




GMP-Featurizer: A parallelized Python package for efficiently computing the Gaussian Multipole features of atomic systems

Xiangyun Lei ¹ and Joseph Montoya ¹

¹ Toyota Research Institute, Los Altos, CA, United States of America

DOI: [10.21105/joss.05476](https://doi.org/10.21105/joss.05476)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Rachel Kurchin](#)  

Reviewers:

- [@isdanni](#)
- [@bdice](#)
- [@yw-fang](#)

Submitted: 04 April 2023

Published: 10 August 2023

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

GMP-Featurizer is a lightweight, accurate, efficient, and scalable software package for calculating the Gaussian Multipole (GMP) features ([Lei & Medford, 2022](#)) for a variety of atomic systems with elements across the periodic table. Starting from the GMP feature computation module from AmpTorch ([AMPTorch, 2020](#)), the capability of GMP-Featurizer has since been greatly improved, including its accuracy and efficiency (please refer to the Overview section for details), as well as the ability to parallelize on different cores, even machines. Moreover, this Python package only has very few dependencies that are all standard Python libraries, plus CFFI for C++ code interfacing and Ray ([Moritz et al., 2018](#)) for parallelization, making it lightweight and robust. A set of unit tests are designed to ensure the reliability of its outputs. A set of extensive examples and tutorials, as well as two sets of pseudopotential files (needed for specifying the GMP feature set), are also included in this package for its users. Overall, this package is designed to serve as a standard implementation for chemical and material scientists who are interested in developing models based on GMP features. The source code for this package is freely available to the public under the Apache 2.0 license.

Statement of need

Representing the local and global environments in atomic systems in a descriptive and efficient way has been an important research topic in the chemistry, chemical engineering, and material science communities. Having good representations, or features, of chemical environments has proven to be vital for building reliable machine learning (ML) models. These models can accurately predict properties of atomic systems, and in limited cases have even been used for discovering or designing new chemicals and materials ([Collins et al., 2017](#); [Zuo et al., 2021](#)). So far, scientists and researchers have designed featurization schemes like the atom-centered symmetry function (ACSF) ([Behler & Parrinello, 2007](#)), the smooth overlap of atomic positions (SOAP) ([Bartók et al., 2017](#)), and the Gaussian Momentum ([Zaverkin & Kästner, 2020](#)) schemes. More recently, graph representation and ML models based on them (e.g. MEGNet ([Chen et al., 2019](#)), CGCNN ([Xie & Grossman, 2018](#))) have been successful. Gaussian Multipole, or GMP ([Lei & Medford, 2022](#)), is a recently developed scheme of featurizing local chemical environments, i.e. the chemical characteristics of spaces near individual atoms in molecules and crystal structures. GMP approximates underlying local electronic environments (e.g. approximated distribution of local electron cloud) using multipole expansion, the theory of which is introduced in a prior publication ([Lei & Medford, 2022](#)). The featurization scheme is flexible, depending only on prior assumptions of atomic identity and position, and is therefore applicable to various atomic systems (molecules, nanoparticles, periodic crystals, etc.) in which atomic arrangements are known. Feature computation is fast, and the representation accuracy is systematically improvable. Moreover, thanks to the deep connection between the GMP

features and physics, we have previously shown that ML models based on these features are transferable (Lei & Medford, 2022). With these characteristics, GMP featurization could be useful to a broad audience for future research in chemistry and materials science. Therefore, having lightweight, reliable, and open software that can calculate these features in a fast and accurate way is desirable.

Overview

The GMP-Featurizer package is mainly written in C++ and Python. C++ is used for the underlying computation module for speed, and Python is used for an intuitive and readable API for scientists and researchers in the community to use. Two of the most widely used Python libraries in chemistry and materials science are ASE (Larsen et al., 2017) and pymatgen (Ong et al., 2013). The GMP-featurizer library provides APIs that can read atomic systems as input from these libraries, though they are not required as dependencies. This allows the users to leverage the I/O functionalities of these two packages to get data into the format required by GMP-Featurizer. On top of that, Ray is used to parallelize the feature computation, and the parallelization efficiency is close to 100%. Overall, this package is designed to be lightweight, easy to use, fast, and accurate.

The main inputs of the workflow are a Python dictionary that contains the necessary hyperparameters for defining the desired GMP feature set, and a list of atomic systems that needs to be featurized. The native way of defining atomic systems is simply a Python dictionary that contains information like lattice vectors, atom positions, atom types, etc. As mentioned, the package also supports both ASE Atoms and pymatgen Structure objects with pre-defined converters. This capability is extensible to other formats with custom-made converters. It also supports the featurization of disordered atomic structures, which is unsupported by many popular featurization methods. Please refer to the examples for more details. The output is simply a list of dictionaries containing the resulting features, and their derivatives if requested, for the atomic structures.

By default, the package computes GMP features at each atom position, but it can also be used to compute the features at any set of reference points inside the atomic system by providing a list of the positions of interest for each atomic structure. Users can also specify the number of cores for parallel computing. Noticeably, the GMP-featurizer package, unlike common local chemical environment featurization schemes (including the original implementation of GMP in AmpTorch), eliminates the need to define a hard distance cutoff for determining the local environment to be featurized. Instead, users simply need to state the desired accuracy of the features. GMP-featurizer then automatically identifies neighboring atoms that will make meaningful contributions to each GMP feature relative to the set accuracy level, and computes the features. This change is significant, because GMP featurizes the underlying electron density of chemical systems, which can vary greatly between elements and hence influence the GMP features differently. For instance, a distant mercury atom may have a larger impact on GMP features than a closer hydrogen atom. Additionally, different GMP features respond differently to changes in atomic positions, making a universal cutoff inefficient and potentially leading to a lot of unnecessary computations (when the cutoff is set too high) or inaccurate features (when set too low). GMP-featurizer's new adaptive scheme circumvents these issues. It not only enhances efficiency and precision but also simplifies the user experience.

Moreover, computed results can be cached locally for convenient reprocessing of datasets, e.g. after augmentation or modification. Two sets of standard pseudopotential files are also provided, which are necessary to specify GMP feature sets, but may be difficult to collect from either commercial or open-source density functional theory systems. Lastly, a series of tutorials are provided in the repository to help users with getting starting and understanding the various features of the codebase.

Acknowledgements

This work was supported by the Energy and Materials Division of the Toyota Research Institute. The authors acknowledge Jens Hummelshøj for helpful discussions regarding unit-testing frameworks.

References

- AMPTorch. (2020). <https://github.com/ulissigroup/amptorch>.
- Bartók, A. P., De, S., Poelking, C., Bernstein, N., Kermode, J. R., Csányi, G., & Ceriotti, M. (2017). Machine learning unifies the modeling of materials and molecules. *Science Advances*, 3(12), e1701816–e1701816. <https://doi.org/10.1126/sciadv.1701816>
- Behler, J., & Parrinello, M. (2007). Generalized neural-network representation of high-dimensional potential-energy surfaces. *Phys. Rev. Lett.*, 98, 146401. <https://doi.org/10.1103/PhysRevLett.98.146401>
- Chen, C., Ye, W., Zuo, Y., Zheng, C., & Ong, S. P. (2019). Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9), 3564–3572. <https://doi.org/10.1021/acs.chemmater.9b01294>
- Collins, C., Dyer, M. S., Pitcher, M. J., Whitehead, G. F. S., Zanella, M., Mandal, P., Claridge, J. B., Darling, G. R., & Rosseinsky, M. J. (2017). Accelerated discovery of two crystal structure types in a complex inorganic phase field. *Nature*, 546(7657), 280–284. <https://doi.org/10.1038/nature22374>
- Larsen, A. H., Mortensen, J. J., Blomqvist, J., Castelli, I. E., Christensen, R., Duřak, M., Friis, J., Groves, M. N., Hammer, B., Hargus, C., Hermes, E. D., Jennings, P. C., Jensen, P. B., Kermode, J., Kitchin, J. R., Kolsbjerg, E. L., 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>
- Lei, X., & Medford, A. J. (2022). A universal framework for featurization of atomistic systems. *The Journal of Physical Chemistry Letters*, 13(34), 7911–7919. <https://doi.org/10.1021/acs.jpcclett.2c02100>
- Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & Stoica, I. (2018). Ray: A distributed framework for emerging AI applications. *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation*, 561–577. ISBN: 9781931971478
- Ong, S. P., Richards, W. D., Jain, A., Hautier, G., Kocher, M., Cholia, S., Gunter, D., Chevrier, V. L., Persson, K. A., & Ceder, G. (2013). Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68, 314–319. <https://doi.org/10.1016/j.commatsci.2012.10.028>
- Xie, T., & Grossman, J. C. (2018). Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14), 145301–145301. <https://doi.org/10.1103/PhysRevLett.120.145301>
- Zaverkin, V., & Kästner, J. (2020). Gaussian moments as physically inspired molecular descriptors for accurate and scalable machine learning potentials. *Journal of Chemical Theory and Computation*, 16(8), 5410–5421. <https://doi.org/10.1021/acs.jctc.0c00347>
- Zuo, Y., Qin, M., Chen, C., Ye, W., Li, X., Luo, J., & Ong, S. P. (2021). Accelerating materials discovery with Bayesian optimization and graph deep learning. *Materials Today*, 51, 126–135. <https://doi.org/10.1016/j.mattod.2021.08.012>