CURATOR: Autonomous Batch Active-Learning Workflow for Catalysts

Xin Yang xinyang@dtu.dk
Renata Sechi resa@dtu.dk
Martin Hoffmann Petersen mahpe@dtu.dk
Arghya Bhowmik arbh@dtu.dk

Heine Anton Hansen
heih@dtu.dk
Department of Energy Conversion and Storage
Technical University of Denmark
Kgs. Lyngby 2800, Denmark

Abstract

Machine learning interatomic potentials (MLIPs) enable molecular simulations at longer time scales without compromising accuracy and at lower computational costs compared to electronic structure methods such as density functional theory (DFT). Application of MLIPs to complex functional-materials development can help to create new scientific insights, however, MLIPs need ad-hoc training for each new system. Reaching sufficient accuracy through large-scale training is data-intensive, and requires a high level of technical proficiency from the user. Reliable MLIP construction requires an appropriate selection of representative structures and calibrated model uncertainty while avoiding undersampling of the state space. Currently, there is a lack of end-to-end automated software to take this complexity away from the end user. In this tutorial, we show how to use CURATOR, an open-source software-based autonomous batch active learning workflow. CURATOR trains message-passing graph neural networks and enables management of model training, production testing, data selection based on uncertainty estimation, optimal batch choice, labeling via DFT-based simulations, and retraining in a user-friendly way.

1 Introduction

One of the most challenging problems of molecular simulations of materials is the computational cost of the simulation itself. Comparisons with experiments are challenging as well since the size of the simulated system and the length of the simulation is very small in comparison to real-life experiments. This discrepancy translates into a general slowdown of the discovery process of the properties of the materials in catalytic reactions.
Machine learning interatomic potentials (MLIP) emerged as a promising alternative to ab-initio molecular dynamics (AIMD) simulations because they allow to consider bigger-sized systems and to increment the simulation length (Unke et al. [2021]). As a result, MLIPs enable to explore a larger state space and thus to understand better the material's characteristics (Friederich et al. [2021]). However, ML models initially needed extensive testing and physical/chemical insight of the user for the hyperparameter selection to create the features. To overcome these challenges, end-to-end MLIPs capable of directly learning the mapping from nuclear charges and Cartesian coordinates of atomic structures to atomic features have been recently developed. One example of this is PaiNN (Schütt et al. [2021]), a message-passing neural network for graph representation recently developed

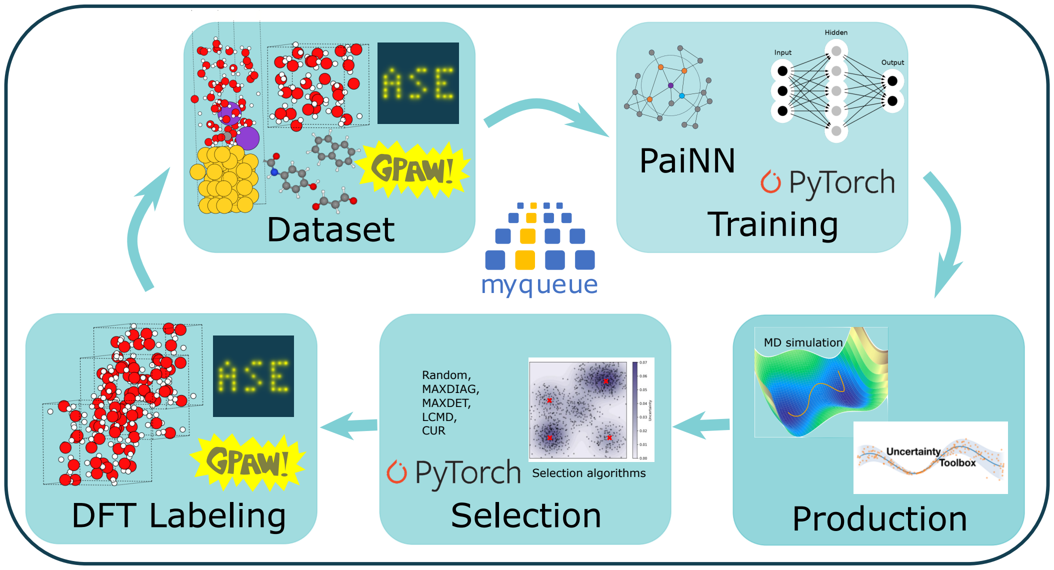


Figure 1: Workflow structure. CURATOR comprehends 5 essential steps, dataset generation, training, production, selection, and Density Functional Theory (DFT) labeling. In the boxes, we indicate the packages and tools whose usage is integrated into the CURATOR workflow.

for treating molecules. Yet, MLIPs trained with an AIMD dataset with undersampled configurational space have limited utility to contribute to material discovery, because the ML production (here, an MD simulation) will represent only a small portion of the state space. A way to treat this problem is to select the most representative structures from the dataset and retrain the model with those. This process of selection and retraining is called active learning. The trained MLIP needs to be aware of the model uncertainty to select these structures. There are several methods for doing it; these methods label the uncertainty of the structures of MLIP simulations based on a certain uncertainty metric, and select the batch of structures with the highest uncertainty for the retraining of the network.

This procedure of training, MD production, labeling, selection, and retraining signifies a lot of data and submitted jobs to manage for the user. For instance, one can train the MLIP on 4 systems for 15 iterations and select 100 structures for DFT labeling for each iteration round. Conviniently, these tasks can be organized with workflows for process management. In this tutorial, we explain the use and the characteristics of CURATOR, an autonomous batch active-learning workflow devised for the construction of high-fidelity graph neural network potentials. This workflow uses equivariant MPNNs, such as PaiNN, to accurately predict properties within chemical systems. To ensure the robustness of simulations driven by the trained MLIPs, our approach supports a variety of methods for uncertainty quantification techniques, as we describe later in this work. These methods allow us to perform batch active learning to efficiently identify the most informative structures from production simulations and expand their applicability across a broader chemical space. The workflow management is autonomous and user-friendly thanks to the integration of MyQueue (Mortensen et al. [2020]). In this way, the user can always maintain an overview of the status of the jobs, the workflow can be initiated and tasks can be resubmitted, stopped, and iterated. Usually, we use MyQueue for handling tasks with Slurm on the HPC; however, the notebook for this tutorial runs on Colab and we utilize MyQueue in the "local" configuration, thus only with the Colab resources. In this tutorial, we explain the functionality of the workflow, provide a step-by-step tutorial in the form of a Jupyter notebook based on GPAW (Enkovaara et al. [2010]) calculators applied on a water box, and explain the workflow output.

2 Workflow structure

The structure of our workflow is displayed in Figure 1; a standard run comprehends 5 steps: Dataset generation, training, production, selection, and labeling. Typically, we run the workflow iteratively to enhance the quality of the MLIP, e.g., for refining node features. In the following, we briefly present the basic functions of each step.

Dataset As the workflow's first step,a training dataset of an ab-initio molecular dynamics simulation is needed. To do this, here we use GPAW since we intend to use only open-source codes for this tutorial; however, it is also possible to use VASP (Kresse and Furthmüller [1996]). The dataset is stored as ASE trajectory files (Larsen et al. [2017]).

Training The second step of the workflow is the training of the machine learning algorithm. Hereby, we use PaiNN as model and train it on energy and forces. We introduce an efficient method for the gradient computation, which calculates forces and stress based on the energy derivative with respect to relative position vectors (Yang et al. [2023]), see explanation in 5.1. This procedure improves both simulation speed and memory usage, enabling the use of a larger training batch size and further enhancing model performance. It is possible to specify several models to train at the same time, each of which with different number of nodes and layers. When having multiple iterations, weights and biases are initialized by loading the models' checkpoints from the previous iteration cycle. In principle, other message-passing neural networks can be implemented at this place.

Data production- MD run After the training of the MLIP, the workflow passes the trained model to data production. In this case, this is a ASE molecular dynamics simulation. The user can indicate the desired simulation conditions for the MD production run. Based on the training and on the system, one can adjust the parameters for the molecular dynamics production, such as the time step or the simulation length.

Batch-mode selection Next, we perform batch activate learning, which is possible using several strategies for the selection. To do this, we estimate the uncertainty of each frame of the data produced by the trained model. This procedure maps the structures into a feature kernel matrix that we can analyze in different ways to select the most informative ones. Active learning can be performed in mainly two ways, by naive active learning, which prioritizes the uncertainty for the selection, or by using optimal batch active learning methods that consider both uncertainty and diversity for the selection of the batch. The disadvantage of naive active learning is that it selects only multiple informative but similar samples for the next iteration, thus resulting in a limited improvement of the model between one training iteration and the next one. The second class of methods, instead, try to select structures that are informative, while minimizing the redundancy (e.g., similar subsequent structures) among the samples in the batch. In the configuration file, the user specifies which method will be used for selection and how many structures will be selected for labeling. The methods for the batch-mode selection are random, naive active learning (MAXDIAG), greedy determinant maximization (MAXDET), Largest cluster maximum distance (LCMD), see (Zaverkin et al. [2022], Yang et al. [2023]) for further description of features kernel matrices and selection methods.

The performance of the available selection methods and feature kernels is displayed in Figure 2, in which we report the results for the MD17 dataset Chmiela et al. [2017].

Labeling As a final step of a workflow cycle, perform a DFT single-point calculation of the selected structures. The DFT calculation has the same level of theory as the initial dataset to enable the later training. The structures are then stored in an ASE trajectory file.

3 Run the workflow

For this tutorial, we use a system of 4 water molecules moving in a 5 Å cubic box and perform Langevin molecular dynamics simulations. See the appendix 5.2 for the simulation details. Link to the GitHub repository `https://github.com/rena-96/curator-ai4mat` and to the tutorial in the form of a YouTube video `https://youtu.be/9QerCuRIuSM`.

The installation of this workflow requires previous installation of PyTorch Scatter, TOML, MyQueue packages, and the possibility of using GPUs. CURATOR has only been tested for Python>=3.8.0 and PyTorch>=1.10. Once installed the workflow package, the user creates a new directory `my_project` and copies the workflow scripts from the folder `script`. So that the context of the `my_project` directory looks like:

```
$ config.toml flow.py train.py labeling.py select.py
$ md_run.py my_training_set.traj
```

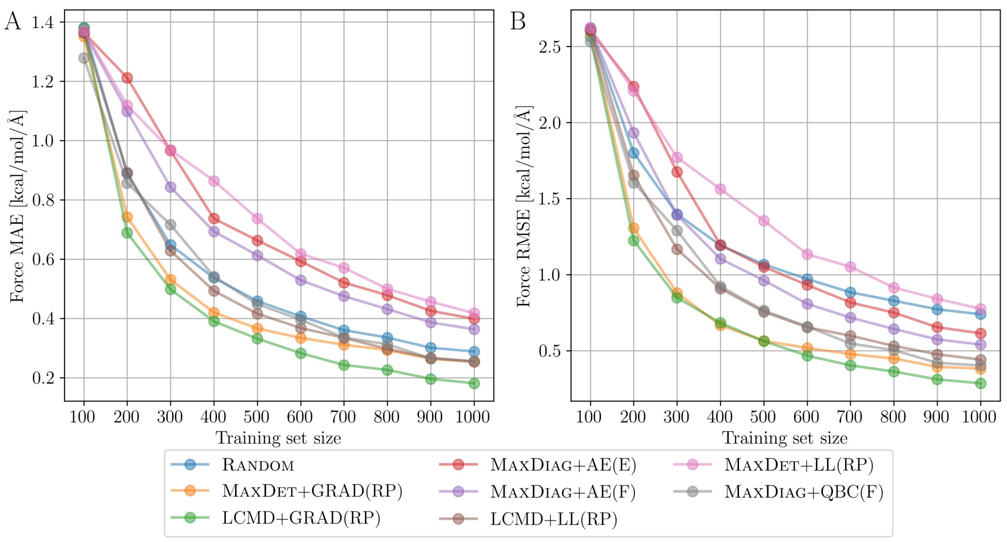


Figure 2: Model performance for forces by using different choices and combinations of feature kernel and selection methods for the MD17 dataset. Left: mean absolute error (MAE) [kcal/mol/Å]; Right: root mean square error (RMSE)[kcal/mol/Å] .

Then MyQueue is initiated in the same directory to create a folder `./myqueue` where the configuration files for the jobs submission will be stored. The configuration files can be for example the characteristics of the machines in the cluster, the status of the jobs, etc. After setting up the `config.toml` file (see SI), the user runs `mq workflow flow.py` to submit all the workflow tasks. The workflow parameters are organized and forwarded to MyQueue using TOML. In `config.toml` all necessary parameters are set for the workflow and they can be edited for each iteration. Thanks to MyQueue the user can visualize the status of each single job in the queue. MyQueue will submit the jobs after the others until reaches the number of iterations specified in the `config.toml` file. We describe the output structure in detail in the supplementary information.
In Figure 3, we analyze the MLIP from the fourth iteration (i.e., iteration 3) of a 4 water molecules system. The training shows the fourth iteration results of an NN with 124 nodes and 3 layers, the production run the temperature and the energy of the ML molecular simulation as a function of production time, and the mean squared displacement (MSD) of the water molecules in the ML molecular simulation. As one can see, the hydrogen atoms move more than the oxygens. This indicates that the model needs more iterations in order to equilibrate the MLIP and also we are at the first steps of the simulations. With more MD steps and more workflow iterations, the model can equilibrate. The novelty of the workflow is that one can easily check on each step and visualize the product simulation and analyze the data, as done hereby.

4 Conclusion and outlook

In this tutorial, we presented the CURATOR workflow, and its main features, and showed how to use it with the example of a simulation of a water box. This workflow runs using only open-source software and has a variety of selection methods for batch-active learning. The training itself is made easier by the efficient computation of the forces. In the future, we aim to integrate other message-passing neural network codes, improve the uncertainty estimation for the production, and introduce the nudged elastic band method besides MD. Most importantly, our immediate goal is to improve the installation and workflow configuration process.

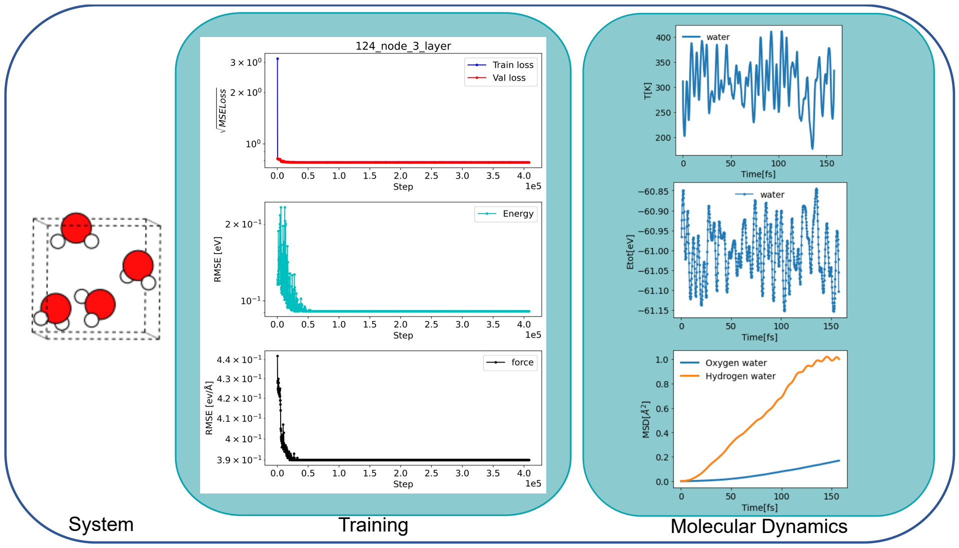


Figure 3: Fourth iteration of the workflow for the 4 water molecules example. On the left, we display the system we are training on; on the center, an example of the training in the PaiNN model; on the right, we analyze some quantities from the molecular simulation produced with the training of the model.

5 Supplementary Material

5.1 Forces calculation

The GNN model computes total potential energy E and forces as the sum of individual atomic components (Chmiela et al. [2017]), i.e.

$$E = \sum_{i \in N} E_i \tag{1}$$

$$\text{and} \quad \vec{F_i} = -\nabla_i E \tag{2}$$

Instead, they rely solely on the relative position vector $\vec{r_{ij}} = \vec{r_j} - \vec{r_i}$ and its length $\| \vec{r_{ij}} \|$ in the message layers, which are typically obtained via neighbor-list algorithms from various codes like ASE, ASAP3, MatScipy, or NNPOps. Therefore, the atomic energy is exclusively a function of $\vec{r_{ij}}$:

$$E_i = E_i(\{\vec{r_{ij}}\}_{i \neq j}) \tag{3}$$

Our implementation calculates forces $\vec{F_i}$ of atom i as negative gradients of the total potential energy with respect to the model descriptors (Yang et al. [2023]), i.e. relative position vectors $\vec{r_{ij}}$:

$$\begin{aligned}
\vec{F_i} &\equiv -\frac{\partial E}{\partial \vec{r_i}} \equiv -\sum_i \frac{\partial E_i}{\partial \vec{r_i}} \\
&= -\sum_{j \neq i} \left(\frac{\partial E_j}{\partial \vec{r_i}} \right) - \frac{\partial E_i}{\partial \vec{r_i}} \\
&= -\sum_{j \neq i} \left(\sum_{k \neq j} \frac{\partial E_j}{\partial \vec{r_{jk}}} \frac{\partial \vec{r_{jk}}}{\partial \vec{r_i}} + \frac{\partial E_i}{\partial \vec{r_{ij}}} \frac{\partial \vec{r_{ij}}}{\partial \vec{r_i}} \right) \\
&= -\sum_{j \neq i} \left(\frac{\partial E_i}{\partial \vec{r_{ij}}} - \frac{\partial E_j}{\partial \vec{r_{ji}}} \right).
\end{aligned} \tag{4}$$

In this way, the forces can be computed with only $-\partial E / \partial \vec{r_{ij}}$ that can be directly obtained with automatic differentiation (see e.g., PyTorch (Paszke et al. [2017])). With $\vec{F_i} = \sum_{j \neq i} \vec{F_{ij}}$, where

$\vec{F_{ij}}$ is the force exerted by atom j on atom i, our implementation in equation 4 enables a pairwise computation of the forces:

$$\vec{F_{ij}} = -\vec{F_{ji}} = -\left(\frac{\partial E_i}{\partial \vec{r_{ij}}} - \frac{\partial E_j}{\partial \vec{r_{ji}}}\right). \tag{5}$$

Furthermore, this antisymmetric property is advantageous for reducing the effort required to compute the stress of the chemical system by using explicit analytical expressions for virial tensors (Knuth et al. [2015]). The virial tensors can then be calculated by:

$$W = \sum_i W_i = -\frac{1}{2}\sum_i \sum_{j \neq i} r_{ij} \otimes \vec{F_{ij}}. \tag{6}$$

5.2 AIMD simulation details

We use ASE to create a cubic box of 4 water molecules. The box has an edge of 5 Å. We set up a GPAW calculator with `mode="fd"` and PBE functional. We perform Langevin dynamics at temperature $T = 300K$ and friction 0.01. After equilibrating the trajectory, we perform another simulation of 5 ps and timestep 0.5fs. We use this trajectory as a training dataset for the example of the run of the workflow.

5.3 Configuration file

In the following, we report the TOML file, i.e., the configuration file the workflow uses to parse the arguments of each step. The conf.toml is the file in which we build the workflow; it has several blocks, each of them starting with the square brackets `[keyword]`. The `[keyword.feature]` means that this block is setting up a specific feature of the keyword. The keywords are the different steps of the workflow: train, MD, select, and label. The features can be, depending on the keyword, .ensemble, .runs, .resource, etc.

The structure of each line is `key = value`. If value is a string, then it is written between single or double quotation marks, if value is a boolean or a number, then quotation marks are not needed.

```
[global]
root = '.'                    # Root directory
random_seed = 3407

[train]
cutoff = 1.5                 # Cutoff radius of machine learning potential
val_ratio = 0.1              # The ratio of validation points in the provided dataset
num_interactions = 3         # Number of message-passing layers
node_size = 128              # Node feature size
output_dir = 'model_output' # Model output path
dataset = 'path/to/file/moldyn3.traj' # Dataset for training
max_steps = 500000            # Maximum steps for training
device = 'cuda'              # Use GPU
batch_size = 16              # Batch size for training
initial_lr = 0.0001          # Initial learning rate
forces_weight = 0.95         # Ratio of force loss to total loss
log_interval = 2000          # Evaluate model every 2000 steps
normalization = false        # Normalize energy in the dataset
atomwise_normalization = false  # Normalize atomic energy, scale
                             # the output atomic energy to the same level.
stop_patience = 1000         # When test loss is larger than training loss
                             # for p times, training stops.
plateau_scheduler = true     # Use ReduceLROnPlateau scheduler
                             # to decrease lr when learning curve plateaus
random_seed = 3407           # Random seed ensures the reproducibility of experiments
```

```
[train.ensemble]
# For training multiple models in parallel

108_node_3_layer = {node_size = 108, num_interactions = 3}

112_node_3_layer = { node_size = 112, num_interactions = 3}

116_node_3_layer = { node_size = 116, num_interactions = 3}

124_node_3_layer = { node_size = 124, num_interactions = 3}

[train.resource]
nodename = 'my-gpu-node'    # Specify the node for training
tmax = '3d'                 # Time limit for each job
cores = 8                   # Cores on the node

[MD.runs.water]
# Parameters for MD
init_traj = 'path/to/file/moldyn3.traj'  #Initial configuration for running MD
time_step = 0.1            # Time step for MD
temperature = 300          # Temperature for MD
device = 'cuda'            # Use GPU
start_indice = 1           # Select initial configuration
max_steps = 20000          # Maximum MD steps
min_steps = 251            # Minimum MD steps
dump_step = 5              # Dump a structure for every 100 steps
print_step = 2             # Print MD info for every 2 steps
num_uncertain = 10         # If 10 uncertain structures are collected,
                           #the simulation stops
random_seed = 3407         # Reproducibility

[MD.resource]
nodename = 'my-gpu-node'
tmax = '7d'
cores = 8

[select.runs]
water = {
    'method' = 'MD',
    'train_set' = 'path/to/file/moldyn3.traj',
    'kernel' = 'full-g',          # Features kernel matrix
    'selection' = 'lcmd_greedy',  # Selection method
    'n_random_features' = 500,    # Number of random projections
    'batch_size' = 10,            # No. selected structures  for active learning
    'device' = 'cuda',
    'random_seed' = 3407
}

[select.resource]
nodename = 'my-gpu-node'
tmax = '7d'
cores = 8

[labeling.runs]
method = 'GPAW'  # Use a GPAW calculator for the DFT labeling

[labeling.runs.water]
nupdown = 48
```

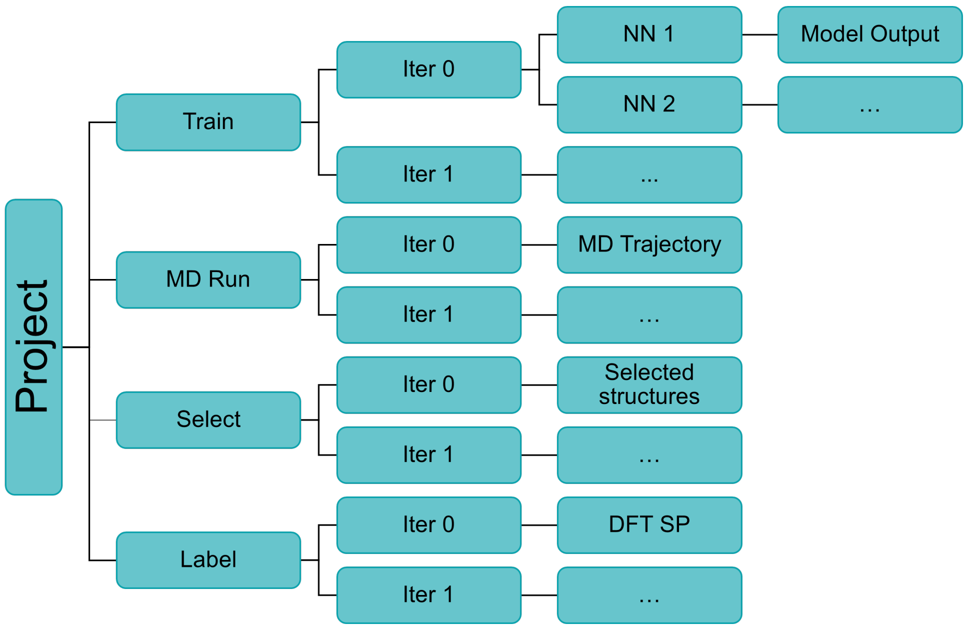
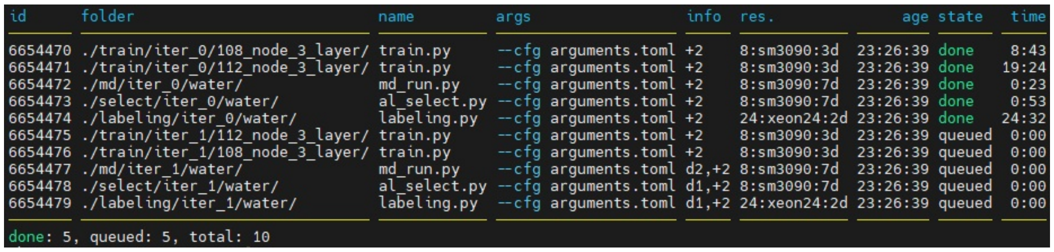


Figure 4: Structure of the workflow output. Each rectangle represents a directory with the output of one step of the workflow. The use of the dots indicates that the output has the same structure as the previous iteration. Abbreviations: molecular dynamics (MD), neural network (NN), density functional theory (DFT), single-point calculation (SP).



```
id      folder                          name         args                     info  res.              age state   time

6654470 ./train/iter_0/108_node_3_layer/ train.py     --cfg arguments.toml +2    8:sm3090:3d  23:26:39 done     8:43
6654471 ./train/iter_0/112_node_3_layer/ train.py     --cfg arguments.toml +2    8:sm3090:3d  23:26:39 done    19:24
6654472 ./md/iter_0/water/               md_run.py    --cfg arguments.toml +2    8:sm3090:7d  23:26:39 done     0:23
6654473 ./select/iter_0/water/           al_select.py --cfg arguments.toml +2    8:sm3090:7d  23:26:39 done     0:53
6654474 ./labeling/iter_0/water/         labeling.py  --cfg arguments.toml +2    24:xeon24:2d 23:26:39 done    24:32
6654475 ./train/iter_1/112_node_3_layer/ train.py     --cfg arguments.toml +2    8:sm3090:3d  23:26:39 queued   0:00
6654476 ./train/iter_1/108_node_3_layer/ train.py     --cfg arguments.toml +2    8:sm3090:3d  23:26:39 queued   0:00
6654477 ./md/iter_1/water/               md_run.py    --cfg arguments.toml d2,+2 8:sm3090:7d  23:26:39 queued   0:00
6654478 ./select/iter_1/water/           al_select.py --cfg arguments.toml d1,+2 8:sm3090:7d  23:26:39 queued   0:00
6654479 ./labeling/iter_1/water/         labeling.py  --cfg arguments.toml d1,+2 24:xeon24:2d 23:26:39 queued   0:00

done: 5, queued: 5, total: 10
```

Figure 5: Output of the command `mq ls`, with which it is possible to visualize the job status of the workflow.

```
[labeling.resource]
nodename = 'my-node-24'
tmax = '2d'
cores = 24
```

5.4 Output structure

After running `flow.py`, the output of different jobs will be organized into various directories and subdirectories. The schema in Figure 4 illustrates the output structure of the workflow. In Python logic, the workflow names the iterations starting from zero.

While the workflow is running, it is possible to visualize the job information and status with the command `mq ls`. This feature is particularly convenient for errors related to the workflow, such as big uncertainty, wrong batch size, etc. Figure 5 shows a case in which all the jobs of the first iteration are completed and the jobs of the second iteration are queued. The columns indicate, from the left to the right, the job id, the path to the output, the file that has been executed, the `toml` file that the Python file is parsing, the specifics of the job, the age of the job from the submission point, the state and the time it needed to complete it.

References

Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, 3(5):e1603015, 2017. doi: 10.1126/sciadv.1603015. URL `https://www.science.org/doi/abs/10.1126/sciadv.1603015`.

J. Enkovaara, C. Rostgaard, J. J. Mortensen, J. Chen, M. Dułak, L. Ferrighi, J. Gavnholt, C. Glinsvad, V. Haikola, H. A. Hansen, H. H. Kristoffersen, M. Kuisma, A. H. Larsen, L. Lehtovaara, M. Ljungberg, O. Lopez-Acevedo, P. G. Moses, J. Ojanen, T. Olsen, V. Petzold, N. A. Romero, J. Stausholm-Møller, M. Strange, G. A. Tritsaris, M. Vanin, M. Walter, B. Hammer, H. Häkkinen, G. K. H. Madsen, R. M. Nieminen, J. K. Nørskov, M. Puska, T. T. Rantala, J. Schiøtz, K. S. Thygesen, and K. W. Jacobsen. Electronic structure calculations with GPAW: a real-space implementation of the projector augmented-wave method. *J. Phys.: Condens. Matter*, 22(25):253202, 2010. doi: 10.1088/0953-8984/22/25/253202.

Pascal Friederich, Florian Häse, Jonny Proppe, and Alán Aspuru-Guzik. Machine-learned potentials for next-generation matter simulations. *Nature Materials*, 20(6):750–761, 2021.

Franz Knuth, Christian Carbogno, Viktor Atalla, Volker Blum, and Matthias Scheffler. All-electron formalism for total energy strain derivatives and stress tensor components for numeric atom-centered orbitals. *Computer Physics Communications*, 190:33–50, 2015.

Georg Kresse and Jürgen Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational materials science*, 6(1):15–50, 1996.

Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, Eric D Hermes, Paul C Jennings, Peter Bjerre Jensen, James Kermode, John R Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W Jacobsen. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017. URL `http://stacks.iop.org/0953-8984/29/i=27/a=273002`.

Jens Jørgen Mortensen, Morten Gjerding, and Kristian Sommer Thygesen. Myqueue: Task and workflow scheduling system. *Journal of Open Source Software*, 5(45):1844, 2020.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.

Oliver T. Unke, Stefan Chmiela, Huziel E. Sauceda, Michael Gastegger, Igor Poltavsky, Kristof T. Schütt, Alexandre Tkatchenko, and Klaus-Robert Müller. Machine learning force fields. *Chemical Reviews*, 121(16):10142–10186, 2021. doi: 10.1021/acs.chemrev.0c01111. URL `https://doi.org/10.1021/acs.chemrev.0c01111`. PMID: 33705118.

Xin Yang, Arghya Bhowmik, Tejs Vegge, and Heine Anton Hansen. Neural network potentials for accelerated metadynamics of oxygen reduction kinetics at au–water interfaces. *Chem. Sci.*, 14:3913–3922, 2023. doi: 10.1039/D2SC06696C. URL `http://dx.doi.org/10.1039/D2SC06696C`.

Viktor Zaverkin, David Holzmüller, Ingo Steinwart, and Johannes Kästner. Exploring chemical and conformational spaces by batch mode deep active learning. *Digital Discovery*, 1:605–620, 2022. doi: 10.1039/D2DD00034B. URL `http://dx.doi.org/10.1039/D2DD00034B`.