DeePMD-GNN: A DeePMD-kit Plugin for External Graph Neural Network Potentials

Jinzhe Zeng,[†] Timothy J. Giese,[†] Duo Zhang,[‡,¶,§] Han Wang,[‖,⊥] and Darrin M. York[*,†]

Laboratory for Biomolecular Simulation Research, Institute for Quantitative Biomedicine and Department of Chemistry and Chemical Biology, Rutgers University, Piscataway, NJ 08854, USA, AI for Science Institute, Beijing, 100080, P. R. China, DP Technology, Beijing, 100080, P.R. China, Academy for Advanced Interdisciplinary Studies, Peking University, 100871, P.R. China, National Key Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, Fenghao East Road 2, Beijing 100094, P.R. China, and HEDPS, CAPT, College of Engineering, Peking University, Beijing 100871, P.R. China

E-mail: Darrin.York@rutgers.edu

[*]To whom correspondence should be addressed

[†]Laboratory for Biomolecular Simulation Research, Institute for Quantitative Biomedicine and Department of Chemistry and Chemical Biology, Rutgers University, Piscataway, NJ 08854, USA

[‡]AI for Science Institute, Beijing, 100080, P. R. China

[¶]DP Technology, Beijing, 100080, P.R. China

[§]Academy for Advanced Interdisciplinary Studies, Peking University, 100871, P.R. China

[‖]National Key Laboratory of Computational Physics, Institute of Applied Physics and Computational Mathematics, Fenghao East Road 2, Beijing 100094, P.R. China

[⊥]HEDPS, CAPT, College of Engineering, Peking University, Beijing 100871, P.R. China

Abstract

Machine learning potentials (MLPs) have revolutionized molecular simulation by providing efficient and accurate models for predicting atomic interactions. MLPs continue to advance and have had profound impact in applications that include drug discovery, enzyme catalysis and materials design. The current landscape of MLP software presents challenges due to the limited interoperability between packages, which can lead to inconsistent benchmarking practices and necessitates separate interfaces with molecular dynamics (MD) software. To address these issues, we present DeePMD-GNN, a plugin for the DeePMD-kit framework that extends its capabilities to support external graph neural network (GNN) potentials. DeePMD-GNN enables the seamless integration of popular GNN-based models, such as NequIP and MACE, within the DeePMD-kit ecosystem. Furthermore, the new software infrastructure allows GNN models to be used within combined quantum mechanical/molecular mechanical (QM/MM) applications using the range corrected ΔMLP formalism. We demonstrate the application of DeePMD-GNN by performing benchmark calculations of NequIP, MACE, and DPA-2 models developed under consistent training conditions to ensure fair comparison.

Introduction

In recent years, many machine learning potentials (MLP) have been developed to model the potential energy of atomistic systems.[1–6] These developments have resulted in numerous software packages that implement each new MLP;[7–19] however, the software is often limited to support only those MLPs developed within a particular research team. Some of the popular packages include: DeePMD-kit[7,20,21] (used to develop Deep Potential models[22–24]), SchNetPack[8,16] (used to develop for SchNet[25]), TorchANI[12] (used to develop various ANI models[26,27]), and the NequIP,[28] and MACE packages.[29] The emergence of separate software ecosystems has several disadvantages. First, it is inconvenient and inefficient to have users learn new software with the release of each new MLP. This has led to the release of support software, such as MLatom,[30] that creates workflows which try to run MLP packages in a unified way. Second, it is inconvenient and inefficient to have developers interface each MLP package with molecular dynamics (MD) software to enable their use in simulation.[14,31,32] Finally, the different infrastructures make it difficult to train the various models in a consistent manner due to differences in the optimization algorithms, the definition of the loss function, the treatment of learning rates and training steps,[4] and the availability of active learning strategies.

The present work introduces the DeePMD-GNN package, a DeePMD-kit plugin for external graph neural network potentials. The location of DeePMD-GNN within the broader DeePMD-kit software ecosystem is illustrated in Figure 1. To demonstrate its capabilities, we created plugin interfaces for two popular GNN potentials, NequIP[28] and MACE.[29] With the aid of DeePMD-GNN, these models can be trained and used in the DeePMD-kit package in the same way as other Deep Potential models to enable a wealth of applications in chemistry, biology and materials science. Furthermore, the plugin interface allows the GNN potentials to be used within range corrected QM/MM-ΔMLP applications.[33,34] Semiempirical or approximate density-functional tight-binding methods are computationally efficient, but have inherent limitations[35–37] that prevent them from achieving the accu-

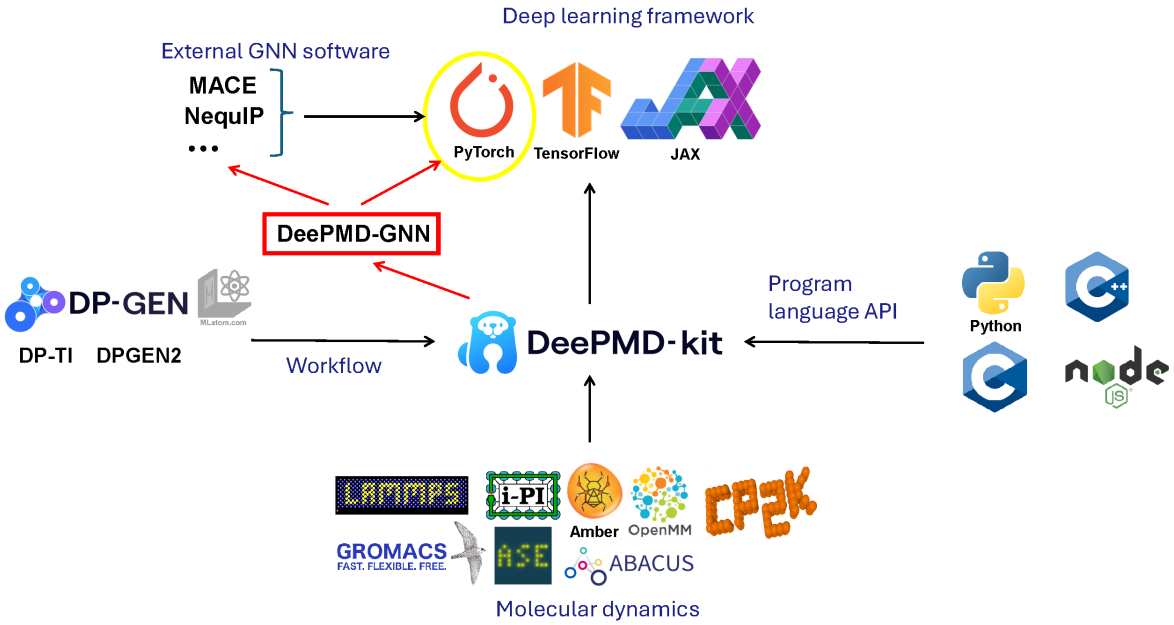

Figure 1: The location of DeePMD-GNN in the DeePMD-kit software ecosystem. The arrows indicate dependency flow, and red color indicates new software and interfaces developed in the current work. Software packages shown in the figure include: (1) DeePMD-kit[20] and DeePMD-GNN; (2) External GNN software: MACE,[29] NequIP,[28] and so on; (3) Deep learning framework: TensorFlow,[47] PyTorch,[48] and JAX;[49] (4) Molecular dynamics packages: LAMMPS,[50] i-PI,[51] Amber,[40] OpenMM,[52] CP2K,[53] GROMACS,[54] ASE,[55] and ABACUS;[56] (5) Workflow packages: DP-GEN[57] and its next generation, MLatom,[30] and DP-TI; (6) Program language API: Python, C, C++, and Node.js.

racy of much more computationally intensive *ab initio* QM methods. The range corrected QM/MM-ΔMLP strategy uses neural network to introduce short-range nonelectrostatic corrections to an inexpensive (semiempirical) QM/MM base model to reproduce target *ab initio* QM/MM energies and forces. The DeePMD-GNN plugin greatly extends the capability of recently developed interoperable software infrastructure[38,39] within Amber[40] for design of next-generation QM/MM-ΔMLP models and their application to biochemical reactions[41,42] and drug discovery.[43–45] The new software interfaces are demonstrated by comparing benchmark calculations of NequIP,[28] MACE,[29] and DPA-2[24] models developed with a consistent training strategy. The errors are compared using structures from the QDπ dataset.[46]

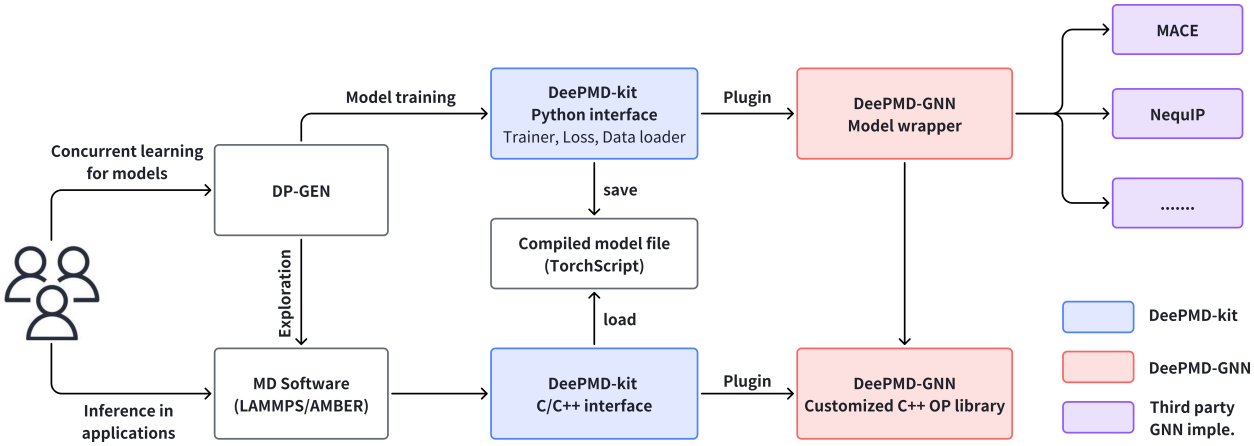

Figure 2: The software architecture of the DeePMD-GNN package. The boxes represent software components, and the arrows indicate dependency flow.

Software Description

The DeePMD-GNN package is an open-source project hosted on GitHub and licensed under LGPL-3.0. It is a Python/C++ mixed source project that is packaged with CMake[58] and scikit-build-core.[59] The software dependencies include DeePMD-kit,[20] NequIP,[28] MACE,[29] and PyTorch.[48]

Software Infrastructure: The software infrastructure used to train and apply MLPs is illustrated in Figure 2 to highlight the components provided by the DeePMD-GNN package. The diagram depicts two use cases: model generation by concurrent learning and model inference within molecular simulation applications. The DP-GEN software[57] provides an interface to the DeePMD-kit Python package[20] to train a model; that is, optimize the network parameters. The DeePMD-kit Python package is interfaced to the external GNN PyTorch software via a generic model wrapper, and the graph edges are prepared by a custom C++ operator library provided by DeePMD-GNN. When the DeePMD-kit Python package has finished the parametrization, it saves the GNN and its parameters to a serialized TorchScript model file. To use the trained model in a molecular simulation, one must run a version of the MD software that has been interfaced to the DeePMD-kit C/C++ library. The DeePMD-kit C/C++ interface can load and evaluate the saved TorchScript model file

with the aid of the C++ operator library provided by DeePMD-GNN. Consequently, it is not necessary to implement the C/C++ interface for each Python-implemented GNN model, thus simplifying the integration process. The DP-GEN software can also train models using a query–by–committee active learning strategy that involves parametrization of several network parameter sets. DP-GEN will then conduct exploration for additional training data by using the current model parameters within MD simulations. If a simulation encounters a sample that produces significant disagreement between the models, then it is saved. A subset of the saved samples are selected at random for labeling and used to parametrize the network in the next active learning iteration.

Software Features: The DeePMD-GNN package adapts the Deep Potential - Range Correction (DPRc) method[33] for use with GNN potentials for the development of range-corrected GNN models. This method is used to create ΔMLP corrections for semiempirical quantum models in QM/MM applications.[34,60] A range-corrected ΔMLP potential corrects both the QM and the nearby QM/MM interactions in a manner that produces a smooth potential energy surface as MM atoms enter and exit the vicinity of the QM region. To use GNN potentials with this approach, the MM atom energy bias is set to zero and the GNN topology excludes edges connecting pairs of MM atoms. The application and comparison of GNN and Deep Potentials range-corrected ΔMLP QM/MM applications using the DeePMD-GNN infrastructure will be the subject of forthcoming work.

Benchmark Comparison of Graph Neural Network Models

A key usage of the DeePMD-GNN package is to train and benchmark different GNN potentials in a consistent manner. As a brief demonstration, we present benchmark calculations using the DPA-2,[24] MACE,[29] and NequIP[28] potentials. These GNNs are trained for use as pure MLPs and QM/MM-ΔMLPs, where the ΔMLP is a correction to the GFN2-xTB semiempirical method.[61,62] Each model is trained consistently against the QDπ dataset.[46]

This dataset includes energies and forces calculated with ωB97M-D3(BJ)/def2-TZVPPD[63] for over 1.5 million structures that were collected from subsets of the SPICE[64] and ANI,[65,66] datasets, in addition to smaller datasets that include neutral and charged compounds covering the chemical space of 15 elements: H, Li, C, N, O, F, Na, P, S, Cl, K, Br, and I. The QDπ dataset is split into training and test sets with a 19:1 ratio.

The DPA-2 model is benchmarked at three different sizes: small (S), medium (M), and large (L). The DPA-2 (S), DPA-2 (M), and DPA-2 (L) models use 3, 6, and 12 representation-transformer (reperformer) layers, respectively. The DPA-2 (M) and DPA-2 (L) model's reperformer pair-atom representation is updated with a gated self-attention layer, whereas the DPA-2 (S) model is not. The remaining hyperparameters are the same in the model sizes. Specifically, the representation-initializer layer is encoded from the local environment within a 6 Å cutoff radius and 1 Å of smoothing; the reperformer layers are calculated with a 4 Å cutoff and 1 Å of smoothing; three-body embedding is included within a 4 Å cutoff; the embedding network consists of 3 hidden layers with 25, 50, and 100 neurons; the embedding submatrix size is 12; the fitting network consists of 3 hidden layers with 240 neurons; and the dimensions of the invariant single-atom and pair-atom representations are set to 120 and 32, respectively. Furthermore, the localized single-atom representation update mechanism excludes the self-attention layer.

The MACE model is benchmarked at two different sizes that differ only in the maximum rotational order used to communicate equivariant messages. The MACE (S) model's message passing mechanism uses a symmetry order of 0 with 256 embedding channels, and and the MACE (M) model uses a symmetry order of 1 with 128 embedding channels. The remaining hyperparameters are the same between the two models. The radial features are calculated from a 6 Å cutoff, 8 Bessel functions, and a order 5 polynomial envelope. The features were fed to a 3-layer perceptron consisting of 64 neurons/layer. The angular description of the environment is expanded in spherical harmonics to order 3. The MLP is calculated from 2 message passing layers with a correlation order of 3.

A single NequIP model is trained. The radial features are calculated from a 6 Å cutoff, 8 Bessel functions, and embedded with a 1-layer perceptron consisting of 64 neurons/layer. The MLP consists of 4 message passing layers using a maximum irreducible representation order of 2, and the hidden features were configured to use a maximum order of 1 using 32 channels and both even and odd parity.

All models are trained with the same loss function, learning rate, training steps, and floating point precision (FP32) using the Adam stochastic gradient descent method.[67] The number of training steps is set to 1 million. The learning rate exponentially decays from 1×10^{-3} to 3.51×10^{-8}. The weighted contribution of the energy errors to the loss function increases from 1 eV^{-2} and 20 eV^{-2} during the training, whereas the contributions from the force errors decrease from 100 $\mathrm{eV}^{-2}\mathrm{Å}^2$ to 1 $\mathrm{eV}^{-2}\mathrm{Å}^2$. The batch size is set to $\lceil 256/N \rceil$, where N is the number of atoms in a conformation.

Table 1: Energy (E, in unit kcal/mol) and force (F, in unit kcal/(mol·Å)) mean absolute errors (MAE) and root mean square errors (RMSE) of several GNN models against the QDπ dataset. QM-ΔMLP models prefixed by Δ use GFN2-xTB as a base QM model that are supplemented by a ΔMLP correction. Also shown are uncorrected QM models at the semiempirical GFN2-xTB level. t(infer) is the inference time (seconds) for the whole training set. The MLPs were evaluated on a single NVIDIA V100 GPU card, and the GFN2-xTB semiempirical energy was calculated on 32 AMD EPYC 7742 CPU cores.

	Training set				Test set				
Model	E MAE	E RMSE	F MAE	F RMSE	E MAE	E RMSE	F MAE	F RMSE	t(infer)
Pure MLPs									
DPA-2 (S)	3.19	7.26	3.12	8.11	3.19	5.18	3.11	5.94	2728
DPA-2 (M)	2.02	6.28	2.04	7.27	2.02	3.65	2.03	4.70	6996
DPA-2 (L)	1.73	6.07	1.77	7.08	1.75	3.32	1.77	4.43	13713
MACE (S)	2.56	7.03	2.25	7.54	2.55	4.73	2.24	5.09	2585
MACE (M)	1.98	6.62	1.74	7.17	1.97	4.09	1.74	4.54	4723
NequIP	4.49	8.88	3.65	8.79	4.46	7.12	3.64	6.80	1622
QM									
GFN2-xTB	···	···	4.36	9.58	···	···	4.38	7.84	4048
QM-ΔMLPs									
ΔDPA-2 (S)	1.27	5.70	1.25	6.64	1.27	2.58	1.25	3.82	6776
ΔDPA-2 (M)	0.98	5.57	0.99	6.53	0.98	2.31	0.99	3.63	11044
ΔDPA-2 (L)	0.89	5.54	0.92	6.50	0.89	2.23	0.92	3.58	17761
ΔMACE (S)	1.19	5.71	1.08	6.60	1.19	2.60	1.07	3.73	6633
ΔMACE (M)	0.95	5.61	0.85	6.51	0.95	2.38	0.85	3.57	8771
ΔNequIP	1.75	6.02	1.46	6.79	1.74	3.22	1.45	4.06	5670

Table 1 shows the energy and force mean absolute errors (MAE) and root mean square

errors (RMSE) of the DPA-2, MACE, and NequIP models against the QDπ dataset. The GFN2-xTB+ΔMLP models are consistently better than the pure MLP models. This observation is consistent with previous comparisons that used Deep Potential models.[22–24] Among the pure MLPs, DPA-2 (L) yields the lowest errors, and the NequIP model produces the largest errors. Among the GFN2-xTB+ΔMLP models, the ΔDPA-2 (L) and ΔNequIP models similarly produce the lowest and largest errors, respectively.

Table 1 also shows the inference time needed to calculate the whole training set with a single NVIDIA V100 GPU card and 32 AMD EPYC 7742 CPU cores, where the MLP is evaluated on the GPU and GFN2-xTB is calculated on the CPUs. The pure MLP models can be ordered from most to least expensive to evaluate: DPA-2 (L) > DPA-2 (M) > MACE (M) > DPA-2 (S) > MACE (S) > NequIP. The GFN2-xTB+ΔMLP models are about 1.5 times more expensive than the pure MLP models.

Conclusions

The DeePMD-GNN package makes a significant step forward in addressing key limitations in the current MLP software ecosystem and advancing the state-of-the-art enabling technology for molecular simulations using MLPs. By enabling the integration of external GNN potentials, such as NequIP and MACE, within the DeePMD-kit framework, it reduces the need for users to learn multiple software packages and ensures consistency in benchmarking practices. Furthermore, the DeePMD-GNN package includes infrastructure allowing the DeePMD-kit C/C++ library to read and use GNN models saved as TorchScript files. In this manner, PyTorch implementations of GNN models become immediately available in MD software that is interfaced to DeePMD-kit. The incorporation of the range-corrected ΔMLP strategy within DeePMD-GNN further allows GNN models to be used as corrections for semiempirical QM/MM calculations. We benchmarked several GNNs against the QDπ dataset to highlight the utility of DeePMD-GNN in providing a unified platform for fair and efficient

evaluation of advanced MLP methods. DeePMD-GNN is also the first plugin developed for the DeePMD-kit package, and thereby serves as an example for future development of plugins. Of particular note is the interface of DeePMD-kit with new software infrastructure in the Amber software package that enables simulations next-generation QM/MM-ΔMLP models together with a wide range of advanced alchemical free energy and free energy surface methods. We anticipate that DeePMD-GNN will facilitate a wide range of new applications that leverage GNNs to gain predictive insight into drug discovery, biocatalysis and materials design.

Data Availability

Source code for the project can be found at `https://gitlab.com/RutgersLBSR/deepmd-gnn` or `https://github.com/deepmodeling/deepmd-gnn`.

Acknowledgments

The work of Jinzhe Zeng, Timothy J. Giese, and Darrin M. York is supported by the National Institutes of Health (No. GM107485 to D.M.Y.) and the National Science Foundation (CSSI Frameworks Grant No. 2209718 to D.M.Y). The work of Han Wang is supported by the National Key R&D Program of China (Grant No. 2022YFA1004300) and the National Natural Science Foundation of China (Grant No. 12122103). Computational resources were provided by the Office of Advanced Research Computing (OARC) at Rutgers, The State University of New Jersey; the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296 (supercomputer Expanse at SDSC through allocation CHE190067)

References

(1) Behler, J. Perspective: Machine learning potentials for atomistic simulations. *J. Chem. Phys.* **2016**, *145*, 170901.

(2) Butler, K. T.; Davies, D. W.; Cartwright, H.; Isayev, O.; Walsh, A. Machine learning for molecular and materials science. *Nature* **2018**, *559*, 547–555.

(3) Noé, F.; Tkatchenko, A.; Müller, K.-R.; Clementi, C. Machine Learning for Molecular Simulation. *Annu. Rev. Phys. Chem.* **2020**, *71*, 361–390.

(4) Pinheiro Jr, M.; Ge, F.; Ferré, N.; Dral, P. O.; Barbatti, M. Choosing the right molecular machine learning potential. *Chem. Sci.* **2021**, *12*, 14396–14413.

(5) Manzhos, S.; Carrington Jr, T. Neural Network Potential Energy Surfaces for Small Molecules and Reactions. *Chem. Rev.* **2021**, *121*, 10187–10217.

(6) Zeng, J.; Cao, L.; Zhu, T. Neural network potentials. In *Quantum Chemistry in the Age of Machine Learning*; Dral, P. O., Ed.; Elsevier, 2022; Chapter 12, pp 279–294.

(7) Wang, H.; Zhang, L.; Han, J.; E, W. DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Comput. Phys. Commun.* **2018**, *228*, 178–184.

(8) Schütt, K. T.; Kessel, P.; Gastegger, M.; Nicoli, K. A.; Tkatchenko, A.; Müller, K.-R. SchNetPack: A Deep Learning Toolbox For Atomistic Systems. *J. Chem. Theory Comput.* **2019**, *15*, 448–455.

(9) Chmiela, S.; Sauceda, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. sGDML: Constructing accurate and data efficient molecular force fields using machine learning. *Comput. Phys. Commun.* **2019**, *240*, 38–45.

(10) Unke, O.; Meuwly, M. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *J. Chem. Theory Comput.* **2019**, *15*, 3678–3693.

(11) Lee, K.; Yoo, D.; Jeong, W.; Han, S. SIMPLE-NN: An efficient package for training and executing neural- network interatomic potentials. *Comput. Phys. Commun.* **2019**, *242*, 95–103.

(12) Gao, X.; Ramezanghorbani, F.; Isayev, O.; Smith, J. S.; Roitberg, A. E. TorchANI: A Free and Open Source PyTorch-Based Deep Learning Implementation of the ANI Neural Network Potentials. *J. Chem. Inf. Model.* **2020**, *60*, 3408–3415.

(13) Dral, P. O.; Ge, F.; Xue, B.-X.; Hou, Y.-F.; {Pinheiro Jr}, M.; Huang, J.; Barbatti, M. MLatom 2: An Integrative Platform for Atomistic Machine Learning. *Top. Curr. Chem. (Cham)* **2021**, *379*, 27.

(14) Singraber, A.; Behler, J.; Dellago, C. Library-Based LAMMPS Implementation of High-Dimensional Neural Network Potentials. *J. Chem. Theory Comput.* **2019**, *15*, 1827–1840.

(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.* **2022**, *156*, 114801.

(16) Schütt, K. T.; Hessmann, S. S. P.; Gebauer, N. W. A.; Lederer, J.; Gastegger, M. SchNetPack 2.0: A neural network toolbox for atomistic machine learning. *J. Chem. Phys.* **2023**, *158*, 144801.

(17) Fan, Z. et al. GPUMD: A package for constructing accurate machine-learned potentials and performing highly efficient atomistic simulations. *J. Chem. Phys.* **2022**, *157*, 114801.

(18) Novikov, I. S.; Gubaev, K.; Podryabinkin, E. V.; Shapeev, A. V. The MLIP package: moment tensor potentials with MPI and active learning. *Mach, Learn.,: Sci, Technol,* **2021**, *2*, 25002.

(19) Yanxon, H.; Zagaceta, D.; Tang, B.; Matteson, D. S.; Zhu, Q. PyXtal_FF: a python library for automated force field generation. *Mach, Learn.,: Sci, Technol,* **2021**, *2*, 27001.

(20) Zeng, J. et al. DeePMD-kit v2: A software package for deep potential models. *J. Chem. Phys.* **2023**, *159*, 054801.

(21) Liang, W.; Zeng, J.; York, D. M.; Zhang, L.; Wang, H. Learning DeePMD-Kit: A Guide to Building Deep Potential Models. In *A Practical Guide to Recent Advances in Multiscale Modeling and Simulation of Biomolecules*; Wang, Y., Zhou, R., Eds.; AIP Publishing, 2023; Chapter 6, pp 1–20.

(22) Zhang, L.; Han, J.; Wang, H.; Saidi, W.; Car, R.; E, W. End-to-end Symmetry Preserving Inter-atomic Potential Energy Model for Finite and Extended Systems. In *Advances in Neural Information Processing Systems 31*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc., 2018; pp 4436–4446.

(23) Zhang, D.; Bi, H.; Dai, F.-Z.; Jiang, W.; Liu, X.; Zhang, L.; Wang, H. Pretraining of attention-based deep learning potential model for molecular simulation. *Npj Comput. Mater* **2024**, *10*, 94.

(24) Zhang, D. et al. DPA-2: a large atomic model as a multi-task learner. *npj Comput. Mater* **2024**, *10*, 293.

(25) Schütt, K.; Sauceda, H.; Kindermans, P.; Tkatchenko, A.; Müller, K. SchNet - A Deep Learning Architecture for Molecules and Materials. *J. Chem. Phys.* **2018**, *148*, 241722.

(26) 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.* **2017**, *8*, 3192–3203.

(27) Fink, T.; Bruggesser, H.; Reymond, J.-L. Virtual Exploration of the Small-Molecule Chemical Universe below 160 Daltons. *Angew. Chem. Int. Ed.* **2005**, *44*, 1504–1508.

(28) 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.* **2022**, *13*, 2453.

(29) Batatia, I.; Kovács, D. P.; Simm, G. N. C.; Ortner, C.; Csányi, G. MACE: higher order equivariant message passing neural networks for fast and accurate force fields. 2024.

(30) Dral, P. O. et al. MLatom 3: A Platform for Machine Learning-Enhanced Computational Chemistry Simulations and Workflows. *J. Chem. Theory Comput.* **2024**, *20*, 1193–1213.

(31) Dickel, D.; Nitol, M.; Barrett, C. LAMMPS implementation of rapid artificial neural network derived interatomic potentials. *Comput. Mater. Sci.* **2021**, *196*, 110481.

(32) Chen, M. S.; Morawietz, T.; Mori, H.; Markland, T. E.; Artrith, N. AENET-LAMMPS and AENET-TINKER: Interfaces for accurate and efficient molecular dynamics simulations with machine learning potentials. *J. Chem. Phys.* **2021**, *155*, 74801.

(33) Zeng, J.; Giese, T. J.; Ekesan, Ş.; York, D. M. Development of Range-Corrected Deep Learning Potentials for Fast, Accurate Quantum Mechanical/Molecular Mechanical Simulations of Chemical Reactions in Solution. *J. Chem. Theory Comput.* **2021**, *17*, 6993–7009.

(34) Giese, T. J.; Zeng, J.; Ekesan, Ş.; York, D. M. Combined QM/MM, Machine Learning Path Integral Approach to Compute Free Energy Profiles and Kinetic Isotope Effects in RNA Cleavage Reactions. *J. Chem. Theory Comput.* **2022**, *18*, 4304–4317.

(35) Giese, T. J.; York, D. M. Improvement of semiempirical response properties with charge-dependent response density. *J. Chem. Phys.* **2005**, *123*, 164108.

(36) Giese, T. J.; York, D. M. Density-functional expansion methods: grand challenges. *Theor. Chem. Acc.* **2012**, *131*, 1145.

(37) Kaminski, S.; Giese, T. J.; Gaus, M.; York, D. M.; Elstner, M. Extended Polarization in Third-Order SCC-DFTB from Chemical-Potential Equalization. *J. Phys. Chem. A* **2012**, *116*, 9131–9141.

(38) Giese, T. J.; Zeng, J.; Lerew, L.; McCarthy, E.; Tao, Y.; Ekesan, Ş.; York, D. M. Software Infrastructure for Next-Generation QM/MM-ΔMLP Force Fields. *J. Phys. Chem. B* **2024**, *128*, 6257–6271.

(39) Tao, Y.; Giese, T. J.; Şölen Ekesan,; Zeng, J.; Aradi, B.; Hourahine, B.; Aktulga, H. M.; Götz, A. W.; Merz, Jr, K. M.; York, D. M. Amber free energy tools: Interoperable software for free energy simulations using generalized quantum mechanical/molecular mechanical and machine learning potentials. *J. Chem. Phys.* **2024**, *160*, 224104.

(40) Case, D. A. et al. AmberTools. *J. Chem. Inf. Model.* **2023**, *63*, 6183–6191.

(41) Tao, Y.; Giese, T. J.; York, D. M. Electronic and Nuclear Quantum Effects on Proton Transfer Reactions of Guanine–Thymine (G-T) Mispairs Using Combined Quantum Mechanical/Molecular Mechanical and Machine Learning Potentials. *Molecules* **2024**, *29*, 2703.

(42) Wilson, T. J.; McCarthy, E.; Ekesan, Ş.; Giese, T. J.; Li, N.-S.; Huang, L.; Piccirilli, J. A.; York, D. M.; Lilley, D. M. J. The Role of General Acid Catalysis in the Mechanism of an Alkyl Transferase Ribozyme. *ACS Catal.* **2024**, *14*, 15294–15305.

(43) York, D. M. Modern Alchemical Free Energy Methods for Drug Discovery Explained. *ACS Phys. Chem. Au* **2023**, *3*, 478–491.

(44) Zeng, J.; Tao, Y.; Giese, T. J.; York, D. M. QDπ: A Quantum Deep Potential Interaction Model for Drug Discovery. *J. Chem. Theory Comput.* **2023**, *19*, 1261–1275.

(45) Zeng, J.; Tao, Y.; Giese, T. J.; York, D. M. Modern semiempirical electronic structure methods and machine learning potentials for drug discovery: Conformers, tautomers, and protonation states. *J. Chem. Phys.* **2023**, *158*, 124110.

(46) Zeng, J.; Giese, T. J.; Götz, A.; York, D. M. The QDπ dataset, training data for drug-like molecules and biopolymer fragments and their interactions. unpublished.

(47) Abadi, M. et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015; `https://www.tensorflow.org/`, Software available from tensorflow.org.

(48) Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019; `https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf`.

(49) Bradbury, J.; Frostig, R.; Hawkins, P.; Johnson, M. J.; Leary, C.; Maclaurin, D.; Necula, G.; Paszke, A.; VanderPlas, J.; Wanderman-Milne, S.; Zhang, Q. JAX: composable transformations of Python+NumPy programs. 2018; `http://github.com/jax-ml/jax`.

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

(51) Litman, Y. et al. i-PI 3.0: A flexible and efficient framework for advanced atomistic simulations. *J. Chem. Phys.* **2024**, *161*, 062504.

(52) Eastman, P. et al. OpenMM 8: Molecular Dynamics Simulation with Machine Learning Potentials. *J. Phys. Chem., B* **2024**, *128*, 109–116.

(53) Kühne, T. D. et al. CP2K: An electronic structure and molecular dynamics software package - Quickstep: Efficient and accurate electronic structure calculations. *J. Chem. Phys.* **2020**, *152*, 194103.

(54) Abraham, M. J.; Murtola, T.; Schulz, R.; Páll, S.; Smith, J. C.; Hess, B.; Lindahl, E. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* **2015**, *1*, 19–25.

(55) Hjorth Larsen, A. et al. The atomic simulation environment—a Python library for working with atoms. *J. Phys. Condens. Matter* **2017**, *29*, 273002.

(56) Li, P.; Liu, X.; Chen, M.; Lin, P.; Ren, X.; Lin, L.; Yang, C.; He, L. Large-scale ab initio simulations based on systematically improvable atomic basis. *Comput. Mater. Sci.* **2016**, *112*, 503–517.

(57) Zhang, Y.; Wang, H.; Chen, W.; Zeng, J.; Zhang, L.; Han, W.; E, W. DP-GEN: A concurrent learning platform for the generation of reliable deep learning based potential energy models. *Comput. Phys. Commun.* **2020**, *253*, 107206.

(58) Martin, K.; Hoffman, B. *Mastering CMake: Version 3.1*; Kitware Incorporated.

(59) Schreiner, H., III; Fillion-Robin, J.-C.; McCormick, M. Scikit-build-core. 2024.

(60) Giese, T. J.; Zeng, J.; York, D. M. Multireference Generalization of the Weighted Thermodynamic Perturbation Method. *J. Phys. Chem. A* **2022**, *126*, 8519–8533.

(61) Bannwarth, C.; Ehlert, S.; Grimme, S. GFN2-xTB—An Accurate and Broadly Parametrized Self- Consistent Tight-Binding Quantum Chemical Method with Multipole Electrostatics and Density-Dependent Dispersion Contributions. *J. Chem. Theory Comput.* **2019**, *15*, 1652–1671.

(62) Bannwarth, C.; Caldeweyher, E.; Ehlert, S.; Hansen, A.; Pracht, P.; Seibert, J.;

Spicher, S.; Grimme, S. Extended tight-binding quantum chemistry methods. *WIREs Comput. Mol. Sci.* **2020**, *11*, e01493.

(63) Mardirossian, N.; Head-Gordon, M. ωB97M-V: A combinatorially optimized, range-separated hybrid, meta-GGA density functional with VV10 nonlocal correlation. *J. Chem. Phys.* **2016**, *144*, 214110.

(64) Eastman, P.; Behara, P. K.; Dotson, D. L.; Galvelis, R.; Herr, J. E.; Horton, J. T.; Mao, Y.; Chodera, J. D.; Pritchard, B. P.; Wang, Y.; De Fabritiis, G.; Markland, T. E. SPICE, A Dataset of Drug-like Molecules and Peptides for Training Machine Learning Potentials. *Sci. Data* **2023**, *10*, 11.

(65) 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* **2020**, *7*, 134.

(66) Devereux, C.; Smith, J. S.; Huddleston, K. K.; Barros, K.; Zubatyuk, R.; Isayev, O.; Roitberg, A. E. Extending the Applicability of the ANI Deep Learning Molecular Potential to Sulfur and Halogens. *J. Chem. Theory Comput.* **2020**, *16*, 4192–4202.

(67) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization. 2017; `https://arxiv.org/abs/1412.6980`.

Graphical TOC Entry