugin included in PANNA now supports OpenMP parallelization over the cell atoms. Alternatively, the internal checkpoint format, can be imported in ASE[38] through a new calculator included in PANNA 2.0. The performance of this new plugin is tested in Sec. III B. Extension of PANNA potentials for modern MD packages such as the differentiable JAX-MD[46] will be supported in the next version.

B. Long range interactions

PANNA 2.0 supports a new method to address long-range electrostatic interactions. Most MLIP schemes rely an a locality approximation. While this is often safe to do in neutral systems, and might work for shielded charges, it is bound to fail when electrostatics plays a role in a range larger than the effective cutoff radius or in charged systems.

In recent years, to address this challenge, methods that couple a local network predicting atomic electronegativity with a system-wide charge equilibration scheme have been proposed[47–49]. Ref. 47 (implemented in 50) only deals in the electrostatic part, and Ref. 48 proposes to employ two different networks (one dependent on the other). The implementation within PANNA is based on Ref. 51 and relies on a single network to predict the coefficients for a Taylor expansion of the energy in local charges, up to the second order. This allows PANNA to compute the total energy, including electrostatics, by evaluating a single network and solving a linear charge equilibration system.

While an in-depth theoretical description and extensive benchmarks of this approach is out of the scope of this work and is presented elsewhere[51], we summarize here the fundamentals. The total energy can be approximated as the sum of a short-range contribution E_{SR}, and a long range Coulomb term $E_{\mathrm{Coul}} = \sum_{i<j} V_{ij} q_i q_j$, where q_i is the charge of atom i, and V_{ij} depends on the atomic environment. In the charge equilibration scheme, one minimizes the charge-dependent energy contribution:

$$E_{\mathrm{Qeq}} = \sum_i \left(\chi_i q_i + \frac{1}{2} J_i q_i^2 \right) + E_{\mathrm{Coul}}, \tag{2}$$

where χ_i and J_i represent the atomic electronegativities and hardnesses, respectively. This minimization is performed under the constraint $\sum_i q_i = Q$ imposing a total charge Q through a Lagrange multiplier. The novel approach implemented in PANNA rewrites the short-range energy as:

$$\begin{aligned} E_{\mathrm{SR}} = \sum_i \Big[\; & E_i^{(0)}(G_i) + \left(E_i^{(1)}(G_i) + \chi_i^0 \right) q_i + \\ & \frac{1}{2} \left(E_i^{(2)}(G_i) + J_i^0 \right) q_i^2 \Big], \end{aligned} \tag{3}$$

where χ_i^0 and J_i^0 are fixed reference values, and all terms $E_i^{(\alpha)}$ are the outputs of the network for atom i, depending on its local descriptor G_i. This expression can be seen as a Taylor expansion of the energy in the charge variable, and is implemented by tasking the neural network to produce the three Taylor expansion coefficients for each atom.

Finally, the charge equilibration can be written as the system of equations

$$\sum_j \left[\left(E_i^{(2)}(G_i) + J_i^0 \right) \delta_{ij} + V_{ij} \right] q_j + \\ + E_i^{(1)}(G_i) + \chi_i^0 + \lambda = 0, \tag{4}$$

where δ_{ij} is the Kronecker delta and λ is the Lagrange multiplier for the total charge. We introduce a new network layer to solve this system of equations, leading to the predicted local charges and the total energy. While the ground-truth local charge information can be used during training, it is not strictly necessary. This allows the use of many publicly available datasets where no local charge decomposition information has been stored. Interestingly, we find that the absence of reference local atomic charges (an approximate partition in many cases) can even improve the ability of the network to predict total energies and forces (see Sec. III C).

III. BENCHMARKS

A. rMD17

The rMD17 benchmark set consists of configurations of 10 small organic molecules, with energies and forces computed in DFT with a tight convergence threshold[52]. In recent years it has been commonly used to benchmark the data efficiency of MLIPs, specifically restricting the training set to a budget of 1000 randomly selected configurations per molecule. While this has shown the high data efficiency of equivariant GNN, the only typically reported MLP with BP type descriptors (ANI[7]) seemed to fall considerably behind. We demonstrate here the performance of PANNA with an equivalent BP type network for comparison.

The computational details are as follows: An mBP type descriptor[28] with a maximum cutoff of 5Åwas used with 24 radial bins. For the angular part, 8 bins for the average radius, and 8 for the angles, i.e. a total of 64 angular bins were used. Considering 2, 3 or 4 species, this resulted in descriptors of size 240, 456, and 736, respectively. We then trained networks with 2 hidden layers of size 256 and 128, training for 10^6 steps with learning rate 10^{-4}, and then reducing it to 10^{-6} over a further 10^6 steps. We employed quadratic loss with a force cost of 1 and a small 10^{-5} L1 regularization.

The validation MAE in energy and force components is reported in Table I, alongside the results of ANI and the state-of-the-art selected from literature considering all architectures including kernel methods. While the supremacy of equivariant GNN remains unreachable for this kind of networks (all SOTA values are found to be from various equivariant GNNs), errors from PANNA are considerably lower than ANI, for almost all molecules. This is particularly interesting as the most performant ANI network reported thus far in these benchmarks were obtained by retraining a pretrained network

TABLE I. Mean absolute error in energy (meV on the whole molecule) and forces (meV/Å per component) of different models trained on 1000 configurations from each molecule in the rMD17 dataset[52]. The ANI results are taken from Ref. 16, where ANI was either trained from scratch, column "ANI (rand)" or starting from a pretrained model, column "ANI (pre)". In the last column we report the state of the art (SOTA), i.e. the best result found for any model, giving priority to the force error, and the respective reference.

		PANNA	ANI[53] (pre)	ANI[53] (rand)	SOTA
Aspirin	E	10.6	16.6	25.4	2.2[25]
	F	32.9	40.6	75.0	6.6
Azobenzene	E	5.8	15.9	19.0	1.2[26]
	F	18.4	35.4	52.1	2.6
Benzene	E	1.0	3.3	3.4	0.3[26]
	F	5.4	10.0	17.4	0.2
Ethanol	E	2.9	2.5	7.7	0.4[25,26]
	F	16.5	13.4	45.6	2.1
Malonaldehyde	E	4.0	4.6	9.4	0.6[26]
	F	24.3	24.5	52.4	3.6
Naphthalene	E	3.0	11.3	16.0	0.2[26]
	F	13.2	29.2	52.2	0.9
Paracetamol	E	6.3	11.5	18.2	1.3[25]
	F	22.0	30.4	63.3	4.8
Salicylic acid	E	4.1	9.2	13.5	0.9[26]
	F	19.4	29.7	53.0	2.9
Toluene	E	3.9	7.7	12.6	0.5[25]
	F	15.9	24.3	52.9	1.5
Uracil	E	2.4	5.1	8.4	0.6[26]
	F	13.7	21.4	44.1	1.8

("ANI (pre)" in Table I) for increased accuracy [16]. Our results show that even for the same method and data, the differences in training and implementation can have a major impact on the final model quality, strenghtening the argument that well-written and well-documented MLIP generation packages are needed to reach a consistent quality of applications in the literature.

B. Dataset scaling: Carbon

To further assess the generalization capacity of our model, as a function of the size of the dataset, we consider a more challenging problem: a dataset with more than 60000 configurations of various allotropes of Carbon, created in a recent study using an evolutionary algorithm and the previous version of PANNA[12]. The datasets consists mostly of configurations with 16 or 24 Carbon atoms, and a few larger (200 atoms) configurations; it includes configurations under high pressure, snapshots of high temperature MD and highly defected configurations (see Ref. 12 for a complete description and construction procedure).

We split the dataset in 50000 randomly chosen configura-

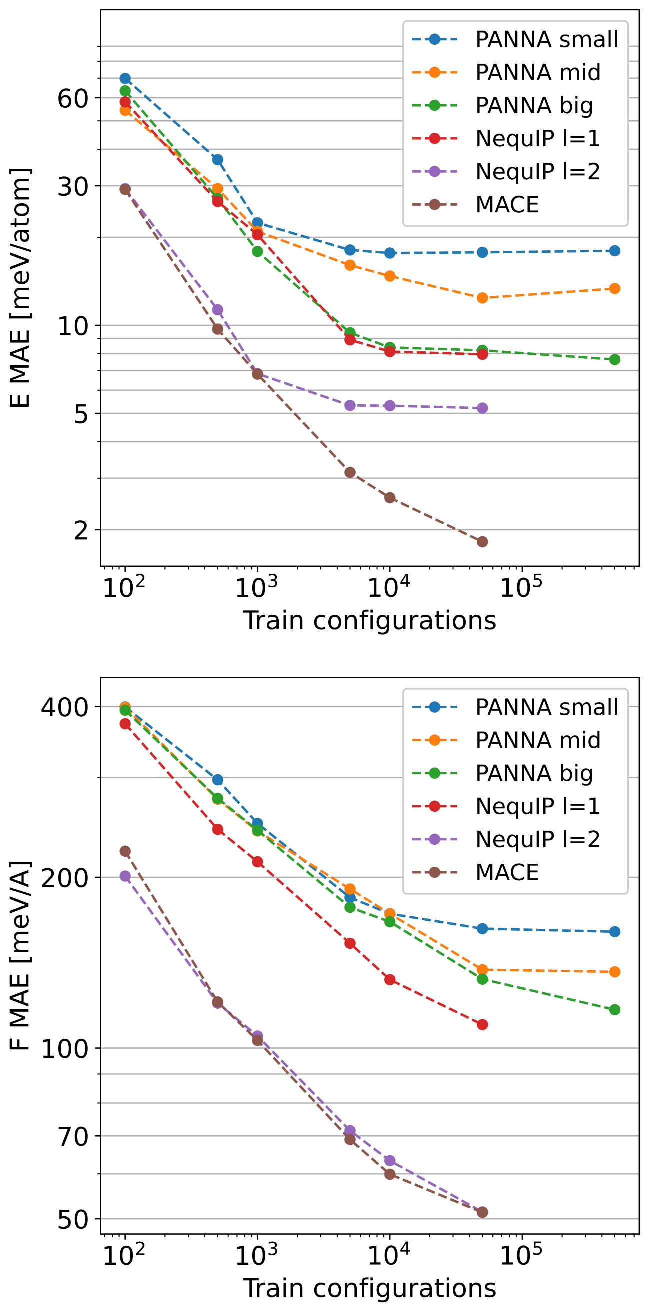

FIG. 1. Scaling of the mean absolute error in energy per atom (top) and forces per component (bottom) on the validation set, for different models, as a function of the size of the training dataset. See the main text for further details on the training, including the definition of the three architectures PANNA small, mid and big.

tions for training and we save the rest for validation. In order to generate well sampled training datasets of different sizes, we employ a farthest point clustering algorithm: we consider the cosine fingerprint distance as defined in Ref. 12, we then start from a set of a single configuration (the lowest in energy) and progressively add the configuration that is farthest. In this way we generate datasets ranging from 100 to the whole 50000 configurations. In order to have a balanced and efficient validation set, we sample 1000 configurations from the ones set aside with the same approach. We have verified that a larger sampling of 5000 configurations does not affect the results.

For PANNA, we employ a mBP descriptor with 5 Å cutoff, 24 radial bins for the two body and 8 radial and 16 angular bins for the three body terms, for a total size of 152. We train 3 different networks, a small one with two layers of sizes 64 and 32, a middle one with two layers of 256 and 128 nodes and a large one with three layers of sizes 1024, 512 and 256. Networks are trained on batches of 10 examples with a starting learning rate of 10^{-4} for a number of steps ranging from 100000 for the smallest dataset, to 6 millions for the largest one, after which a further quench to a learning rate 10^{-6} for 1 million steps is performed for all datasets larger than 1000 points.

As reference state-of-the-art models we consider NequIP[24] and MACE[25]. We train both on all datasets relying on default parameters as needed: for NequIP we consider two models, with $\ell = 1$ and $\ell = 2$, both with 4 interaction blocks, 32 features and a radial network with 8 basis functions, and 2 hidden layers of size 64. For MACE we use the standard setup with 128 even scalars and 128 odd $\ell = 1$ irreps. We use the same cutoff of 5 Å for all models as this is a commonly used cutoff and well converged for our network (see the Supplementary Material for a test), but it has to be noted that the effective receptive field for GNNs will be larger depending on the number of layers.

For all networks we train until loss convergence. An important remark needs to be made about the training dynamics: There is an apparent trade-off between energy and force components in the loss especially close to convergence and for the case of very large datasets. To tackle this, MACE training schedule implements stochastic weight averaging (SWA)[54] and increases the energy weight for the loss in the lass 20% of the training. We find that in the case of NequIP and PANNA where a standard non-averaged optimizers such as Adam[45] with fixed energy and loss weighing is used, the energy loss decreases very minimally even when increasing the relative weight of it in the loss component, even in long training scenarios, suggesting the SWA and energy re-weighing during training can be valuable. Overall we observe the force error to be more stable for all networks (given also the larger number of force data), and because the trade-off is hard to quantify for each model, we focus more on the force error during analysis.

Without significant hyperparameter tuning we do not expect these to be the best possible networks (including the one of PANNA), yet informative for a typical user experience. Fig. 1 shows the MAE of the error in energy per atom and in forces per component on the validation set. Overall, equivariant models, especially with high ℓ orders, perform better. As mentioned, the failure to improve the energy error for the larger datasets is visible for PANNA and NequIP. Among the PANNA models, we can see all models obtain similar results for small datasets, and as the dataset becomes larger the smaller model seems to reach its capacity and its performance drops.

TABLE II. Time per step per atom to run a Langevin MD on a small Carbon cell with different codes, invoked through ASE on GPU.

	PANNA small	PANNA middle	PANNA big	NequIP $\ell = 1$	NequIP $\ell = 2$	MACE
Time [ms]	0.78	0.79	0.79	3.32	4.88	3.09

Considering that MLP architectures such as BP networks are much more computationally affordable due to simple underlying tensor algebra compared to irrep algebra of equivariant GNNs, it would be desirable to find a strategy to overcome for their data-inefficiency. Here we show a potential workaround with a data augmentation experiment. Starting with the largest training dataset, for each example we create 10 copies by perturbing the atomic positions randomly with a small Gaussian noise of standard deviation 0.075Å. We then take the best MACE network and use it to compute energy and forces, obtaining a new dataset of half a million configurations, at a fraction of the DFT cost. Retraining PANNA models on this new larger dataset shows that a further improvement in accuracy can be obtained for large enough models. This "knowledge distillation" procedure is well known in the ML literature[55] and here too it proves to be a potential approach to keep less data-efficient models viable at a reduced cost.

Lastly, we consider the computational performances of these potentials when used for inference: we take one of the configurations from the dataset with 16 atoms and perform 1000 steps of Langevin dynamics at a temperature of 300 K with ASE[38] on a A100 GPU, discarding the first few steps which typically require extra setup time, not representative of the speed of the codes. To judge the natural scalability of different algorithms we refrain from using any specialized optimization techniques such as CUDA based featurization library implemented in TorchANI[33] for BP networks, as similar ones for tensor product in the equivariant GNNs are not yet widely available. Table II reports the time per step per atom of each code, as an average of 5 repetitions. We note that due to small system size, these values should be taken as upperbound, since GPU utilization is far from optimal for these system sizes. Hardware specific strategies such as multi-instance or multi-process (MIG or MPS) GPU features can potentially bring significant improvements to GPU use. These and further optimization opportunities of BP networks will be employed in future versions of PANNA. Nevertheless, without specific strategies, a raw comparison of algorithms on similar grounds show that PANNA is consistently faster, as expected from a simpler architecture. It is noteworthy that larger PANNA models are not slower, as the computational bottleneck is in the calculation of the descriptor, hence, acceleration CUDA libraries as mentioned earlier, or descriptors with lesser computational load such as ACE[15] can bring further speedup.

TABLE III. Training and validation RMSE of different quantities for the sodium chloride cluster with a total charge of +1 for long range (LR) and short range (SR) models, and models from Ref 48 (2G, 4G).

Model		Charge [me]	Energy [meV/atom]	Force [meV/Å]
SR				
	train	-	1.6	49
	test		1.6	50
LR				
$\gamma_q > 0$	train	11.5	0.4	17
	test	11.8	0.4	18
$\gamma_q = 0$	train	237.3	0.3	17
	test	238.9	0.3	18
2G	train	-	1.7	58
	test	-	1.7	57
4G	train	15.9	0.5	32
	test	15.8	0.5	33

C. Long range: NaCl clusters

In this section, we demonstrate the long range electrostatic approach implemented in PANNA on charged sodium chloride clusters. The training set is obtained from Ref. 48 and comprises of configurations of $Na_9Cl_8^+$—shown in Fig.2a—and $Na_8Cl_8^+$, obtained by removing the Na atom in the rightmost corner of Fig. 2a. Each cluster has a total charge of +1. Here we compare the accuracy of long range model within PANNA with that reported in Ref. 48.

The MLIP is constructed with mBP atomic environment descriptors of size 45, two hidden layers each with 15 nodes and an output layer with 3 nodes. As explained in Sec. II B, in PANNA reference charges can be either used as an extra target in the loss function or omitted: we present here one model with ($\gamma_q > 0$) and one without ($\gamma_q = 0$) this extra loss. We also compare with the PANNA model in the absence of long range electrostatics (SR for short range), with the same architecture but only predicting energy and forces.

In Table III, RMSE in charges, energy and forces for PANNA models are compared with the results obtained in Ref. 48 with and without long range interactions, denoted as 2G- and 4G-HDNNP, respectively. The PANNA model with long range electrostatics reaches the lowest RMSE in energy and forces irrespective of the use of atomic charges as target. It is reassuring for verification reasons that, as a baseline, The PANNA model without electrostatics attains similar RMSE in energy and forces to 2G-HDNNP[48] that also omits this contribution.

We examine in further detail the performance on the potential energy surface of these systems by computing the energy and force acting on a Na atom, indicated by 2 in Fig 2a, when moved along the arrow depicted in the same figure. Fig. 2(b) shows the force on the Na atom projected along the direction shown by the arrow. The DFT results are obtained from Ref. 48. As expected, for the PANNA model without electrostatics, we obtain similar trends to those reported in Ref. 48 for

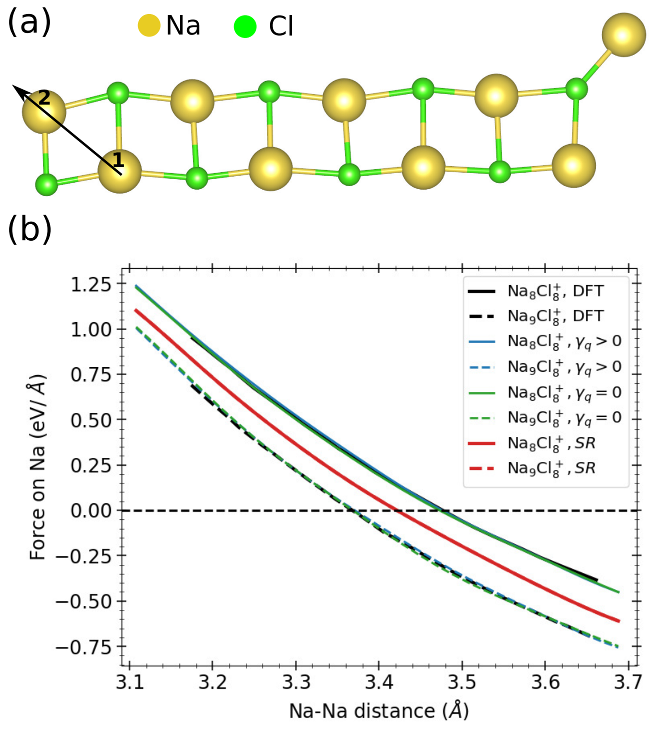

FIG. 2. Comparison of energy and forces between MLIPs and DFT. (a) Atomic structure of $Na_9Cl_8^+$. (b) Projected force on Na atom 1 in the direction of the arrow shown, as a function of the distance between Na atom 1 and 2.

the 2G-HDNNP, where the equilibrium distances for $Na_8Cl_8^+$ and $Na_9Cl_8^+$ are the same. Instead, model with long range electrostatics accurately reproduces the DFT forces for both $Na_9Cl_8^+$ and $Na_8Cl_8^+$, with and without regressing against reference charges. We note that the forces as a function of distance is smooth, suggesting stable dynamics and possibility of obtaining energy differences through integrating the forces if needed.

IV. CONCLUSION

We have given a brief overview of PANNA 2.0, the latest version. Besides the support for the new version of the Tensorflow back end—a needed upgrade to run on newer hardware where previous versions are becoming increasingly harder to obtain—this new version features several improvements aimed at simplifying the training procedure for the end user. Removing the need to precompute descriptors simplifies the exploration of new parameters, or training on very large datasets; new figures of merit and validation on-the-fly make it easier to monitor the optimization in real time. Importantly, PANNA 2.0 introduces support for long-range electrostatics, which opens the possibility to tackle charged systems that were not accessible before.

Moreover, we have shown in a series of benchmarks that while the PANNA models are not as data efficient as the newest equivariant GNN architectures, they can be more accurate than what previously reported for similar models, and they do show an accuracy-scaling power law dependence on the size of the dataset that is comparable to some equivariant models. We have also proposed the "knowledge distillation" scheme to employ the more data efficient networks to extend the training set for the less data efficient ones. Paired with fast MD plugins, these results point towards a possibility where simple architectures like PANNA can become the workhorse of large scale simulations, trading minimal accuracy for a faster computation. We will keep improving PANNA with state of the art optimization techniques such as CUDA based featurization libraries, and support for new descriptors and improved architectures to move towards making this possibility a reality in materials modeling.

V. DATA AVAILABILITY

The dataset used in Sec. III B can be downloaded at 56, while the other dataset are available in their respective publications. The best network for each architecture of those presented in Sec. III B can be downloaded at 57.

VI. SUPPLEMENTARY MATERIAL

A supplementary document presents tests on the network accuracy as a function of the cutoff radius of the descriptor, for the data presented in Sec. III B.

[1] J. Behler, "Perspective: Machine learning potentials for atomistic simulations," The Journal of chemical physics **145**, 170901 (2016).
[2] O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Machine learning force fields," Chemical Reviews **121**, 10142–10186 (2021).
[3] H. J. Kulik, T. Hammerschmidt, J. Schmidt, S. Botti, M. A. L. Marques, M. Boley, M. Scheffler, M. Todorović, P. Rinke, C. Oses, A. Smolyanyuk, S. Curtarolo, A. Tkatchenko, A. P. Bartók, S. Manzhos, M. Ihara, T. Carrington, J. Behler, O. Isayev, M. Veit, A. Grisafi, J. Nigam, M. Ceriotti, K. T. Schütt, J. Westermayr, M. Gastegger, R. J. Maurer, B. Kalita, K. Burke, R. Nagai, R. Akashi, O. Sugino, J. Hermann, F. Noé, S. Pilati, C. Draxl, M. Kuban, S. Rigamonti, M. Scheidgen, M. Esters, D. Hicks, C. Toher, P. V. Balachandran, I. Tamblyn, S. Whitelam, C. Bellinger, and L. M. Ghiringhelli, "Roadmap on machine learning in electronic structure," Electronic Structure **4**, 023004 (2022).
[4] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," Nature **559**, 547–555 (2018).
[5] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, "Finding density functionals with machine learning," Phys. Rev. Lett. **108**, 253002 (2012).
[6] J. Behler, "Atom-centered symmetry functions for constructing high-dimensional neural network potentials," The Journal of Chemical Physics **134**, 074106 (2011).
[7] J. S. Smith, O. Isayev, and A. E. Roitberg, "ANI-1: an extensible neural network potential with dft accuracy at force field computational cost," Chem. Sci. **8**, 3192–3203 (2017).
[8] S. De, A. P. Bartók, G. Csányi, and M. Ceriotti, "Comparing molecules and solids across structural and alchemical space," Phys. Chem. Chem. Phys. **18**, 13754–13769 (2016).
[9] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian approximation potentials: The accuracy of quantum mechanics, without the electrons," Phys. Rev. Lett. **104**, 136403 (2010).

[10]G. Csányi, S. Winfield, J. R. Kermode, A. De Vita, A. Comisso, N. Bernstein, and M. C. Payne, "Expressive programming for computational physics in fortran 95+," IoP Comput. Phys. Newsletter , Spring 2007 (2007).

[11]V. L. Deringer and G. Csányi, "Machine learning based interatomic potential for amorphous carbon," Physical Review B **95**, 094203 (2017).

[12]Y. Shaidu, E. Küçükbenli, R. Lot, F. Pellegrini, E. Kaxiras, and S. de Gironcoli, "A systematic approach to generating accurate neural network potentials: the case of carbon," npj Computational Materials **7**, 52 (2021).

[13]N. Artrith and J. Behler, "High-dimensional neural network potentials for metal surfaces: A prototype study for copper," Physical Review B **85**, 045439 (2012).

[14]A. C. P. Jain, D. Marchand, A. Glensk, M. Ceriotti, and W. A. Curtin, "Machine learning for metallurgy iii: A neural network potential for al-mg-si," Phys. Rev. Mater. **5**, 053805 (2021).

[15]R. Drautz, "Atomic cluster expansion for accurate and transferable interatomic potentials," Physical Review B **99**, 014104 (2019).

[16]D. P. Kovács, C. v. d. Oord, J. Kucera, A. E. Allen, D. J. Cole, C. Ortner, and G. Csányi, "Linear atomic cluster expansion force fields for organic molecules: beyond rmse," Journal of chemical theory and computation **17**, 7696–7711 (2021).

[17]J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning* (PMLR, 2017) pp. 1263–1272.

[18]N. Lubbers, J. S. Smith, and K. Barros, "Hierarchical modeling of molecular energies using a deep neural network," The Journal of Chemical Physics **148** (2018), 10.1063/1.5011181, 241715, https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/1.5011181/16654049/241715_1_online.pdf.

[19]K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "Schnet–a deep learning architecture for molecules and materials," The Journal of Chemical Physics **148**, 241722 (2018).

[20]J. Gasteiger, J. Groß, and S. Günnemann, "Directional message passing for molecular graphs," arXiv preprint arXiv:2003.03123 (2020).

[21]J. Gasteiger, F. Becker, and S. Günnemann, "Gemnet: Universal directional graph neural networks for molecules," Advances in Neural Information Processing Systems **34**, 6790–6802 (2021).

[22]M. Haghighatlari, J. Li, X. Guan, O. Zhang, A. Das, C. J. Stein, F. Heidar-Zadeh, M. Liu, M. Head-Gordon, L. Bertels, *et al.*, "Newtonnet: A newtonian message passing network for deep learning of interatomic potentials and forces," Digital Discovery **1**, 333–343 (2022).

[23]V. G. Satorras, E. Hoogeboom, and M. Welling, "E (n) equivariant graph neural networks," in *International conference on machine learning* (PMLR, 2021) pp. 9323–9332.

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

[25]I. Batatia, D. P. Kovacs, G. Simm, C. Ortner, and G. Csányi, "Mace: Higher order equivariant message passing neural networks for fast and accurate force fields," Advances in Neural Information Processing Systems **35**, 11423–11436 (2022).

[26]A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, "Learning local equivariant representations for large-scale atomistic dynamics," Nature Communications **14**, 579 (2023).

[27]I. Batatia, S. Batzner, D. P. Kovács, A. Musaelian, G. N. Simm, R. Drautz, C. Ortner, B. Kozinsky, and G. Csányi, "The design space of e (3)-equivariant atom-centered interatomic potentials," arXiv preprint arXiv:2205.06643 (2022).

[28]R. Lot, F. Pellegrini, Y. Shaidu, and E. Küçükbenli, "Panna: Properties from artificial neural network architectures," Computer Physics Communications **256**, 107402 (2020).

[29]H. Wang, L. Zhang, J. Han, and W. E, "Deepmd-kit: A deep learning package for many-body potential energy representation and molecular dynamics," Computer Physics Communications **228**, 178 – 184 (2018).

[30]J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, Y. Wang, H. Ye, P. Tuo, J. Yang, Y. Ding, Y. Li, D. Tisi, Q. Zeng, H. Bao, Y. Xia, J. Huang, K. Muraoka, Y. Wang, J. Chang, F. Yuan, S. L. Bore, C. Cai, Y. Lin, B. Wang, J. Xu, J.-X. Zhu, C. Luo, Y. Zhang, R. E. A. Goodall, W. Liang, A. K. Singh, S. Yao, J. Zhang, R. Wentzcovitch, J. Han, J. Liu, W. Jia, D. M. York, W. E, R. Car, L. Zhang, and H. Wang, "Deepmd-kit v2: A software package for deep potential models," (2023), arXiv:2304.09409 [physics.chem-ph].

[31]N. Artrith and A. Urban, "An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for tio2," Computational Materials Science **114**, 135 – 150 (2016).

[32]A. Khorshidi and A. A. Peterson, "Amp: A modular approach to machine learning in atomistic simulations," Computer Physics Communications **207**, 310 – 324 (2016).

[33]X. Gao, F. Ramezanghorbani, O. Isayev, J. S. Smith, and A. E. Roitberg, "Torchani: a free and open source pytorch-based deep learning implementation of the ani neural network potentials," Journal of chemical information and modeling **60**, 3408–3415 (2020).

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

[35]N. Artrith, A. Urban, and G. Ceder, "Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species," Physical Review B **96**, 014112 (2017).

[36]M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," (2015), software available from tensorflow.org.

[37]A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," Comp. Phys. Comm. **271**, 108171 (2022).

[38]A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, *et al.*, "The atomic simulation environment—a python library for working with atoms," Journal of Physics: Condensed Matter **29**, 273002 (2017).

[39]E. B. Tadmor, R. S. Elliott, J. P. Sethna, R. E. Miller, and C. A. Becker, "The potential of atomistic simulations and the knowledgebase of interatomic models," JOM **63**, 17 (2011).

[40]"PANNA – Properties from Artificial Neural Network Architectures," `https://gitlab.com/PANNAdevs/panna` (2023).

[41]"PANNA documentation," `https://pannadevs.gitlab.io/pannadoc/` (2023).

[42]P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. B. Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. D. Corso, S. de Gironcoli, P. Delugas, R. A. D. Jr, A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H.-Y. Ko, A. Kokalj, E. Küçükbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H.-V. Nguyen, A. O. de-la Roza, L. Paulatto, S. Poncé, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni, "Advanced capabilities for materials modelling with quantum espresso," Journal of Physics: Condensed Matter **29**, 465901 (2017).

[43]G. Kresse and J. Furthmüller, "Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set," Phys. Rev. B **54**, 11169–11186 (1996).

[44]C. W. Glass, A. R. Oganov, and N. Hansen, "Uspex—evolutionary crystal structure prediction," Computer Physics Communications **175**, 713 – 720 (2006).

[45]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv **1412.6980** (2014).

[46]S. S. Schoenholz and E. D. Cubuk, "Jax m.d. a framework for differentiable physics," in *Advances in Neural Information Processing Systems*, Vol. 33 (Curran Associates, Inc., 2020).

[47]S. A. Ghasemi, A. Hofstetter, S. Saha, and S. Goedecker, "Interatomic potentials for ionic systems with density functional accuracy based on charge densities obtained by a neural network," Phys. Rev. B **92**, 045131 (2015).

[48]T. W. Ko, J. A. Finkler, S. Goedecker, and J. Behler, "A fourth-generation

high-dimensional neural network potential with accurate electrostatics including non-local charge transfer," Nature Communications **12**, 398 (2021).

[49] L. D. Jacobson, J. M. Stevenson, F. Ramezanghorbani, D. Ghoreishi, K. Leswing, E. D. Harder, and R. Abel, "Transferable neural network potential energy surfaces for closed-shell organic molecules: Extension to ions," Journal of Chemical Theory and Computation **18**, 2354–2366 (2022), pMID: 35290063, https://doi.org/10.1021/acs.jctc.1c00821.

[50] M. Amsler, S. Rostami, H. Tahmasbi, E. R. Khajehpasha, S. Faraji, R. Rasoulkhani, and S. A. Ghasemi, "Flame: A library of atomistic modeling environments," Computer Physics Communications **256**, 107415 (2020).

[51] Y. Shaidu, F. Pellegrini, E. Küçükbenli, R. Lot, and S. de Gironcoli, "Incorporating long-range electrostatics in neural network potentials via variational charge equilibration from shortsighted ingredients," [Manuscript submitted for publication] (2023).

[52] A. S. Christensen and O. A. Von Lilienfeld, "On the role of gradients for machine learning of molecular energies and forces," Machine Learning: Science and Technology **1**, 045018 (2020).

[53] C. Devereux, J. S. Smith, K. K. Huddleston, K. Barros, R. Zubatyuk, O. Isayev, and A. E. Roitberg, "Extending the applicability of the ani deep learning molecular potential to sulfur and halogens," Journal of Chemical Theory and Computation **16**, 4192–4202 (2020).

[54] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," CoRR **abs/1803.05407** (2018), 1803.05407.

[55] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531 (2015).

[56] "Carbon dataset," `https://doi.org/10.5281/zenodo.8095485` (2023).

[57] "Carbon potentials," `https://doi.org/10.5281/zenodo.8095733` (2023).