

Computer Programs in Physics



Tadah! a Swiss army knife for developing and deployment of machine learning interatomic potentials

Marcin Kirsz^{a, *}, Ayobami Daramola^{a, *}, Andreas Hermann^{a, *}, Hongxiang Zong^{b, *},
Graeme J. Ackland^{a, *, *}

^a Centre for Science at Extreme Conditions and SUPA, School of Physics and Astronomy, University of Edinburgh, Edinburgh EH9 3FD, United Kingdom

^b State Key Laboratory for Mechanical Behavior of Materials, Xi'an Jiaotong University, Xi'an 710049, China

ARTICLE INFO

The review of this paper was arranged by Prof. Andrew Hazel

Dataset link: <https://git.ecdf.ed.ac.uk/tadah>

Keywords:

Machine learning interatomic potentials
Molecular dynamics
Scientific machine learning
Physics aware machine learning
Computational modelling

ABSTRACT

The *Tadah!* code provides a versatile platform for developing and optimizing Machine Learning Interatomic Potentials (MLIPs). By integrating composite descriptors, it allows for a nuanced representation of system interactions, customized with unique cutoff functions and interaction distances. *Tadah!* supports Bayesian Linear Regression (BLR) and Kernel Ridge Regression (KRR) to enhance model accuracy and uncertainty management. A key feature is its hyperparameter optimization cycle, iteratively refining model architecture to improve transferability. This approach incorporates performance constraints, aligning predictions with experimental and theoretical data. *Tadah!* provides an interface for LAMMPS, enabling the deployment of MLIPs in molecular dynamics simulations. It is designed for broad accessibility, supporting parallel computations on desktop and HPC systems. *Tadah!* leverages a modular C++ codebase, utilizing both compile-time and runtime polymorphism for flexibility and efficiency. Neural network support and predefined bonding schemes are potential future developments, and *Tadah!* remains open to community-driven feature expansion. Comprehensive documentation and command-line tools further streamline the development and application of MLIPs.

Program summary

Program title: Tadah!

CPC Library link to program files: <https://doi.org/10.17632/vy6y3tjdr3.1>

Developer's repository link: <https://git.ecdf.ed.ac.uk/tadah>

Licensing provisions: GPLv3

Programming language: C++

Supplementary material: Installation instructions and usage examples are available in the *Tadah!* online documentation at <https://tadah.readthedocs.io>

Nature of problem: Atomistic modelling, particularly molecular dynamics, is among the most popular techniques used in physics and chemistry research. Accurate and efficient methods are required to generate forces for such simulations. Quantum mechanical calculation of the electronic structure is the “gold standard” here, but is restricted to relatively small systems. Hence, interatomic potentials have a role in allowing large scale simulations, provided they have adequate accuracy.

Over the past two decades, the paradigm has shifted from developing interatomic potentials using physics-informed functional forms to generating machine learning interatomic potentials (MLIPs) with more flexible mathematical forms, albeit lacking clear interpretability.

MLIPs typically lack physical insights in trained models, requiring comprehensive datasets from methods like density functional theory during model parametrization. While training on energies and derivatives may ensure good interpolation within the training data, it frequently lacks transferability, failing environments unlike those in the training data. At the same time the ability to fit a model to experimental data increasingly seems to be lost. In essence, the MLIP framework includes a feature vector (descriptor/fingerprint) and a regression method (e.g., kernel ridge regression or neural networks). Currently, there are numerous frameworks, but there is no consensus on a superior approach. Competing factors such as accuracy, transferability, efficiency, and dataset requirements

* Corresponding author.

E-mail address: gjackland@ed.ac.uk (G.J. Ackland).

influence user decisions. Because of the fragmentation of the methodologies, users frequently need to compile multiple software tools, which then often prove to be not end-user-ready.

A niche exists for a more flexible framework within which users can mix and match descriptor and regression methods. This would allow MLIP developers to experiment, test, and rapidly deploy models to production, using the methods best suited to their system, application and target outputs.

It is desirable to provide additional tools for fitting and verification of models. Furthermore, deployment in molecular dynamics (MD) settings, requires effort to integrate the MLIP with third-party MD software. Plugins for standard MD codes should be available alongside MLIP fitting codes.

Solution method: *Tadah!* is a unified program designed for the development of advanced machine learning interatomic potentials using various descriptors, cutoffs, and regression methods. It also allows rapid deployment of these through the MD LAMMPS plugin [1]. It supports mono- and multi-species systems as well as advanced (custom) fitting procedures. The software provides unique pathways to embed prior domain knowledge into a model and offers a range of additional tools for dataset conversion, manipulation, and plotting.

Tadah! allows users to incorporate their physical insight by combining multiple descriptors into a single composite descriptor. Each constituent descriptor can be further customized by specifying a unique cutoff function and distance. Additionally, users can define specific chemical species pairs that each constituent descriptor will target, enabling tailored interactions within the model.

Tadah! also provides users with a nested fitting procedure that allows the optimization of model hyperparameters with a custom-defined loss function. In principle, the loss function can incorporate any quantity obtainable from MD simulation and be compared against experimental values for scoring.

The *Tadah!* toolkit provides a command-line interface (CLI) binary as well as a C++ API for advanced use. The majority of features are available via the CLI, while the API allows developers and users to rapidly implement and test new descriptors and various regression strategies. It supports seamless integration of new types of descriptors, with an API that enables their addition. We follow the philosophy: “Implement once, use it everywhere across *Tadah!* and LAMMPS.” The code is open source, and we aim to drive development based on user feedback. We encourage users to request new features and provide feedback. We are also open to individual contributions and collaborations.

Additional comments including restrictions and unusual features: *Tadah!* provides two end-user packages: *Tadah!MLIP* for the development of MLIPs and *Tadah!LAMMPS* for deployment in molecular dynamics settings. The *Tadah!MLIP* software can be built as a desktop version using OpenMP. For users working with large datasets or complex descriptors, a massively parallel version of *Tadah!MLIP* is available with MPI, designed for HPC architectures.

References

- [1] A.P. Thompson et al., LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, *Comput. Phys. Comm.* 271 (2022) 10817.

1. Introduction

Atomic-level simulations rely heavily on the choice of interatomic potentials. While empirical potentials derived from experimental data are computationally efficient, they often lack accuracy. In contrast, ab initio potentials, grounded in quantum mechanics, provide robust and transferable predictions, albeit at a greater computational cost. The emergence of machine learning interatomic potentials (MLIPs) allows for the design of models that combine computational efficiency with the capacity for high accuracy [1].

Numerous software packages have been developed for training MLIPs [2–11]. Python-based tools support diverse descriptors but are inherently slower. Conversely, Fortran and C/C++ packages offer speed but are often restricted to a single descriptor type or regression method. Although Python codes can be sped up using external routines, the escalating complexity of fitting routines eventually limits these optimizations. The current landscape is dominated by packages employing neural networks as their preferred regression method.

Constructing an MLIP begins with prior knowledge. This involves selecting the functional form for the feature vector (descriptor), regression methods, training database, and machine learning (ML) hyperparameters (HP). These decisions must be made before initiating regression. These choices control the model’s complexity and its capacity for accuracy, computational efficiency, and transferability between different atomic environments. There is typically a manual and iterative process where the model is trained on the training data and fine-tuned with the validation set [12]. Using different pieces of software is suboptimal from the user’s perspective, as it significantly limits the ability to experiment. Moreover, optimizing hyperparameters is crucial during model

development. Surprisingly, to the best of our knowledge, this topic remains inadequately addressed within the MLIP community [13–15].

In this work, we present novel, semi-automated training methods designed to facilitate the development of general-purpose MLIP models [16,17]. These methods are implemented in the *Tadah!* software. The software is fully interfaced with LAMMPS [18], allowing for efficient large-scale molecular dynamics simulations.

2. Methods and discussion

The goal of interatomic potentials is to represent the potential energy surface (PES) for a given set of atomic coordinates. MLIPs extend the classical approach by decomposing the system’s total potential energy, U_{tot} into local atomic energies U_i for each atom. As illustrated in Fig. 1, local atomic environments are defined within a user-defined cutoff distance r_c .

$$U_{tot} = \sum_i^N U_i = \sum_i^N \mathcal{M}(\mathbf{d}_i) \quad (1)$$

In eq. (1), a descriptor vector, \mathbf{d}_i , captures information about the local atomic environment of the i -th atom within a predefined cutoff radius r_c . When integrated into a trained ML model, \mathcal{M} , it predicts the local energy U_i . The summation is over all atoms in the simulation. The forces can be obtained by taking the derivatives of U_{tot} with respect to the atomic coordinates.

The *Tadah!* software supports multiple types of descriptors and regression methods, as well as the ability to construct composite descriptors. This section provides a brief overview of the theory behind MLIPs. The features unique to *Tadah!* are discussed in more detail. In particular, we present the generalized form of the descriptor vectors currently

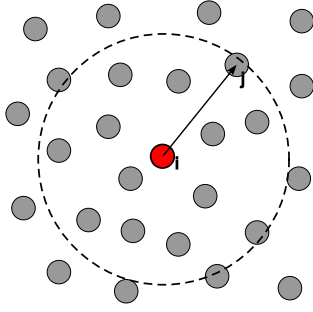


Fig. 1. Local atomic environment of atom i . The atom j is considered as within the local environment of atom i if the separation between atoms is within user defined cutoff distance r_c .

supported by *Tadah!* and the semi-automated nested fitting procedure designed to reduce manual effort during model parametrization and improve transferability.

2.1. Parameters

Both \mathcal{M} and (\mathbf{d}_i) are functions.¹ They contain two types of parameters, *Learned parameters* (LPs) which are optimised in the training process (e.g. regression coefficients) and *hyperparameters* (HPs) which are not (e.g. cutoffs, and even the length of the descriptor vector). *Tadah!* offers an external loop which can optimise the hyperparameters either to improve the fitting, or to optimise some emergent property such as equation of state.

2.2. Generalised descriptor

The descriptor vector (\mathbf{d}) captures information about the local atomic environment of an atom. To construct \mathbf{d} , we first define a set of hyperparameters that specify its functional form. Note that vector component $\mathbf{d}^{(p)}$ of \mathbf{d} may have the same functional form with a different set of HPs.

For example, one can construct a simple Lennard-Jones type descriptor with two vector components, corresponding to attraction and repulsion respectively, and consider the exponents of particle separation r as the following sets of hyperparameters: $\{-6\}_0$ and $\{-12\}_1$. Additional examples may include descriptors that use Gaussian functions, where hyperparameters might define the width and position of the Gaussians. Chemical species-dependent weights can also be used to tailor the descriptor for specific interactions, enhancing its sensitivity to the chemical composition of the environment.

The *Tadah!* software supports two- and many-body types of descriptors. The specific angular type descriptor is omitted, as three-body interactions can generally be incorporated into a many-body format. This approach avoids the cost associated with the computation of angular descriptors [19]. The list of currently supported types of descriptors is available in online documentation [20].

2.2.1. Two-body components in the descriptor

The simplest form of component in the descriptor is a two-body functional form. It is motivated by the idea that the energy depends on pairwise interactions between atoms. A two-body p -th component of the descriptor of the i -th atom is

$$\mathbf{d}_i^{(p)} = \sum_{j \neq i} B^{\{\zeta\}_p}(r_{ij}) f_c^{\{\zeta\}_p}(r_{ij}) \quad (2)$$

where B is a descriptor specific function with a set of hyperparameters $\{\zeta\}_p$, f_c is a cutoff function which ensures that energy goes smoothly to

zero at the cutoff distance, and r_{ij} is a separation between i and j atoms. The summation is over all neighbours of central atom i which are within r_c . Note that r_c is included in the $\{\zeta\}_p$ subset of hyper parameters as it might be required by the B functions as well.

2.2.2. Many-body components of the descriptor

The many-body descriptors of the i -th atom are motivated by the idea that the energy depends on the local density. They satisfy the following form

$$\mathbf{d}_i^{(p)} = \mathcal{D}^{\{\zeta\}_p}(\rho_i^{\{\zeta\}_p}) \quad (3)$$

where the vector of atomic density ρ_i of the i -th atom is built by expanding the density using the basis set of choice.

$$\rho_i^{\{\zeta\}_p} = \left(\sum_j \psi_1^{\{\zeta\}_p}(\mathbf{r}_{ij}), \dots, \sum_j \psi_{\max}^{\{\zeta\}_p}(\mathbf{r}_{ij}) \right) \quad (4)$$

For linear regression, the functional \mathcal{D} in eq. (3) must ensure invariance under permutations of atoms of the same species, as well as inversion, translation, and rotation of the system. In principle, these invariances can also be incorporated into a custom non-linear model.

2.2.3. Composite descriptors

Tadah! offers the flexibility to incorporate physical insights into the modelling process by choosing the physical functional forms of the descriptor components. These multiple components are combined into a single, unified custom descriptor vector, referred to as a composite descriptor (Fig. 2). This approach allows for a more nuanced representation of the system's interactions. Each constituent descriptor within this composite can be further tailored by specifying a unique cutoff function and an interaction distance, enhancing the precision and relevance of the model. Furthermore, it is possible to define particular chemical species pairs that each constituent descriptor will target. This capability enables the creation of highly specific interactions, ensuring the model accurately reflects the complex dynamics of the system being studied. By leveraging these features, models can be developed that are not only highly customized but also deeply informed by domain-specific knowledge.

2.3. Regression

Tadah! uses the regularized form of the normal equation during the regression stage to find the optimal parameters that minimize the error between predicted and actual data:

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t} \quad (5)$$

where \mathbf{w} is the optimized weight vector, \mathbf{X} is the design matrix which is constructed using descriptor vectors, \mathbf{t} is the target vector consisting of energies, forces, and stresses, λ is the regularization parameter, and \mathbf{I} is the identity matrix. This closed-form solution is equivalent to minimizing the loss function, defined as the sum of squared errors, with the advantage of being exact.

An evidence approximation algorithm is used to estimate λ , which helps prevent model overfitting by shrinking the weights. Users also have the option to select this term manually. Models regularized with this approach may exhibit higher bias on training data but achieve lower variance and better accuracy on test data.

Tadah! currently supports two regression methods: Bayesian Linear Regression (BLR) and Kernel Ridge Regression (KRR) [21]. The key difference between KRR and BLR lies in the computation of the design matrix \mathbf{X} . Once constructed, both methods apply regularized linear regression (eq. (5)). For KRR, the design matrix \mathbf{X} is replaced by the sparse kernel matrix \mathbf{K} , constructed using a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$. Here, the index i goes over all descriptors in the training dataset, while index j goes over all preselected basis vectors. For BLR, each row of \mathbf{X} is post-processed with the basis function of choice, resulting in the Φ matrix.

¹ Strictly, \mathcal{M} is a functional and (\mathbf{d}_i) is a vector of functions.

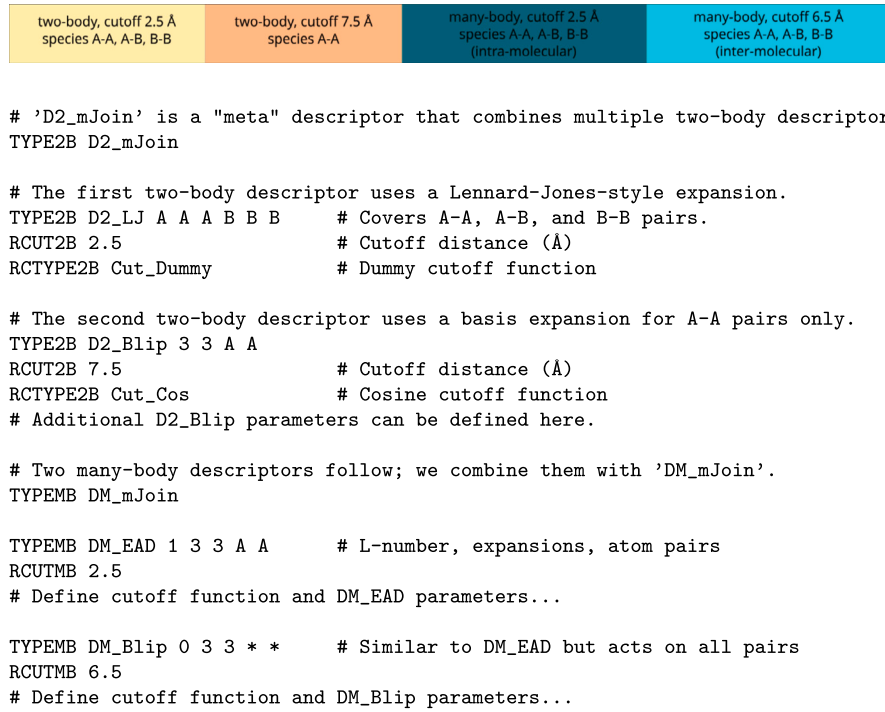


Fig. 2. A schematic representation of the composite descriptor (top) and a sample configuration snippet (bottom) illustrating how it is defined in *Tadah!*. Parameter values are illustrative and not physically meaningful. A and B here are placeholders for actual chemical species. Additional descriptors can be appended as needed.

For both algorithms, the design matrix can include descriptor vectors for structural potential energies and optionally for forces and stresses. Various kernel types and basis functions are supported. For details, refer to the online documentation [20].

In BLR, regression is framed as a probabilistic model, allowing for robust predictions that incorporate uncertainty. BLR assumes a prior distribution over the weight vector \mathbf{w} . The model updates this prior to a posterior distribution based on the observed data. This approach not only predicts outputs but also provides a measure of uncertainty for predictions, which is valuable for assessing model confidence.

Our KRR implementation leverages the Empirical Kernel Map (EKM) [22], enabling the processing of large datasets and producing sparse outputs. EKM kernelizes algorithms that use vectors by projecting new vectors into a feature space defined by basis vectors and a kernel function. This allows standard (not kernelized) algorithms to operate without modifications. *Tadah!* provides several tools for selecting a suitable set of basis descriptor vectors, such as simple random selection or recursive finding of linearly independent vectors in a kernel space [23,24]. Thanks to EKM, tools from BLR can be applied to the sparse matrix \mathbf{K} , allowing for the prediction of errors and uncertainty estimation in the model.

2.4. Nested fitting procedure

One of the major challenges for MLIPs is their poor transferability beyond the training dataset. *Tadah!* addresses this by incorporating a hyperparameter optimization cycle. Unlike standard approaches that primarily fit forces and energies, *Tadah!* employs an iterative procedure to generate “trial” potentials with varying hyperparameters. These are applied via LAMMPS to evaluate macroscopic properties, fitting them to theoretical or experimental data. The best-performing² combination of descriptors and kernels are chosen resulting in increased performance, transferability, and applicability.

² Either in terms of the fit to training data, execution speed, or a custom user-defined metric.

In developing MLIPs, both LPs and HPs are crucial. The model architecture, including the choice of ML algorithm and descriptors, reflects prior knowledge and is fine-tuned through HP optimization. The global objective function, known as the evaluation function in ML literature, plays a central role.

The aim of HP optimization is to minimize the global loss function $\mathcal{L}(\mathcal{M}, \mathcal{T})$, which is a common choice for a global objective function, where \mathcal{M} is the model and \mathcal{T} is the training dataset.

$$\theta^* = \arg \min_{\theta \in \Theta} \mathcal{L}(\mathcal{M}, \mathcal{T}) \quad (6)$$

Here, the model is parameterized by a set of HPs θ , and the goal is to find the θ^* which minimize $\mathcal{L}(\mathcal{M}, \mathcal{T})$ within a search space Θ .

HP optimization differs from other optimization problems. While it is theoretically possible to obtain the gradient of the loss function with respect to the HPs, in practice, this is rare due to discontinuities or non-differentiable search surfaces. Care must be taken to avoid overfitting HPs to a particular dataset, as different datasets may require different optimal HP values.

In general, MLIPs achieve high accuracy for constructing PES for local atomic configurations similar to training data, owing to using generic functions with numerous free parameters. Traditional MLIP model development involves adjusting HPs during manual iterative training and validation, followed by final testing. This approach is simplistic, resource-intensive and time-consuming.

The *Tadah!* global loss function (GLF) is defined as:

$$\mathcal{L}_g(\mathcal{M}, \mathcal{T}) = \sum_{\alpha} \omega_{\alpha} \mathcal{L}_{\alpha}(\mathcal{M}, \mathcal{T}) \quad (7)$$

In this equation, \mathcal{L}_{α} represents the model error associated with the α^{th} constraint, and the ω_{α} are weighting parameters that indicate the importance of α in the fitting procedure. *Tadah!* provides several different loss functions. The simplest constraint loss function takes the following form:

$$\mathcal{L}_{\alpha}(\mathcal{M}, \mathcal{T}) = \left| \mathcal{P}_{\alpha}(\mathcal{M}, \mathcal{T}) - t_{\alpha} \right|^N \quad (8)$$

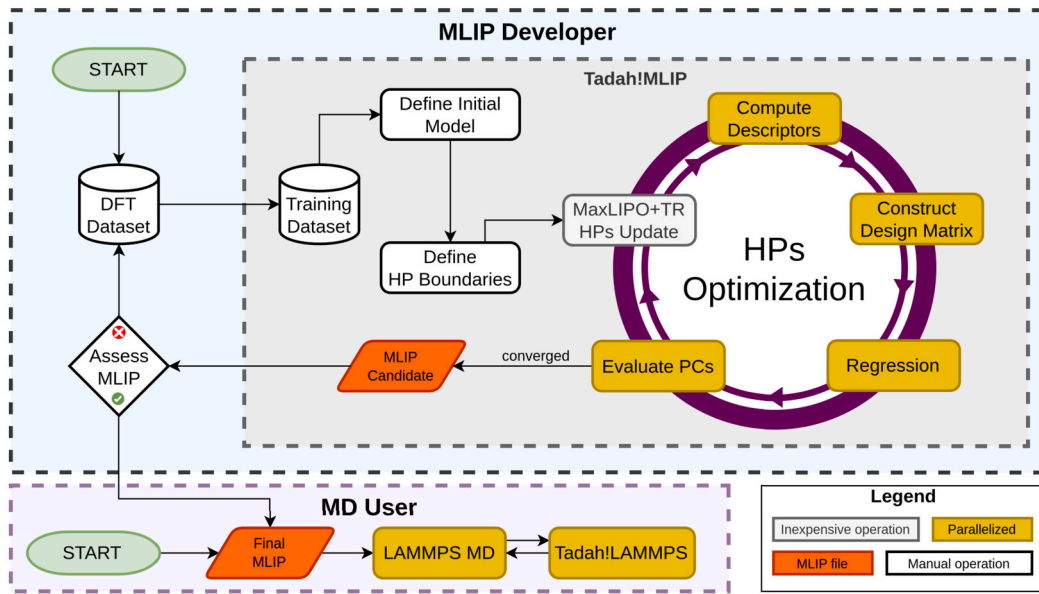


Fig. 3. The hyperparameter optimization routine as implemented in *Tadah!*. The MLIP development starts with constructing a DFT dataset. Users can employ existing datasets or build new ones using external tools. The training stage involves sampling from the DFT data to create a training dataset with the *Tadah!MLIP* toolkit. Next, the MLIP developer defines the model and selects which HPs to optimize, along with setting appropriate search space and performance constraints. Once configured, the automated optimization process begins, generating “trial” potentials and evaluating them against PCs. Upon completion, the MLIP candidate can be manually evaluated by the developer before being distributed to MD users. *Tadah!LAMMPS* is an independent plugin of *Tadah!MLIP* and is required to run MD simulations with the final MLIP.

Here, \mathcal{P}_α denotes the prediction for the α^{th} constraint, with t_α as the target value. The power N determines the type of loss function: setting $N = 1$ results in an absolute loss, while $N = 2$ yields the commonly used quadratic loss function.

The weight factor ω_α has units of the corresponding constraint α raised to the N^{th} power. This ensures that the product in eq. (7) is unitless. The interpretation of the weighting parameters is intuitive—they control the numerical precision of the obtained loss for a given α relative to other weights. For example, doubling a weight makes its associated constraint twice as important as before.

The GLF evaluates the model’s performance not just on the validation set but also by incorporating performance constraints (PC) on the model’s physical predictions. These constraints enhance the predictive power of the interatomic potential. PCs can be divided into two types: the RMSE fit to target vector \mathbf{t} (eq. (5)), and those that are physically motivated, such as the model’s ability to reproduce specific surface energies or energy differences between crystal structures. *Tadah!* provides a flexible interface that allows users to use custom LAMMPS scripts to evaluate the PCs of their choice.

In contrast, search space constraints (SSC) define the configurational space for HPs, which the optimization algorithm explores to satisfy the PCs. SSCs are applied more directly to the model’s architecture, influencing parameters like the positions and widths of Gaussians in a descriptor or a cutoff distance.

The global optimization algorithm (GOA), as illustrated in Fig. 3, works iteratively to optimize model architecture with respect to training data, validation sets, search space, and performance constraints. The training and validation data sets are constructed by the user and remain unchanged throughout the process, as *Tadah!* currently lacks tools for automating this step. The success of the potential depends on the quality of the training data and *Tadah!* allows the user to tune the training data to their intended application. The primary goal of the GOA is to enhance model architecture and, consequently, its transferability.

Optimization begins by defining target PCs and SSCs in a configuration file, where the user assigns a weight to each PC indicating its importance. The automated iterative process involves: selecting candidate HPs from SSCs, training the model with new settings, and evaluating

performance against PCs. This cycle repeats until convergence criteria, like a specific GLF value, are met, or until manually stopped. The potential is refined through both changes in HPs and values of \mathbf{w} (Fig. 3 and Fig. 4).

The HP selection process is managed by the *MaxLIPO+TR* algorithm from the Dlib C++ library [24]. An enhancement of the original LIPO algorithm [25], it estimates the Lipschitz constant to construct an upper bound to the objective function, optimizing towards a global maximum. It evaluates points randomly, comparing their upper bounds to find improvements. To address slow convergence near optima, it employs a trust region method (+TR), assuming a quadratic surface to swiftly converge on local optima [26,27]. *Tadah!* also supports multiple global and local optimizers from the NLOpt library [28].

3. Implementation

The *Tadah!* code is open-source and implemented in C++. Developed using Git, it promotes collaboration through a modular structure (see Fig. 5). The software consists of six independent modules that support the user-facing components, *Tadah!MLIP* and *Tadah!LAMMPS*, enabling efficient code reuse. To enable flexible usage as a stand-alone library, it employs generic programming techniques, like class templates, providing compile-time polymorphism for efficient and reusable components.

For command-line interfaces and the LAMMPS plugin, which require run-time polymorphism, *Tadah!* uses a factory method design pattern. This approach allows for selecting model components based on configuration files, combining compile-time efficiency with run-time flexibility. The object-oriented design focuses on performance, using shallow abstraction layers for critical functionalities like descriptor computation to maintain efficiency. API documentation is generated with Doxygen and published online on ReadtheDocs for easy access [20].

Our software development focused on robustness; unit, functional and integration tests are supported by GitLab continuous integration and continuous delivery (CI/CD) to automate processes, catch bugs early, and ensure compatibility with new LAMMPS versions. This approach enables rapid, reliable software releases and simplifies collaboration and the addition of new features.

```

# Global Settings for the Nested Fitting Procedure
MAXCALLS 10000      # Maximum number of optimization cycles
EPS 1e-6            # Convergence threshold for optimization

# Model Parameters to be Optimized within the Search Space Constraints
OPTIM RCUT2B (1,2) 5.0 6.0    # Optimize 1st & 2nd two-body cutoffs within [5.0, 6.0]
OPTIM CGRID2B (1,3-5) 1.0 6.0 # Optimize 1st, 3rd, and 5th CGRID2B within [1.0, 6.0]
OPTIM SGRID2B (1-5) 0.01 0.1  # Optimize SGRID2B within [0.01, 0.1]
OPTIM RCUTMB (3) 1.0 6.0     # Optimize the 3rd many-body cutoff within [1.0, 6.0]

# Performance Constraints
ERMSE 0 100 # Incorporate Energy RMSE (in meV/atom) into the GLF with weight=100

# Example usage of a custom LAMMPS script to evaluate performance at various pressures
LAMMPS --script in.lata --varloss lata_loss 100 --invar P 1 --invar lata_exp 3.3 --outvar lata

```

Fig. 4. A sample configuration file used by the Nested Fitting Procedure, highlighting how performance constraints (PCs) can be integrated. The ERMSE keyword illustrates a basic constraint on the energy RMSE. Additional physics-based constraints can be incorporated via custom LAMMPS scripts, as shown by the LAMMPS --script line. Here, the variable `lata_loss` is defined in a separate script (`in.lata`) to assess how closely the model reproduces certain macroscopic properties (e.g., lattice constants at a given pressure). By adjusting search space constraints (OPTIM entries) and assigning weights to different PCs, users can tailor the optimization to specific applications and balance trade-offs between accuracy, speed, and other desired criteria.

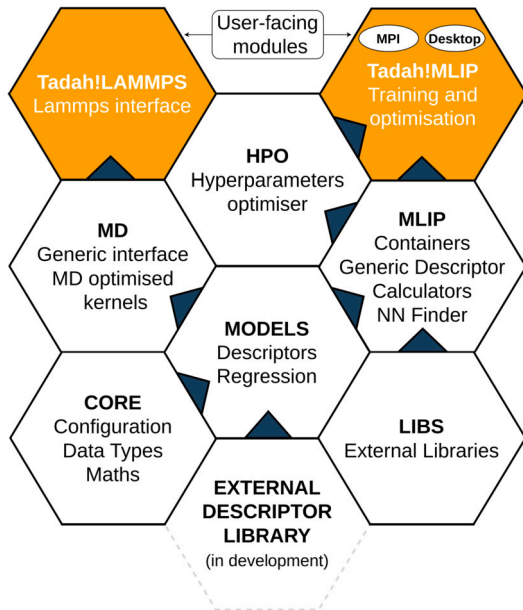


Fig. 5. The *Tadah!* codebase is written in object-oriented C++ and consists of six independent modules, combined to create two user-facing modules. *Tadah!MLIP* is designed for training and optimizing MLIPs, while *Tadah!LAMMPS* deploys them in the MD setting. This modular approach enhances performance, code reusability, and reduces maintenance.

Tadah!MLIP can be compiled on desktops and high-performance computing (HPC) facilities. The desktop version is parallelized with OpenMP and can handle datasets of tens of thousands of configurations. For more demanding tasks, the MPI version, specifically designed for HPC architectures, fully parallelizes descriptor computation and regression using an MPI host-client design pattern, managing extremely large datasets necessary for accurate MLIP parameterization.

3.1. Dependencies and availability

The software is available under the GPLv3 license from <https://git.ecdf.ed.ac.uk/tadah>, and the online documentation is hosted at <https://tadah.readthedocs.io>. The LAMMPS interface and *Tadah!MLIP* toolkit require C++11 and C++17 compatible compilers, respectively. The code has been successfully deployed on a range of Linux architectures, from desktop versions like Alpine or Ubuntu to the HPE Cray Linux Environment, as well as macOS. Fortran routines from LAPACK [29] and

ScaLAPACK [30] (for the *Tadah!* MPI version only) must be available on the user's system. *Tadah!* utilizes CMake for configuration and building of its components. Git and an internet connection are required during the installation process.

The code has not been tested on Windows; however, we expect the LAMMPS plugin to be compatible. For Windows users, we recommend compiling either *Tadah!MLIP* or *Tadah!LAMMPS* using a Linux virtual machine.

The software employs several popular libraries such as Dlib [24], Boost, toml11, and CLI11, which are either contained within the *Tadah!* codebase or automatically downloaded and configured during the installation process when needed for a build.

4. Typical usage

4.1. Using *Tadah!* potentials

Many users are primarily interested in utilizing pre-trained potentials rather than developing them. For this purpose, the *Tadah!LAMMPS* plugin is all they need, as it allows these potentials to be seamlessly integrated with LAMMPS like any other interatomic potential. The potentials are distributed as ASCII files and are generated by the *Tadah!MLIP* software.

To use this feature, users must `git clone` the LAMMPS plugin from the *Tadah!* repository into the `lammps/lib` directory. The remaining compilation process is straightforward and follows the standard LAMMPS library and package installation procedures. For detailed steps, refer to the *Tadah!* documentation [20]. Once compiled, users can invoke *Tadah!* potentials using the standard LAMMPS syntax:

```

pair_style      tadah
pair_coeff       * * pot.tadah ELEMENT1 ELEMENT2

```

where, `pot.tadah` is the filename of the interatomic potential.

4.2. MLIPs development toolkit

This section provides a brief overview of the *Tadah!MLIP* toolkit, designed specifically for the development of Machine Learning Interatomic Potentials (MLIPs). The primary entry point to *Tadah!* is the command-line program `tadah`, which offers various subcommands. These subcommands may require simple configuration files to facilitate MLIPs development. For examples and detailed explanations, please refer to the online documentation [20].

Key subcommands include:

data: Offers dataset manipulation functionalities such as joining, splitting, finding duplicate structures, and selecting subsets. **train:** Performs regression training on a given dataset. **predict:** Predicts energy, forces, and stresses. **hpo:** Conducts hyperparameter optimization and training using nested fitting procedure as described in 2.4. **write:** Dumps dataset configurations into formats like CASTEP `.cell`, VASP `POSCAR`, or LAMMPS data files. **convert:** Assists in extracting relevant data (atomic positions, energies and so on) from CASTEP (`.md`, `.geom`, or `.castep`) or VASP (`OUTCAR` or `vasprun.xml`) DFT calculations to construct training datasets. **analysis:** Provides utilities to plot and visualize cutoff functions, basis functions (such as Gaussians), and descriptors. **properties:** Computes the interaction energy between two atoms using a trained MLIP.

The CLI tools are well-documented and offer helpful descriptions when used with the `-h` or `--help` flags. For instance, `tadah data -h` provides guidance on dataset subcommands. Further documentation is available online.

5. Limitations

Tadah! does not support predefined bonds; hence, everything is treated as atomic and intramolecular bonding must be learned from the dataset. This means that bond breaking should be treated with extreme caution unless included in the training data. *Tadah!* also lacks schemes for incorporating long-distance interactions.

5.1. Ongoing development

Tadah! remains under development. The framework was designed to make it easy to implement enhancements.

Notably, neural networks are currently not supported but could be plugged in to replace the regression step. Implementing new descriptors such as 3-body, MACE or ACE that deviate from the functional forms of eq. (2) or eq. (3) will require modifications to the *Tadah!* codebase.

5.2. Applications

A range of applications are under way using the *Tadah!* code. As with all MLIPs, these are currently subject to a thorough testing for stability and transferability. To date, two *Tadah!* potentials have been published, one for krypton [31] and one for nitrogen [16,17]. These were both parameterized using high-quality coupled cluster quantum chemistry databases rather than DFT, and are designed for execution speed. Both have been initially applied to the fluid state and show excellent agreement with neutron scattering data. The krypton potential outperforms existing models for melt curve and equation of state calculations. The nitrogen potential shows remarkable transferability, describing no fewer than seven crystal phases. Full details can be found in the cited references.

CRedit authorship contribution statement

Marcin Kirsz: Writing – original draft, Software, Methodology, Conceptualization. **Ayobami Daramola:** Writing – review & editing, Software. **Andreas Hermann:** Writing – review & editing, Funding acquisition. **Hongxiang Zong:** Supervision, Conceptualization. **Graeme J. Ackland:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Rights retention statement

For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used Edinburgh University's own platform ELM (Edinburgh Language Model) in order to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Funding

This work was supported by the UK national high-performance computing service, ARCHER2, for which access was obtained via the UKCP consortium and funded by EPSRC grant ref EP/X035891/1. M.K. was supported by the ARCHER2 eCSE software development programme, project ARCHER2-eCSE11-11.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank C.G. Pruteanu for their insightful reading and comments, and A. J. Iwasaki for testing the software and providing valuable feedback.

Data availability

The software is available from: <https://git.ecdf.ed.ac.uk/tadah>.

References

- [1] V.L. Deringer, M.A. Caro, G. Csányi, Machine learning interatomic potentials as emerging tools for materials science, *Adv. Mater.* 31 (46) (2019) 1902765, <https://doi.org/10.1002/adma.201902765>, <https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.201902765>.
- [2] I. Batatia, D.P. Kovacs, G. Simm, C. Ortner, G. Csanyi, Mace: higher order equivariant message passing neural networks for fast and accurate force fields, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), *Advances in Neural Information Processing Systems*, vol. 35, Curran Associates, Inc., 2022, pp. 11423–11436, https://proceedings.neurips.cc/paper_files/paper/2022/file/4a36c3c51af11ed9f34615b81edb5bbc-Paper-Conference.pdf.
- [3] A.P. Bartók, M.C. Payne, R. Kondor, G. Csányi, Gaussian approximation potentials: the accuracy of quantum mechanics, without the electrons, *Phys. Rev. Lett.* 104 (2010) 136403, <https://doi.org/10.1103/PhysRevLett.104.136403>, <https://link.aps.org/doi/10.1103/PhysRevLett.104.136403>.
- [4] H. Yanxon, D. Zagaceta, B. Tang, D.S. Matteson, Q. Zhu, Pyxtal_ff: a python library for automated force field generation open access received pyxtal_ff: a python library for automated force field generation, *Mach. Learn.: Sci. Technol.* 2 (2021) 27001, <https://doi.org/10.1088/2632-2153/abc940>.
- [5] H. Wang, L. Zhang, J. Han, W. E, Deepmd-kit: a deep learning package for many-body potential energy representation and molecular dynamics, *Comput. Phys. Commun.* 228 (2018) 178–184, <https://doi.org/10.1016/j.cpc.2018.03.016>, <https://www.sciencedirect.com/science/article/pii/S0010465518300882>.
- [6] L. Himanen, M.O. Jäger, E.V. Morooka, F. Federici Canova, Y.S. Ranawat, D.Z. Gao, P. Rinke, A.S. Foster, Dscribe: library of descriptors for machine learning in materials science, *Comput. Phys. Commun.* 247 (2020) 106949, <https://doi.org/10.1016/j.cpc.2019.106949>, <https://www.sciencedirect.com/science/article/pii/S0010465519303042>.
- [7] Y. Lysogorskiy, C. van der Oord, A. Bochkarev, S. Menon, M. Rinaldi, T. Hammer-schmidt, M. Mrovec, A. Thompson, G. Csányi, C. Ortner, R. Drautz, Performant implementation of the atomic cluster expansion (pace) and application to copper and silicon, *npj Comput. Mater.* 7 (1) (2021) 1–12, <https://doi.org/10.1038/s41524-021-00559-9>, <https://www.nature.com/articles/s41524-021-00559-9>.
- [8] I.S. Novikov, K. Gubaev, E.V. Podryabinkin, A.V. Shapeev, The mlip package: moment tensor potentials with mpi and active learning, *Mach. Learn.: Sci. Technol.* 2 (2) (2020) 025002, <https://doi.org/10.1088/2632-2153/abc9fe>.
- [9] K.T. Schütt, P. Kessel, M. Gastegger, K.A. Nicoli, A. Tkatchenko, K.R. Müller, Schnet-pack: a deep learning toolbox for atomistic systems, *J. Chem. Theory Comput.* 15 (2019) 448–455, <https://pubs.acs.org/doi/full/10.1021/acs.jctc.8b00908>.

- [10] X. Gao, F. Ramezanghorbani, O. Isayev, J.S. Smith, A.E. Roitberg, Torchani: a free and open source pytorch-based deep learning implementation of the ani neural network potentials, *J. Chem. Inf. Model.* 60 (7) (2020) 3408–3415, <https://doi.org/10.1021/acs.jcim.0c00451>, pMID: 32568524.
- [11] M. Rupp, E. Küçükbenli, G. Csányi, Guest editorial: special topic on software for atomistic machine learning, *J. Chem. Phys.* 161 (6) (2024) 060401, <https://doi.org/10.1063/5.0228461>, https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0228461/20104152/060401_1_5.0228461.pdf.
- [12] L. Yang, A. Shami, On hyperparameter optimization of machine learning algorithms: theory and practice, *Neurocomputing* 415 (2020) 295–316, <https://doi.org/10.1016/j.neucom.2020.07.061>, <https://www.sciencedirect.com/science/article/pii/S0925231220311693>.
- [13] L. Fiedler, N. Hoffmann, P. Mohammed, G.A. Popoola, T. Yovell, V. Oles, J. Austin Ellis, S. Rajamanickam, A. Cangi, Training-free hyperparameter optimization of neural networks for electronic structures in matter, *Mach. Learn.: Sci. Technol.* 3 (4) (2022) 045008, <https://doi.org/10.1088/2632-2153/ac9956>.
- [14] D.F. Thomas du Toit, V.L. Deringer, Cross-platform hyperparameter optimization for machine learning interatomic potentials, *J. Chem. Phys.* 159 (2) (2023) 024803, <https://doi.org/10.1063/5.0155618>, https://pubs.aip.org/aip/jcp/article-pdf/doi/10.1063/5.0155618/18281499/024803_1_5.0155618.pdf.
- [15] D.F.T. du Toit, Y. Zhou, V.L. Deringer, Hyperparameter optimization for atomic cluster expansion potentials, *J. Chem. Theory Comput.* 8 (2024), <http://arxiv.org/abs/2408.00656>.
- [16] M. Kirsz, C.G. Pruteanu, P.I.C. Cooke, G.J. Ackland, Understanding solid nitrogen through molecular dynamics simulations with a machine-learning potential, *Phys. Rev. B* 110 (2024) 184107, <https://doi.org/10.1103/PhysRevB.110.184107>, <https://link.aps.org/doi/10.1103/PhysRevB.110.184107>.
- [17] C.G. Pruteanu, M. Kirsz, G.J. Ackland, Frenkel line in nitrogen terminates at the triple point, *J. Phys. Chem. Lett.* 12 (47) (2021) 11609–11615, <https://doi.org/10.1021/acs.jpclett.1c03206>, pMID: 34812632.
- [18] 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, S.J. Plimpton, LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, *Comput. Phys. Commun.* 271 (2022) 108171, <https://doi.org/10.1016/j.cpc.2021.108171>.
- [19] Y. Zhang, C. Hu, B. Jiang, Embedded atom neural network potentials: efficient and accurate machine learning with a physically inspired representation, *J. Phys. Chem. Lett.* 10 (17) (2019) 4962–4967, <https://doi.org/10.1021/acs.jpclett.9b02037>, pMID: 31397157.
- [20] M. Kirsz, G.J. Ackland, Tadah! Machine learning interatomic potentials software, <https://tadah.readthedocs.io>, 2024.
- [21] C.M. Bishop, *Machine Learning and Pattern Recognition*, Springer-Verlag, 2006.
- [22] B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, A.J. Smola, Input space versus feature space in kernel-based methods, *IEEE Trans. Neural Netw.* 10 (5) (1999) 1000–1017, <https://doi.org/10.1109/72.788641>, cited by: 1065.
- [23] Y. Engel, S. Mannor, R. Meir, The kernel recursive least-squares algorithm, *IEEE Trans. Signal Process.* 52 (8) (2004) 2275–2285, <https://doi.org/10.1109/TSP.2004.830985>.
- [24] D.E. King, Dlib-ml: a machine learning toolkit, *J. Mach. Learn. Res.* 10 (2009) 1755–1758.
- [25] C. Malherbe, N. Vayatis, Global optimization of Lipschitz functions, in: *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, ICML'17, JMLR.org, 2017, pp. 2314–2323.
- [26] S.M. Goldfeld, R.E. Quandt, H.F. Trotter, Maximization by quadratic hill-climbing, *Econometrica* 34 (3) (1966) 541, <https://doi.org/10.2307/1909768>.
- [27] D.C. Sorensen, Newton's method with a model trust region modification, *SIAM J. Numer. Anal.* 19 (2) (1982) 409–426, <https://doi.org/10.1137/0719026>, <http://epubs.siam.org/doi/10.1137/0719026>.
- [28] Steven G. Johnson, The Nlopt nonlinear-optimization package, <https://github.com/stevengj/nlopt>, 2007.
- [29] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, *LAPACK Users' Guide*, 3rd edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
- [30] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R.C. Whaley, *ScaLAPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.
- [31] A.J. Iwasaki, M. Kirsz, C.G. Pruteanu, G.J. Ackland, An accurate machine-learned potential for krypton under extreme conditions, *J. Phys. Chem. Lett.* 16 (2025) 1559–1566.