graph-pes: graph-based machine-learning models for potential-energy surfaces

John L. A. Gardner[1] and **Volker L. Deringer**[1]

1 Inorganic Chemistry Laboratory, Department of Chemistry, University of Oxford, Oxford OX1 3QR, United Kingdom

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 22 June 2025
Published: unpublished

Summary

We present `graph-pes`, an open-source toolkit for accelerating the development, training, and deployment of machine-learned interatomic potential (MLIP) models that act on graph representations of atomic structures. The `graph-pes` toolkit comprises three components:

1. **The graph_pes Python package**: a modular framework containing all functionality required to build, train, and evaluate graph-based MLIPs. The package includes a mature data pipeline for converting atomic structures into graph representations (`AtomicGraphs`), a fully featured base class for MLIP implementations (`GraphPESModel`), and a suite of common data manipulation routines and model building blocks. We provide independent (re-) implementations of common MLIP architectures out-of-the-box, as well as interfaces to several foundational MLIP models (see below).
2. **The `graph-pes-train` command-line interface** (CLI): a convenience tool for training graph-based MLIPs on datasets of labelled atomic structures directly from the command line. The tool is compatible with any `GraphPESModel` (i.e., those defined in `graph-pes`, user-designed ones, and foundation models) and is designed to be easily extensible via custom loss functions, optimisers, datasets, and more.
3. **Molecular-dynamics drivers** for popular MD engines that allow any `GraphPESModel` to be used in GPU-accelerated MD simulations. We currently provide a `pair style` for use in LAMMPS (Thompson et al., 2022), a `GraphPESCalculator` for use in ASE (Larsen et al., 2017), and an integration with the `torch-sim` package (Gangan et al., 2025).

Statement of need

In recent years, machine-learned PES models, commonly referred to as machine-learned interatomic potentials (MLIPs), have become central tools for computational chemistry and materials science (Deringer et al., 2019).

These models are trained on labels generated by quantum-mechanical methods, but scale much more favourably with system size, making it possible to simulate the dynamics of large systems (millions of atoms and more) over extended timescales. In this way, MLIPs are facilitating the study of complex physical and chemical phenomena at the atomic scale, in turn driving the generation of novel insight and understanding.

Many flavours of MLIPs exist, and with them have arisen a variety of software packages that are typically tailored to training specific architectures (see examples below). Given their unique specialisations, these individual software implementations do not normally conform to a common interface, making it difficult for practitioners to migrate their training and validation pipelines between different model architectures.

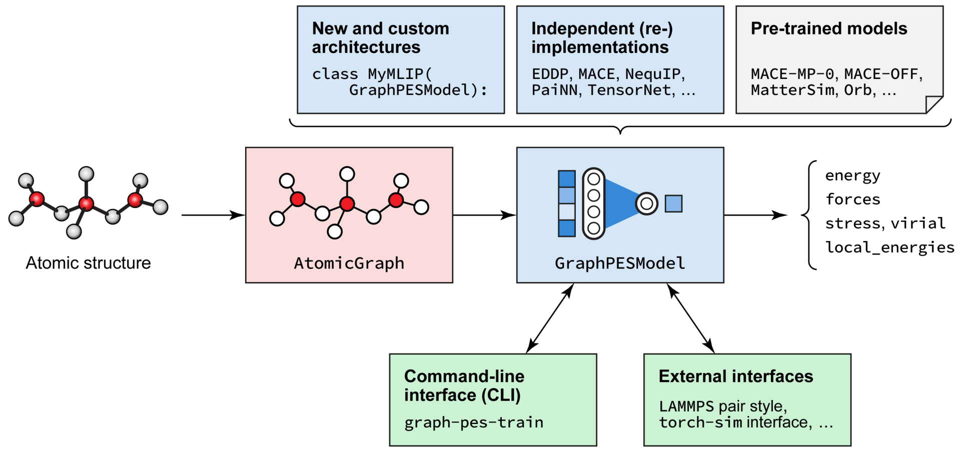
New and custom architectures
class MyMLIP(
GraphPESModel):
Independent (re-) implementations
EDDP, MACE, NequIP, PaiNN, TensorNet, ...
Pre-trained models
MACE-MP-0, MACE-OFF, MatterSim, Orb, ...
Atomic structure
AtomicGraph
GraphPESModel
energy
forces
stress, virial
local_energies
Command-line interface (CLI)
graph-pes-train
External interfaces
LAMMPS pair style, torch-sim interface, ...

Figure 1: Schematic overview of the functionality of `graph-pes`. The core components are highlighted in colour. Red: The `AtomicGraph` class is used to represent atomic structures and incorporates the notion of locality via a neighbour list. Blue: The `GraphPESModel` class is the general base class for all models defined using `graph-pes`. We provide independent, stand-alone (re-) implementations of popular architectures, together with interfaces allowing access to several pre-trained and "foundational" MLIP models. Custom, user-defined MLIPs are easy to create, and are fully compatible with the rest of `graph-pes`'s functionality. Green: `graph-pes` includes a CLI for easy training, and interfaces to multiple external simulation tools for evaluating MLIP models.

`graph-pes` provides a **unified interface and software framework** for defining, training, and working with graph-based MLIP models. This reduces the barrier to entry for researchers wanting to implement new MLIP architectures, and allows practitioners to easily explore different MLIP architectures for their specific use cases: training scripts require as little as one line of code to swap between model architectures, while validation scripts can be written in an architecture-agnostic manner, with `LAMMPS` input scripts, `ASE` calculators, and `torch-sim` simulations requiring no changes other than pointing to a different model file.

Features and implementation

Below, we briefly summarise the key design choices, components, and features of `graph-pes`, and further emphasise the advantages of having a unified interface for all MLIP architectures. For an extended overview of the `graph-pes` framework, and comprehensive documentation, please visit this URL.

Representing atomic structures with graphs

An atomic structure containing N atoms is completely defined by the positions of its atoms ($\mathbf{R} \in \mathbb{R}^{N\times 3}$) and their chemical identities ($Z \in \mathbb{Z}_+^N$).[1] A graph representation of the atomic structure incorporates this complete description, together with an edge list ($\mathbf{E} \in \mathbb{Z}^{E\times 2}$) indicating which atoms are within the local environment of others (defined, for instance, using a fixed cut-off radius). The resulting graph, $G = \{\mathbf{R}, Z, \mathbf{E}\}$, is thus an extremely general representation of chemical structure with a built-in definition of locality.

We therefore define the `AtomicGraph` class as the base data structure in `graph-pes`, and provide convenience methods to convert these to and from `ase.Atoms` objects (Hjorth Larsen et al., 2017).

[1]This statement is assuming that the structure is isolated: defining a periodic structure requires the trivial addition of a unit cell and periodic boundary conditions.

Writing performant code to implement common graph-based operations can be challenging: we therefore provide optimised implementations to access many derived properties (such as `number_of_atoms`, `neighbour_distances`, and `number_of_neighbours`) as well as to perform common graph-based operations (such as `index_over_neighbours`, `sum_over_neighbours`, and `sum_per_structure`). All of these functions work for both single and batched graph instances, simplifying the implementation of new MLIP models, and making their forwards passes easily readable.

Model implementations

All MLIP models in `graph-pes` are implemented as subclasses of the `GraphPESModel` base class, which itself inherits from the `torch.nn.Module` class. These models take an (optionally batched) `AtomicGraph` as input, and are able to return a collection of PES property predictions, including the total energy, atomic forces, and cell stress tensors.

Implementations need only define a forward pass that returns a local energy for each atom in the graph, or a total energy for the entire structure; we use the functionality from `torch.autograd` to automatically calculate force, and stress tensors in a conservative manner (Paszke et al., 2019). For faster modelling, we also fully support models that return direct force and stress tensor predictions (e.g., `TensorNet` or `orb-v3-*` with their optional direct force readout heads).

Building on the `GraphPESModel` class, we provide independent (re-) implementations of popular MLIP architectures, including `PaiNN` (Schütt et al., 2021), `EDDP` (Pickard, 2022), `NequIP` (Batzner et al., 2022), `MACE` (Batatia et al., 2022), and `TensorNet` (Simeon & de Fabritiis, 2023). We use building blocks provided by the `e3nn` (Geiger & Smidt, 2022) package to implement models that act on spherical tensor decompositions.

Furthermore, we provide an `AdditionModel` implementation, which makes energy, force, and stress predictions as a sum over several independent models. This allows `graph-pes` to add the following features onto any other model architecture:

- **Offset energies**. A common feature of quantum-mechanical labelling methods is that the “reference energy” of an isolated atom is (i) non-zero, (ii) different for each element, and (iii) varying between different levels of theory/method. We provide the `EnergyOffset` model to account for this, removing the need to include these contributions in other model implementations within `graph-pes`.
- **Pair repulsions.** Adding smooth, short-ranged, repulsive pair repulsion contributions on top of many-bodied model predictions guarantees correct model behaviour in the short-range limit, and can act to stabilise MD simulations. We provide the `LennardJones`, `Morse`, and `ZBLCoreRepulsion`, all with smoothed cutoffs and optionally learnable parameters, to trivially add these repulsions to any other model implementation.

Training and validation

We provide the `graph-pes-train` CLI tool for training any `GraphPESModel` on datasets of labelled atomic structures. Configuration for this tool is specified via a hierarchically-structured YAML file, with separate sections for specifying the `model`, `data`, `loss`, and `fitting` parameters. As well as training from scratch, we also support the loading of pre-trained models, allowing for fine-tuning of existing models on new datasets. In this manner, a wide variety of pre-train/fine-tune strategies are supported, including synthetic pre-training (Gardner et al., 2024), foundation model fine-tuning (see below), and frozen transfer learning (Radova et al., 2025).

Under the hood, `graph-pes-train` builds upon the `PyTorch Lightning` (Falcon & The PyTorch Lightning team, 2019) training loop, allowing for a wide range of advanced features, including learning rate scheduling, stochastic weight averaging, gradient clipping, and more. By using the `data2objects` package (Gardner, 2024) to parse configuration files, we also support the use of arbitrary, user-defined components, including custom loss functions, model architectures, optimisers, and datasets.

Because all models conform to the same interface, all training features can be used with any model architecture. Similarly, all downstream model uses can be written in an architecture-agnostic manner, allowing for MD, relaxations, and other scripts to be written once, and then used with any MLIP architecture, for example in defining extended validation beyond simple error metrics (Morrow et al., 2023).

Easy access to foundation models

A topical and recent area of research is the development of universal or "foundational" MLIPs that can describe the potential-energy surface of a wide range of systems. `graph-pes` integrates directly with the `mace-torch`, `mattersim`, and `orb-models` packages to provide access to, among others, the `MACE-MP` (Batatia et al., 2024), `MatterSim` (Yang et al., 2024), `orb-v2` (Neumann et al., 2024), `MACE-OFF` (Kovács et al., 2025), `Egret-v1` (Mann et al., 2025), and `orb-v3` (Rhodes et al., 2025) families of models. Each of these integrations generates `GraphPESModels` that are directly compatible with all `graph-pes` features, including fine-tuning, validation pipelines, and MD simulations.

Related work

`graph-pes` is beginning to drive a substantial number of projects within our research group, and we hope that it will be useful to many others. In recent preprints, we have described the use of `graph-pes` for fitting NequIP models to datasets created using the `autoplex` software (Liu et al., 2024), for assessing the zero-shot performance of different graph-network MLIP models (Mahmoud et al., 2025), and for fine-tuning and distilling atomistic foundation models (Gardner et al., 2025).

Relevant alternative packages that offer training and validation functionaltiy for *specific* ML-PES architectures include: `schnetpack` (Schütt et al., 2019, 2023), `deepmd-kit` (Wang et al., 2018; Zeng et al., 2023), `nequip` (Batzner et al., 2022), `mace-torch` (Batatia et al., 2022), `torchmd-net` (Pelaez et al., 2024), and `fairchem` (Shuaibi et al., 2025). The `MatterTune` package (Kong et al., 2025) provides a unified interface for fine-tuning atomistic foundation models, but does not create models with a common interface, or allow for training arbitrary MLIP architectures from scratch.

Acknowledgements

We thank Zoé Faure Beaulieu, Krystian Gierczak, and Daniel Thomas du Toit for early testing and feedback. J.L.A.G. acknowledges a UKRI Linacre - The EPA Cephalosporin Scholarship, support from an EPSRC DTP award [grant number EP/T517811/1], and from the Department of Chemistry, University of Oxford. This work was supported by UK Research and Innovation [grant number EP/X016188/1].

References

Batatia, I., Benner, P., Chiang, Y., Elena, A. M., Kovács, D. P., Riebesell, J., Advincula, X. R., Asta, M., Avaylon, M., Baldwin, W. J., Berger, F., Bernstein, N., Bhowmik, A., Blau, S. M., Cărare, V., Darby, J. P., De, S., Della Pia, F., Deringer, V. L., … Csányi, G. (2024). *A foundation model for atomistic materials chemistry* (No. arXiv:2401.00096). https://doi.org/10.48550/arXiv.2401.00096

Batatia, I., Kovacs, D. P., Simm, G. N. C., Ortner, C., & Csanyi, G. (2022, October). MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields. *Advances in Neural Information Processing Systems.*

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., & Kozinsky, B. (2022). E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, *13*(1), 2453. https://doi.org/10.1038/s41467-022-29939-5

Deringer, V. L., Caro, M. A., & Csányi, G. (2019). Machine Learning Interatomic Potentials as Emerging Tools for Materials Science. *Advanced Materials*, *31*(46), 1902765. https://doi.org/10.1002/adma.201902765

Falcon, W., & The PyTorch Lightning team. (2019). *PyTorch Lightning* (Version 1.4). https://doi.org/10.5281/zenodo.3828935

Gangan, A. S., Cohen, O. A., Riebesell, J., Goodall, R., Kolluru, A., & Falletta, S. (2025). *Torch-Sim* (Version 0.1.0). https://doi.org/10.5281/zenodo.7486816

Gardner, J. L. A. (2024). data2objects: Self-documenting config files for use with python. In *GitHub repository*. GitHub. https://github.com/jla-gardner/data2objects

Gardner, J. L. A., Baker, K. T., & Deringer, V. L. (2024). Synthetic pre-training for neural-network interatomic potentials. *Machine Learning: Science and Technology*, *5*(1), 015003. https://doi.org/10.1088/2632-2153/ad1626

Gardner, J. L. A., Toit, D. F. T. du, Mahmoud, C. B., Beaulieu, Z. F., Juraskova, V., Paşca, L.-B., Rosset, L. A. M., Duarte, F., Martelli, F., Pickard, C. J., & Deringer, V. L. (2025). *Distillation of atomistic foundation models across architectures and chemical domains* (No. arXiv:2506.10956). https://doi.org/10.48550/arXiv.2506.10956

Geiger, M., & Smidt, T. (2022). *E3nn: Euclidean Neural Networks* (No. arXiv:2207.09453). https://doi.org/10.48550/arXiv.2207.09453

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

Kong, L., Shoghi, N., Hu, G., Li, P., & Fung, V. (2025). *MatterTune: An Integrated, User-Friendly Platform for Fine-Tuning Atomistic Foundation Models to Accelerate Materials Simulation and Discovery* (No. arXiv:2504.10655). https://doi.org/10.48550/arXiv.2504.10655

Kovács, D. P., Moore, J. H., Browning, N. J., Batatia, I., Horton, J. T., Pu, Y., Kapil, V., Witt, W. C., Magdău, I.-B., Cole, D. J., & Csányi, G. (2025). *MACE-OFF: Transferable Short Range Machine Learning Force Fields for Organic Molecules* (No. arXiv:2312.15211). https://doi.org/10.48550/arXiv.2312.15211

Larsen, A. H., Mortensen, J. 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., 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

Liu, Y., Morrow, J. D., Ertural, C., Fragapane, N. L., Gardner, J. L. A., Naik, A. A., Zhou, Y., George, J., & Deringer, V. L. (2024). *An automated framework for exploring and learning potential-energy surfaces* (No. arXiv:2412.16736). https://doi.org/10.48550/arXiv.2412.

16736

Mahmoud, C. B., El-Machachi, Z., Gierczak, K. A., Gardner, J. L. A., & Deringer, V. L. (2025). *Assessing zero-shot generalisation behaviour in graph-neural-network interatomic potentials* (No. arXiv:2502.21317). https://doi.org/10.48550/arXiv.2502.21317

Mann, E. L., Wagen, C. C., Vandezande, J. E., Wagen, A. M., & Schneider, S. C. (2025). *Egret-1: Pretrained Neural Network Potentials For Efficient and Accurate Bioorganic Simulation* (No. arXiv:2504.20955). https://doi.org/10.48550/arXiv.2504.20955

Morrow, J. D., Gardner, J. L. A., & Deringer, V. L. (2023). How to validate machine-learned interatomic potentials. *The Journal of Chemical Physics*, *158*(12), 121501. https://doi.org/10.1063/5.0139611

Neumann, M., Gin, J., Rhodes, B., Bennett, S., Li, Z., Choubisa, H., Hussey, A., & Godwin, J. (2024). *Orb: A Fast, Scalable Neural Network Potential* (No. arXiv:2410.22570). https://doi.org/10.48550/arXiv.2410.22570

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, *32*.

Pelaez, R. P., Simeon, G., Galvelis, R., Mirarchi, A., Eastman, P., Doerr, S., Thölke, P., Markland, T. E., & Fabritiis, G. D. (2024). *TorchMD-net 2.0: Fast neural network potentials for molecular simulations*. https://arxiv.org/abs/2402.17660

Pickard, C. J. (2022). Ephemeral data derived potentials for random structure search. *Physical Review B*, *106*(1), 014102. https://doi.org/10.1103/PhysRevB.106.014102

Radova, M., Stark, W. G., Allen, C. S., Maurer, R. J., & Bartók, A. P. (2025). *Fine-tuning foundation models of materials interatomic potentials with frozen transfer learning* (No. arXiv:2502.15582). https://doi.org/10.48550/arXiv.2502.15582

Rhodes, B., Vandenhaute, S., Šimkus, V., Gin, J., Godwin, J., Duignan, T., & Neumann, M. (2025). *Orb-v3: Atomistic simulation at scale* (No. arXiv:2504.06231). https://doi.org/10.48550/arXiv.2504.06231

Schütt, K. T., Hessmann, S. S. P., Gebauer, N. W. A., Lederer, J., & Gastegger, M. (2023). SchNetPack 2.0: A neural network toolbox for atomistic machine learning. *The Journal of Chemical Physics*, *158*(14), 144801. https://doi.org/10.1063/5.0138367

Schütt, K. T., Kessel, P., Gastegger, M., Nicoli, K. A., Tkatchenko, A., & Müller, K.-R. (2019). SchNetPack: A Deep Learning Toolbox For Atomistic Systems. *Journal of Chemical Theory and Computation*, *15*(1), 448–455. https://doi.org/10.1021/acs.jctc.8b00908

Schütt, K. T., Unke, O. T., & Gastegger, M. (2021). *Equivariant message passing for the prediction of tensorial properties and molecular spectra* (No. arXiv:2102.03150). https://doi.org/10.48550/arXiv.2102.03150

Shuaibi, M., Das, A., Sriram, A., Misko, Barroso-Luque, L., Gao, R., Goyal, S., Ulissi, Z., Wood, B., Xie, T., Yoon, J., Wander, B., Kolluru, A., Barnes, R., Sunshine, E., Tran, K., Xiang, Levine, D., Shoghi, N., … Michel, K. (2025). *FAIRChem* (Version 2.2.0). https://doi.org/10.5281/zenodo.15587498

Simeon, G., & de Fabritiis, G. (2023). *TensorNet: Cartesian Tensor Representations for Efficient Learning of Molecular Potentials* (No. arXiv:2306.06482). https://arxiv.org/abs/2306.06482

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. (2022). LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, *271*, 108171. https://doi.org/10.1016/j.cpc.2021.108171

Wang, H., Zhang, L., Han, J., & E, W. (2018). DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics. *Computer Physics Communications*, *228*, 178–184. https://doi.org/10.1016/j.cpc.2018.03.016

Yang, H., Hu, C., Zhou, Y., Liu, X., Shi, Y., Li, J., Li, G., Chen, Z., Chen, S., Zeni, C., Horton, M., Pinsler, R., Fowler, A., Zügner, D., Xie, T., Smith, J., Sun, L., Wang, Q., Kong, L., … Lu, Z. (2024). *MatterSim: A Deep Learning Atomistic Model Across Elements, Temperatures and Pressures* (No. arXiv:2405.04967). https://doi.org/10.48550/arXiv.2405.04967

Zeng, J., Zhang, D., Lu, D., Mo, P., Li, Z., Chen, Y., Rynik, M., Huang, L., Li, Z., Shi, S., Wang, Y., Ye, H., Tuo, P., Yang, J., Ding, Y., Li, Y., Tisi, D., Zeng, Q., Bao, H., … Wang, H. (2023). DeePMD-kit v2: A software package for deep potential models. *The Journal of Chemical Physics*, *159*(5), 054801. https://doi.org/10.1063/5.0155600