The Journal of Open Source Software

AmpTorch: A Python package for scalable fingerprint-based neural network training on multi-element systems with integrated uncertainty quantification

Muhammed Shuaibi[1*], Yuge Hu[2*], Xiangyun Lei[2], Benjamin M. Comer[2], Matt Adams[1], Jacob Paras[3], Rui Qi Chen[2], Eric Musa[4], Joseph Musielewicz[1], Andrew A. Peterson[5], Andrew J. Medford[2], and Zachary Ulissi[1¶]

1 Department of Chemical Engineering, Carnegie Mellon University, United States **2** Department of Chemical and Biomolecular Engineering, Georgia Institute of Technology, United States **3** School of Physics and School of Computer Science, Georgia Institute of Technology, United States **4** Department of Chemical Engineering, University of Michigan, United States **5** School of Engineering, Brown University, United States ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.05035

Software

- Review
- Repository
- Archive

Editor: David Hagan

Reviewers:

- @ml-evs
- @ianfhunter

Submitted: 22 August 2022
Published: 26 July 2023

Summary

Machine learning (ML) force fields use ML models to predict the energy (and forces) given a chemical system defined by a set of atomic coordinates. Fingerprint-based and graph-based ML force fields are two major categories of atomistic machine-learning models. Fingerprint-based models convert the atomic coordinates into fixed-length "feature vectors" that are used as inputs to regression models such as neural networks or kernel ridge regression, while graph-based systems convert the coordinates into a molecular graph which is directly input into a deep learning model. Fingerprint-based approaches tend to require less training data and be more interpretable, while graph-based models are typically able to handle more complex chemical systems and are more accurate in the limit of large datasets. Fingerprint-based models are widely used due to their conceptual similarity to traditional force-fields, but most existing packages for fingerprint-based neural network force fields are limited to systems with relatively few (< 5) elements and relatively small ($< 10^6$) datasets.

This work introduces `AmpTorch`, a Python/C++ package that leverages the Gaussian Multipole (GMP) fingerprinting scheme to build atomistic neural network models (Lei & Medford, 2022). It provides an efficient training routine scalable to $\sim 10^6$ training points, is compatible with many-element systems ($50+$ elements), and supports statistically rigorous uncertainty quantification (UQ) during the prediction step (Hu et al., 2022). The fingerprint-based structure provides relatively simple models compared to graph-based alternatives, and the structure of the fingerprinting scheme allows the package to handle systems with an arbitrary number of elements and arbitrary boundary conditions (isolated, periodic, semi-periodic).

`AmpTorch` is a PyTorch adaptation of the Atomistic Machine-learning Package `AMP` (Khorshidi & Peterson, 2016). It maintains the modular structure of `AMP`, with separate modules for generating fingerprints and training neural network models. `AmpTorch` supports standard symmetry function fingerprints and high-dimensional neural network potentials, but is able to scale to larger datasets and more complex chemical systems than `AMP` or other existing codes for feature-based ML potentials (for example, Atom-Centered Symmetry Functions (Behler, 2015) whose fingerprinting dimension scales with the number of elements). `AmpTorch` outsources traditional boilerplate trainer code to `Skorch` (Tietz et al., 2017). `Skorch` serves

as a PyTorch wrapper that allows users to easily modify traditional neural network training strategies. AmpTorch's use of Skorch allows users to trivially modify training strategies, model architectures, and hyperparameters.

The scalability of AmpTorch is a result of its fingerprinting scheme, software design, and database management. The GMP fingerprints have a fixed vector length regardless of the number of elements, and using Gaussian functions allows all necessary integrals to be computed analytically. The analytical solutions are coupled with a C++ implementation, ensuring the efficiency of the fingerprint calculation. The GMP fingerprints are also naturally compatible with the SingleNN (Liu & Kitchin, 2020) neural network structure, allowing for neural networks that have the same architecture and number of fitted parameters regardless of the number of elements present in the dataset.

In addition, AmpTorch leverages the B-tree-based database management library, LMDB (Chu & Hedenfalk, 2010), to resolve possible memory issues when it comes to loading and training on large datasets. This allows AmpTorch to be trained on datasets that are too large to fit into temporary memory, although reading and writing data from LMDB is slower than using temporary memory.

AmpTorch also implements UQ as an optional feature during the prediction. Supported UQ methods include the ensemble method, dropout neural network method, and methods based on latent distances. In particular, AmpTorch includes an implementation of a highly scalable new approach that leverages distances in the latent space and the "conformal prediction" statistical technique to provide statistically rigorous error bars on complex pre-trained models.

AmpTorch is designed using standards from the ASE package for handling atomic structures (Hjorth Larsen et al., 2017). It takes a list of ase.Atoms objects with energy (and forces) as input and output ase.Calculator object that can be used to compute the energy (and forces) with the trained model. This structure allows for easy integration with active learning packages such as FINETUNA (Musielewicz et al., 2022) and data visualization tools such as ElectroLens (Lei et al., 2019).

Statement of need

There are numerous software packages for both constructing and applying ML force-fields. These include packages that are based on deep learning and graph convolutional models, kernel-based regression models, and neural networks with fixed feature vectors. Of these packages, only graph convolutional codes have been demonstrated for extremely large and complex datasets with $>$ 10 elements and $>$ 1M training points such as the OC20 dataset in Open Catalyst Project (Chanussot et al., 2021). However, at inference time, adapting these often complex architectures with 1-100M parameter models into atomistic simulation interfaces seeking extremely fast calls is a challenging task. On the other hand, all existing packages for training feature-based models are limited in both the number of elements they can handle (due to the poor scaling of feature vector size) and the number of training points (due to scaling of training kernel-based models and memory management issues in most codes). Thus, AMPTorch aims to fill this gap by developing a toolkit to train fingerprint-based neural network force fields on datasets of an arbitrary number of chemical elements. For this reason, we expect that the code will be widely used by researchers seeking to train ML force-field models for complex systems that can be integrated with existing atomistic simulation pipelines.

Acknowledgements

The authors are grateful for funding from the U.S. Department of Energy's Basic Energy Science, Computational Chemical Sciences Program Office, under Award No. DE-SC0019441.

Behler, J. (2015). Constructing high-dimensional neural network potentials: A tutorial review. *International Journal of Quantum Chemistry*, *115*(16), 1032–1050. https://doi.org/10.1002/qua.24890

Chanussot, L., Das, A., Goyal, S., Lavril, T., Shuaibi, M., Riviere, M., Tran, K., Heras-Domingo, J., Ho, C., Hu, W., Palizhati, A., Sriram, A., Wood, B., Yoon, J., Parikh, D., Zitnick, C. L., & Ulissi, Z. (2021). Open Catalyst 2020 (OC20) Dataset and Community Challenges. *ACS Catalysis*, *11*(10), 6059–6072. https://doi.org/10.1021/acscatal.0c04525

Chu, H., & Hedenfalk, M. (2010). *Lightning Memory-Mapped Database Manager (LMDB)*.

Hjorth Larsen, A., JØrgen Mortensen, J., Blomqvist, J., Castelli, I. E., Christensen, R., Dułak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Bjerre Jensen, P., Kermode, J., Kitchin, J. R., Leonhard Kolsbjerg, E., Kubal, J., Kaasbjerg, K., Lysgaard, S., … Jacobsen, K. W. (2017). The atomic simulation environment—a Python library for working with atoms. *Journal of Physics: Condensed Matter*, *29*(27), 273002. https://doi.org/10.1088/1361-648X/AA680E

Hu, Y., Musielewicz, J., Ulissi, Z. W., & Medford, A. J. (2022). Robust and scalable uncertainty estimation with conformal prediction for machine-learned interatomic potentials. *Machine Learning: Science and Technology*, *3*(4), 045028. https://doi.org/10.1088/2632-2153/ACA7B1

Khorshidi, A., & Peterson, A. A. (2016). Amp: A modular approach to machine learning in atomistic simulations. *Computer Physics Communications*, *207*, 310–324. https://doi.org/10.1016/j.cpc.2016.05.010

Lei, X., Hohman, F., Chau, D. H. P., & Medford, A. J. (2019). ElectroLens: Understanding Atomistic Simulations Through Spatially-resolved Visualization of High-dimensional Features. *2019 IEEE Visualization Conference, VIS 2019*, 196–200. https://doi.org/10.48550/arxiv.1908.08381

Lei, X., & Medford, A. J. (2022). A Universal Framework for Featurization of Atomistic Systems. *Journal of Physical Chemistry Letters*, *13*(34), 7911–7919. https://doi.org/10.1021/acs.jpclett.2c02100

Liu, M., & Kitchin, J. R. (2020). SingleNN: Modified Behler-Parrinello Neural Network with Shared Weights for Atomistic Simulations with Transferability. *Journal of Physical Chemistry C*, *124*(32), 17811–17818. https://doi.org/10.1021/acs.jpcc.0c04225

Musielewicz, J., Wang, X., Tian, T., & Ulissi, Z. (2022). FINETUNA: fine-tuning accelerated molecular simulations. *Machine Learning: Science and Technology*, *3*(3). https://doi.org/10.1088/2632-2153/ac8fe0

Tietz, M., Fan, T. J., Nouri, D., & Bossan, B. (2017). *skorch: A scikit-learn compatible neural network library that wraps PyTorch*.