nature portfolio

Peer Review File

Online Test-time Adaptation for Better Generalization of Interatomic Potentials to Out-of-distribution Data

Corresponding Author: Shufei Zhang

This manuscript has been previously reviewed at another journal that is not operating a transparent peer review scheme. The manuscript was considered suitable for publication without further review at Nature Communications.

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The study by Cui et al. reports the integration of test-time adaption for interatomic potentials. It reports a novel architecture combining existing frameworks with self-supervised tasks, which can be fine-tuned to the distribution of test data during inference without collecting new data. I was pleased to see the authors included both invariant as well as equivariant architectures for the energy and force predictor, and that their framework can significantly improve model performance for both cases. I think this development is very exciting, and succinctly different from other current approaches such as active learning. I therefore recommend to publish the article in Nature Communications after addressing the following (minor) issues:

1) The methodology needs to be described in more detail, either in the main article or in the Supporting Information. As is, it is for example not clear how the molecular dynanamics experiment was run (some software such as OpenMM, LAMMPS, etc, or an implementation by the authors).

2) Another example: It is not clear whether the fine-tuning is permanent, i.e. changes the model when processing each data point, or whether the fine-tuning is only done for the current test data point, and the next test data point is fine-tuned from the same starting point. If the fine-tuning is permanent, how would the order of test data affect the performance, and is there any guidance on best practices? If the fine-tuning is temporary, this should be clearly stated.

3) Or, as another example among others, it is not clear whether the fine-tuning (if permanent, else disregard this point entirely) was also part of the molecular dynamics trajectory (i.e. the interatomic potential changed throughout the trajectory), or whether the fine-tuning stopped at some point. Along the same line, would the authors recommend to stop the fine-tuning during a production run to compute properties from an MD trajectory (and can one even turn off the fine-tuning with their architecture)? Fully exploring the effect of a changing potential throughout an MD run is beyond the current study, but it would be helpful to provide some comments on best practices.

4) The content of the SI should be mentioned where appropriate. For example, the main article describes an MD simulation only with SchNet, but I then discovered data for PaiNN in the Supplementary Information. There should be a pointer in the text, e.g. that analogous simulations for PaiNN are described in the SI. Similarly, the ablation studies in the SI should definitely be mentioned somewhere in the text, as I believe they are very important to highlight that all three supervised tasks as proposed in the TAIP framework are actually beneficial. For all results in the SI, their existence should be mentioned in the main article.

5) I would suggest to also study the performance of TAIP vs SchNet and PaiNN as a function of the training size. I.e. for one of the datasets, train on differently sized random subsets of the full training data (evaluated on the same test data) and report the performances individually. This should be very quick to do, and I expect that TAIP will lead to a performance increase in the low data regime, too. This would be a very valuable information in the current quest toward low-data machine learning potentials.

6) See the code review below for details. I find that the code repository is currently not fit for distribution (yet), and (in its current state) will not allow the community to adopt this approach. I would sincerely recommend to re-work it, including usage instructions and examples.

(Remarks on code availability)
The provided code does not contain any instructions on how to use it, or how to reproduce the study (e.g. a README file, documentation, etc). The code itself seems like it might work, but it includes several hard-coded paths to the personal hard-drives of some of the authors, so it is not usable without modifications. Furthermore, many of the relevant functions do not contain any descriptions or doc-strings, so that re-using the code in a different project seems difficult, especially given that there is no README file to guide the user. For a paper that introduces a new architecture and probably wants the community to utilize and build on their methods, the code repository is in a very poor state, and needs to be reworked before publication. Since the study uses open-source datasets, I would furthermore suggest to include a full example of training and testing that a user can follow along and reproduce.

Reviewer #2

(Remarks to the Author)
The authors present a self-supervised approach for improving the accuracy and stability of machine learning interatomic potentials after the initial fitting stage, without the need for additional reference data. The results are moderately encouraging - the errors are reduced modestly, and the stability of MD simulation is much better. However, as is the habit of machine learning papers, the method is completely inadequately described. The standard for scientific literature, which is that readers should be able to reproduce the results, is not even vaguely close to being satisfied. Unfortunately, this is entirely par for the course in the ML literature, but it is also entirely unacceptable. If the authors fully explain their method, the paper might be suitable for publication. Without that, it's simply not a scientific publication.

There are many, many places where crucial information is missing. One essential problem is that there is not a step by step explanation of what _exactly_ happens at the initial training, test-time adaptation, and final inference phase. For example, the only place where the authors say explicitly that only the decoder is re-fit during the adaption is in the introduction, Sec 1. Second, the only parameters that are explicitly given are the sizes of the datasets, and, in the supplementary, the hyperparameters for the MLIP fitting. What about the parameters of the auxilliary loss functions (magnitude of noise L, feature recovery exponent gamma, etc)? What about the architectures of the auxilliary loss decoders? Reading this paper, I have _no idea_ how to reproduce what the authors did.

Other issues

1. MD with MLIPs breaks down even without formal rare events (as that term is usually defined, e.g. transitions over moderate energy barriers into new configuration space) or phase transitions. The lack of sufficient smoothness in nearly all MLIPs (which leads to poor extrapolation) is enough that MD will almost invitably explore configurations far enough away from the fitting data (indeed rare because they are at high energy, like atoms coming too close together, but not necessarily associated with real saddle points into new low energy basins) that the energy will become anomalously low and the dynamics will go in an unphysical, and out of the fit, direction.

2. How is it possible that turning the noise magnitude into a categorical feature "reduces abrupt changes"? If it works it works, but categorical variables are inherently abrupt, while the underlying noise magnitude is a continuous and smooth quantity.

3. What does "rebuild the original atomic features" mean? The graph features? Some earlier stage in the featurization? As with so much of the method, the authors don't say.

4. Where do the water/ice fitting configurations come from? They say DFT is used to _evaluate_ the forces and energies for each sampled configuration, but how was the sampling done? An interatomic potential? If so, which one? How much decorrelation between samples?

(Remarks on code availability)
I'm unable to access web sites in China, so I cannot review the code or data

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The authors have addressed my concerns.

(Remarks on code availability)
The code base has substantially improved, and now offers guidance on how to use the method properly.

Reviewer #2

(Remarks to the Author)
The authors have done a much better job at describing the method, addressing nearly all of my earlier comments. The github page is accessible to me, and looks reasonable, with a couple of little comments below. I recommend accepting the article for publication once these are addressed.

My point about rare events was that in the introduction (line 54-56 in the revised draft), the authors claim rare events as the source failure of MLIPs ("Consequently...MD simulation collapse"). But rare events are neither necessary nor sufficient for such failures. The failures can happen just because it's impossible to sample everywhere, and it's easy to make a potential that isn't smooth enough to extrapolate to every relevant point in many-dimensional configuration space. The explanation of the source of failures in this paragraph needs to be revised.

The description in Sec 2 intro is, specifically of a message passing graph NN MLIP. This is not a description of all MLIPs - lots of other methods are also MLIPs, e.g. NNs of other descriptors (Behler-Parrinello), linear models (ACE, SNAP), Gaussian process regression (GAP, FLARE). That's fine, the authors can choose to work with graph NNs, but the description should be clear about this.

github page: README says "Demo.ipynb", but file appears to be "demo.ipynb"

(Remarks on code availability)
The code is reasonably accessible, but documentation could be better.

1. docstrings for, at least, top level functions that are meant to be called by the user.

2. typos in the demo.ipynb, e.g. "python train_new.py" but the script is actually "train.py" according to the text above.

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The code base provided does not reproduce the results in the paper both for testing and training and furthermore contains several bugs:

Testing:

- Running 'python test.py --config config.yaml' (using checkpoint：'./checkpoint/water_taip.pt', dataset:'./processed/newliquid_shifted_ev.pt' as provided) does not work, since the checkpoint was produced with dimensions 43 in the decoder, but TAIP/TAIP_test.py generates dimensions 17. After fixing the dimensions on line 67 and 251 in TAIP/TAIP_test.py, the model loads.

- The code still does not run due to errors in the naming of the dataset (namely an inconsistent mixture of test_dataset/test_loader and train_dataset/train_loader). After fixing these in test.py, the script runs through.

- However, the online repo suggests the energy MAE should be 0.2, and the force MACE 0.04. Yet the script returns an energy MAE of 34652(!) and a force MAE of 1.1(!), thus produces totally unusable predictions. Either the provided model is wrong, or the provided code.

Training:

- Running 'python train.py --config config.yaml ' (using dataset:'./processed/newliquid_shifted_ev.pt' as provided) produces some output, which, however, unfortunately is entirely unusable, since on line 253 in train.py, the target forces of all atoms in the batch are set to the target forces of the first atom(!). I.e., all target forces are entirely wrong. Removing the line allows the training to happen on the correct target forces. I am therefore convinced that the presented results were not generated with the provided code.

- The variable P, which seems to be lambda in the manuscript (the scaling between the energy and force loss) is set to 1000 in the script, while the manuscript reports 100. Running the training with P=100 leads to a very poor convergence of the force MAE (about 0.4 after 50 epochs), so I guess the 100 is wrong. Running with P=1000 gives a better force MAE (about 0.04 after 50 epochs). The run with P=100 became furthermore unstable after 51 epochs, producing NaNs in the backward pass.

- However, in both cases the energy MAE does not converge, but jumps wildly between 1 to 30 even for late epochs for both values of P, and is thus not come close to the reported values.

-The periodic boundary conditions are hard-coded into many of the parts of the architecture, so running any of the non-periodic testcases is impossible without large changes to the code in TAIP/PaiNN.py, and train.py. I was able to run it after several changes, but wonder whether this code was actually ever used to run any of the non-periodic benchmarks.

- It is impossible to test or train baselines. The demo suggests to comment out lines 190-194 (no file mentioned) - I looked through all files and in none of them lines 190-194 would toggle a switch to the baseline model. The code needs to be able to both train and test baseline models, especially since a custom implementation of PaiNN is used.

Data leakage:

- In the code for the initial manuscript, the test set seems to be the same as the validation set (datapoints 1000-1300 for liquid water), which causes data leakage, because the validation set is used to select the best-performing model. In the code for the revised version, the test and validation sets are now distinct, but due to the above issues, I doubt whether this version of the code was actually used for the paper. Further, in the git commit history it is visible that at some point there was a shuffling of the dataset before splitting into train/val/test splits in train.py, but not in test.py, leading to training datapoints being put to the test dataset(!). Now, this only happens in older commits and not the newest ones, so there is a chance that this did not affect the outcomes of the paper, but I wanted to raise it nevertheless.

The reproducibility of the manuscript is therefore currently impossible, and I suggest the authors correct their code, make sure the code reproduces the reported results, and provide working code and instructions for readers to also reproduce these results.

(Remarks on code availability)
See above

Version 3:

Reviewer comments:

Reviewer #1

(Remarks to the Author)
The authors have addressed my concerns and significantly improved the code. I was now able to reproduce the results reported in the paper, and recommend the manuscript to be published.

(Remarks on code availability)
The code now reproduces the results reported in the paper and contains sufficient instructions to follow.

Reviewer #1 (Remarks to the Author):

The study by Cui et al. reports the integration of test-time adaption for interatomic potentials. It reports a novel architecture combining existing frameworks with self-supervised tasks, which can be fine-tuned to the distribution of test data during inference without collecting new data. I was pleased to see the authors included both invariant as well as equivariant architectures for the energy and force predictor, and that their framework can significantly improve model performance for both cases. I think this development is very exciting, and succinctly different from other current approaches such as active learning. I therefore recommend to publish the article in Nature Communications after addressing the following (minor) issues:

Response:

We sincerely appreciate your recognition and valuable suggestions for our work. In the following, we will make a point-to-point response to the comments.

1. The methodology needs to be described in more detail, either in the main article or in the Supporting Information. As is, it is for example not clear how the molecular dynamics experiment was run (some software such as OpenMM, LAMMPS, etc, or an implementation by the authors).

Response:

Thank you for the constructive comments. We have made substantial modifications to the methodology of TAIP in the revised manuscript, trying to make everything clear and reproducible. Specifically, we have added Section 4.2 *Model implementation*, which introduces the details about encoder and self-supervised learning tasks. Regarding the molecular dynamics (MD) experiments, all the simulations were performed using the Atomic Simulation Environment (ASE) Python library, and the details are added in the revised Section 4.4 *MD simulations*.

Revise:

In the revised manuscript, we have reorganized most of the content in Section 2 *Results* and Section 4 *Methods*.

2. Another example: It is not clear whether the fine-tuning is permanent, i.e. changes the model when processing each data point, or whether the fine-tuning is only done for the current test data point, and the next test data point is fine-tuned

from the same starting point. If the fine-tuning is permanent, how would the order of test data affect the performance, and is there any guidance on best practices? If the fine-tuning is temporary, this should be clearly stated.

Response:

Thank you for the constructive comments. Regarding the experiments of mean absolute errors (MAEs) on the fixed test sets, i.e., those in Table 1 and Table 2, the fine-tuning is NOT permanent. Specifically, the fine-tuning is only done for the current test data point, and the fine-tuned parameters are not saved. All the test data point is fine-tuned from the same starting point. For these datasets, the test set is constructed with randomly sampled test data, and the order of test data cannot be properly defined. Therefore, we did not test the permanent strategy for these datasets.

During MD simulations, the test sample is generated sequentially. We tested the effect of both permanent and non-permanent strategies. Regarding the non-permanent strategy, i.e., the fine-tuned parameters are not saved, we observed that the stability of MD simulations constantly outperforms the baselines. We further examined the influence of different strategies for the TAIP method. For more detailed discussions, please refer to our response to the next comment.

Revise:

We have added the detailed strategy for test-time adaptation in the revised Section 4.3 *Experimental settings*.

3. Or, as another example among others, it is not clear whether the fine-tuning (if permanent, else disregard this point entirely) was also part of the molecular dynamics trajectory (i.e. the interatomic potential changed throughout the trajectory), or whether the fine-tuning stopped at some point. Along the same line, would the authors recommend to stop the fine-tuning during a production run to compute properties from an MD trajectory (and can one even turn off the fine-tuning with their architecture)? Fully exploring the effect of a changing potential throughout an MD run is beyond the current study, but it would be helpful to provide some comments on best practices.

Response:

Continuing with our response to the previous comment. In addition to the previously mentioned permanent and non-permanent strategies, we also explored other more sophisticated strategies, including adapting the model permanently with an interval of 100 simulation steps and saving the adaptation only if the loss is higher than a certain threshold. The results are shown in Supplementary Figure S4 or Figure R1 as

follows, where we have presented five individual simulations on liquid water using SchNet-TAIP models from randomly selected initial configurations. We found that the overall stability of MD simulations does not improve as we change to other adaptation strategies rather than using the non-permanent adaptation. We have also tested simulations without any adaptations, resulting in worse performances in simulation stabilities. Since the stability of an MD simulation can be affected by many random factors, indeed, it is hard to fully explore the effect of a changing potential on MD simulations in the current study. However, we would suggest using the non-permanent strategy during the production runs as this strategy constantly outperforms the baselines and yields satisfactory performance in our experiments.

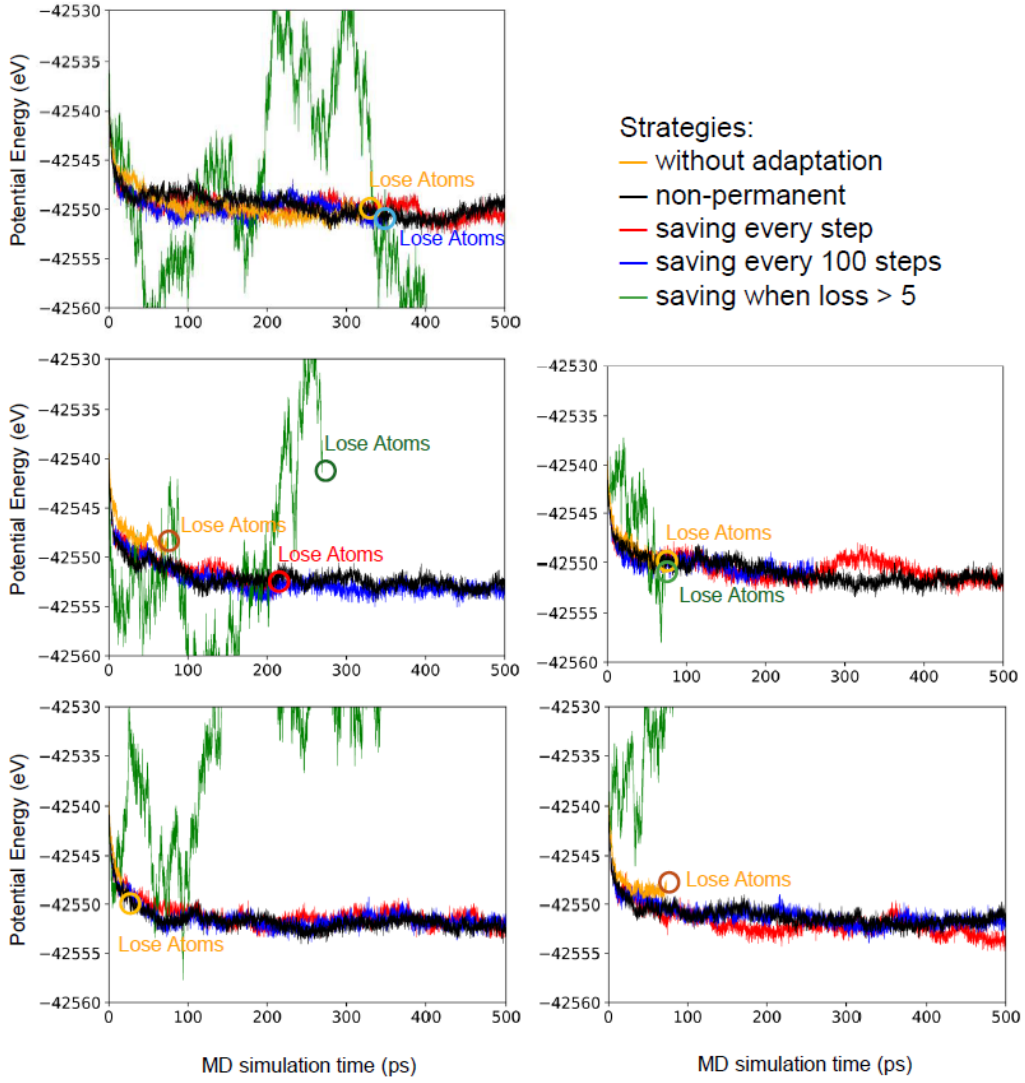


Figure R1. MD simulations with different test-time adaptation strategies. We implement different adaptation strategies for the MD simulations, including simulate without any adaptation, with non-saving adaptation, with saved adaptation every simulation step, with saved adaptation every 100 simulation steps, and with saved adaptation when the loss of the simulation step given by self-supervised learning tasks is larger than a threshold (here, we define it to be 5, which is approximately the average loss in the liquid water test dataset). The experiments are conducted on

liquid water using SchNet-TAIP model at the temperature of 300K with five randomly selected initial configurations. We can observe that the simulations with the simplest non-saving adaptation strategy are quite stable.

Revise:

We have added the detailed strategy for test-time adaptation in the revised Section 4.3 *Experimental settings*. We have added Supplementary Figure S4 to show the impact of different strategies on MD simulations.

4. The content of the SI should be mentioned where appropriate. For example, the main article describes an MD simulation only with SchNet, but I then discovered data for PaiNN in the Supplementary Information. There should be a pointer in the text, e.g. that analogous simulations for PaiNN are described in the SI. Similarly, the ablation studies in the SI should definitely be mentioned somewhere in the text, as I believe they are very important to highlight that all three supervised tasks as proposed in the TAIP framework are actually beneficial. For all results in the SI, their existence should be mentioned in the main article.

Response:

Thank you for the constructive comments. All the contents in the Supplementary Information are now mentioned in the revised main article.

Revise:

We have added analogous simulations with PaiNN to the revised Supplementary Figure S3, and this is mentioned in the revised main text on page 8: "We have also performed MD simulations using the PaiNN baseline and PaiNN-TAIP models, where the results of the simulations are provided in Supplementary Figure S3."

We have mentioned the ablation studies in the revised main text on page 10: "The proposed self-supervised learning tasks are complementary for the improvement of performance, as indicated in the ablation studies in Supplementary Table S5."

5. I would suggest to also study the performance of TAIP vs SchNet and PaiNN as a function of the training size. I.e. for one of the datasets, train on differently sized random subsets of the full training data (evaluated on the same test data) and report the performances individually. This should be very quick to do, and I expect that TAIP will lead to a performance increase in the low data regime, too. This would be a very valuable information in the current quest toward low-data machine learning potentials.

Response:

Thank you for the constructive suggestion. We have tested the performance of TAIP on the different training subsets of ISO17. As expected, the mean absolute errors (MAEs) are decreased with increasing amount of training data, and TAIP leads to a performance increase in all the settings. The results are summarized in Table R1 and added in the revised Supplementary Table S4.

		SchNet	SchNet-TAIP	PaiNN	PaiNN-TAIP
1%	Energy	97.10	**39.51**	88.74	**22.31**
	Force	4.52	**3.93**	3.39	**2.73**
5%	Energy	23.83	**13.22**	18.34	**11.34**
	Force	3.33	**2.89**	2.46	**1.68**
30%	Energy	11.18	**4.99**	3.93	**1.32**
	Force	2.60	**2.07**	1.29	**0.87**
100%	Energy	2.40	**1.13**	0.92	**0.66**
	Force	2.18	**1.78**	0.89	**0.68**

Table R1. Performance of TAIP trained on varying amounts of data from ISO17 dataset. We randomly sampled 1%, 5%, and 30% of the training data from ISO17 dataset to evaluate the performance of TAIP on the same test set. Energy and force MAEs are reported in units of kcal/mol and kcal/mol/Å, respectively. Results of TAIP-based models are shown in bold. It can be noted that TAIP leads to a more obvious performance increase in the low data regime.

Revise:

We have added Supplementary Table S4 and mentioned it in the revised main text on page 7: "We have also studied the performance of TAIP as a function of the training size on the ISO17 dataset. The results are shown in Supplementary Table S4, where the MAEs are decreased with increasing amount of training data, and TAIP leads to a performance increase in all the settings."

6. See the code review below for details. I find that the code repository is currently not fit for distribution (yet), and (in its current state) will not allow the community to adopt this approach. I would sincerely recommend to re-work it, including usage instructions and examples.

Reviewer #1 (Remarks on code availability):

The provided code does not contain any instructions on how to use it, or how to reproduce the study (e.g. a README file, documentation, etc). The code itself seems like it might work, but it includes several hard-coded paths to the personal hard-drives of some of the authors, so it is not usable without modifications. Furthermore,

many of the relevant functions do not contain any descriptions or doc-strings, so that re-using the code in a different project seems difficult, especially given that there is no README file to guide the user. For a paper that introduces a new architecture and probably wants the community to utilize and build on their methods, the code repository is in a very poor state, and needs to be reworked before publication. Since the study uses open-source datasets, I would furthermore suggest to include a full example of training and testing that a user can follow along and reproduce.

Response:

Thank you for the valuable comments. We have reorganized the code of TAIP and uploaded it to GitHub: https://github.com/TaoyongCui/TAIP-codes/tree/main .

We have provided a README file that includes system requirements, an installation guide, and instructions to run this code.

For the convenience of readers to review and use our code, we have provided a Jupyter Notebook file "Demo.ipynb" that provides an example to reproduce the results in this work.

The datasets of water/ice and electrolyte solutions, along with the trajectories and log files of the simulation results referenced in this work, are uploaded to Figshare: https://figshare.com/s/91b8033509412ced0e88 .

Reviewer #2 (Remarks to the Author):

The authors present a self-supervised approach for improving the accuracy and stability of machine learning interatomic potentials after the initial fitting stage, without the need for additional reference data. The results are moderately encouraging - the errors are reduced modestly, and the stability of MD simulation is much better.

Response:

We sincerely appreciate your recognition of our work. In all the experimental settings, mean absolute error (MAE) reduction is well beyond statistical uncertainty. For simple systems such as MD17, it is relatively easy for the model to achieve good performance, while the key challenge lies in complex systems, such as periodic water and electrolytes. In complex and out-of-distribution cases, TAIP results in Force MAEs 21% - 42% lower than baseline. Therefore, we believe that the improvement brought by TAIP is remarkable. Moreover, since interatomic potentials are used to perform MD simulations, the performance related to MD simulations, such as stability, is also an important metric to evaluate TAIP.

However, as is the habit of machine learning papers, the method is completely inadequately described. The standard for scientific literature, which is that readers should be able to reproduce the results, is not even vaguely close to being satisfied. Unfortunately, this is entirely par for the course in the ML literature, but it is also entirely unacceptable. If the authors fully explain their method, the paper might be suitable for publication. Without that, it's simply not a scientific publication. There are many, many places where crucial information is missing. One essential problem is that there is not a step by step explanation of what *exactly* happens at the initial training, test-time adaptation, and final inference phase. For example, the only place where the authors say explicitly that only the decoder is re-fit during the adaption is in the introduction, Sec 1.

Second, the only parameters that are explicitly given are the sizes of the datasets, and, in the supplementary, the hyperparameters for the MLIP fitting.

Response:

We gratefully thank you for the valuable comments. We acknowledge that our previous presentation on the method is not sufficient. In the revised manuscript, we have made a thorough modification and tried our best to make everything clear and reproducible. We have also checked that all the parameters involved in this work are properly defined and provided to ensure our findings are reproducible.

Revise:

In the revised manuscript, we have reorganized most of the content in Section 2 *Results*, and Section 4 *Methods*.

Regarding the three stages in TAIP, more details have been added in the revised Section 2.2 *TAIP framework* (We also added a Section 2.1 *Preliminaries* to introduce basic knowledge about MLIPs).

We have updated Figure 1 to illustrate the TAIP framework.

We have added Section 4.2 *Model implementation* and Section 4.3 *Experimental settings* to explain model details and how we train and evaluate the model.

What about the parameters of the auxiliary loss functions (magnitude of noise L, feature recovery exponent gamma, etc.)?

Response:

Sorry for the omissions. The magnitude of noise, denoted by t in our revised manuscript, is sampled from a uniform distribution and used to generate Gaussian noises. For each configuration with n atoms and $n\times3$ coordinates, we first randomly choose a noise intensity t and generate Gaussian noises with a mean of zero and a

variance of *t*^*2* to perturb the coordinates. Then, we train a classifier, using the combined noisy and noiseless graph feature to produce the noise intensity *t*. The noise intensity label is represented by a one-hot vector, i.e., the intensity of *t* corresponds to a one-hot vector where the *t*^th element is 1 while others are zero.

The gamma in the atom feature recovery task is a hyperparameter used for scaling the cosine error. Introducing a scaling factor gamma>1 in the loss function can down-weight easy sample's contribution in training.

Revise:

The meaning of noisy intensity and exponent gamma are explained in the revised Section 4.2 *Model implementation*, and their values are added in the revised Supplementary Table S1-S3.

What about the architectures of the auxiliary loss decoders? Reading this paper, I have *no idea* how to reproduce what the authors did.

Response:

Sorry for the omissions. In the *atom feature recovery* task, we have designed a *decoder 1* that takes the masked atom features (whose definition is given in the revised Section 4.2 *Model implementation*) as input and produces the atom type of the masked atoms. The atom types are represented by one-hot vectors. The architecture of decoder 1 is a Multi-Layer Perceptron (MLP) with a Sigmoid Linear Unit (SiLU) activation function in between the layers.

In the *pseudo force recovery* task, we have designed a *decoder 2* that takes the masked pseudo forces (whose definition is given in the revised Section 4.2 *Model implementation*) as input and produces the original pseudo forces. The architecture of decoder 2 is inspired by the continuous filter convolution layer from SchNet, and the formula is given in the revised Equation 12.

Revise:

The architectures of decoder 1 and decoder 2 are added in the revised Section 4.2 *Model implementation*.

Other issues

1. MD with MLIPs breaks down even without formal rare events (as that term is usually defined, e.g. transitions over moderate energy barriers into new configuration space) or phase transitions. The lack of sufficient smoothness in nearly all MLIPs

(which leads to poor extrapolation) is enough that MD will almost invitably explore configurations far enough away from the fitting data (indeed rare because they are at high energy, like atoms coming too close together, but not necessarily associated with real saddle points into new low energy basins) that the energy will become anomalously low and the dynamics will go in an unphysical, and out of the fit, direction.

Response:

We agree with you that the break-down issue is almost inevitable for MLIP-based MD simulations, and this is why we propose TAIP, trying to enable more stable MD simulations.

On the other hand, there have been many examples of scientific discovery using MLIP-based MD simulations, for example: *Nature* **589**, 59–64 (2021) and *Science* **371**, 921-925 (2021). Based on past research and our experience, we find that the simulations can remain stable as long as the quality (amount, diversity, etc.) of the training set is good enough.

However, we often need to run MD simulations without enough training data, such as during the sampling process (arxiv:2407.13994). In these scenarios, methods (such as TAIP) that do not rely on additional data to improve the stability of simulations are crucial.

2. How is it possible that turning the noise magnitude into a categorical feature "reduces abrupt changes"? If it works it works, but categorical variables are inherently abrupt, while the underlying noise magnitude is a continuous and smooth quantity.

Response:

Sorry for making you confused. In our view, while noise magnitude is inherently a continuous and smooth variable, for certain complex data distributions, it can be challenging for the model to directly predict this magnitude, which hinders the learning of an effective representation space. Introducing categorical variables can simplify the task of predicting noise intensity, thereby helping our framework more easily learn a relatively smooth representation space.

Revise:

We have modified the description of the noise intensity prediction task in the revised main text on page 4:

"A noisy configuration with a smaller noise intensity is considered closer to the clean one, so that the noise intensity reflects the dissimilarity between noisy and clean data. Instead of directly predicting the precise value of noise, we categorize the intensity

into several bins and determine the bin to which the combined graph feature corresponds. By predicting the noise intensity, the model tries to group graph features of configurations with similar noise intensities, thereby smoothing the configuration representation space."

3. What does "rebuild the original atomic features" mean? The graph features? Some earlier stage in the featurization? As with so much of the method, the authors don't say.

Response:

Sorry for the omissions. The "original atomic feature" refers to the one-hot encoding of the atomic type.

Revise:

The detailed calculation processes regarding the task of atom feature recovery, as well as all other tasks, are provided in the revised Section 4 *Methods*.

4. Where do the water/ice fitting configurations come from? They say DFT is used to *evaluate* the forces and energies for each sampled configuration, but how was the sampling done? An interatomic potential? If so, which one? How much decorrelation between samples?

Response:

Sorry for the omissions. The water and ice configurations come from classical MD simulations using SPC/E force field with a time step of 1 fs. The configurations are sampled every 10 ps, which should be large enough to ensure independence between samples.

Revise:

The detailed description of the water/ice dataset is provided in the revised Section 4.1 *Dataset*.

Reviewer #2 (Remarks on code availability):

I'm unable to access web sites in China, so I cannot review the code or data

Response:

We apologize for not checking the website's accessibility abroad. We have uploaded the code to Github: https://github.com/TaoyongCui/TAIP-codes/tree/main .

We have provided a README file that includes system requirements, an installation guide, and instructions to run this code.

For the convenience of readers to review and use our code, we have provided a Jupyter Notebook file "Demo.ipynb" that provides an example to reproduce the results in this work.

The datasets of water/ice and electrolyte solutions, along with the trajectories and log files of the simulation results referenced in this work, are uploaded to Figshare: https://figshare.com/s/91b8033509412ced0e88 .

Reviewer #2 (Remarks to the Author):

The authors have done a much better job at describing the method, addressing nearly all of my earlier comments. The github page is accessible to me, and looks reasonable, with a couple of little comments below. I recommend accepting the article for publication once these are addressed.

My point about rare events was that in the introduction (line 54-56 in the revised draft), the authors claim rare events as the source failure of MLIPs ("Consequently...MD simulation collapse"). But rare events are neither necessary nor sufficient for such failures. The failures can happen just because it's impossible to sample everywhere, and it's easy to make a potential that isn't smooth enough to extrapolate to every relevant point in many-dimensional configuration space. The explanation of the source of failures in this paragraph needs to be revised.

Response:

Thank you for providing further clarification on this issue. The original statement regarding rare events is modified as:

“However, in fields such as phase transition, catalysis, and crystal growth, the configurational space that needs to be explored is highly complex. This complexity makes it challenging to sample sufficient data for training and easy to make a potential that is not smooth enough to extrapolate to every relevant point.”

The description in Sec 2 intro is, specifically of a message passing graph NN MLIP. This is not a description of all MLIPs - lots of other methods are also MLIPs, e.g. NNs of other descriptors (Behler-Parrinello), linear models (ACE, SNAP), Gaussian process regression (GAP, FLARE). That's fine, the authors can choose to work with graph NNs, but the description should be clear about this.

Response:

Thank you for pointing this out. We have added descriptions of other methods in Section 2.1 Preliminaries:

“Various model architectures have been developed, including descriptor-based methods with neural networks, linear models, and Gaussian process regressions. Recently, graph neural networks have emerged as a particularly powerful architecture for MLIPs.”

github page: README says "Demo.ipynb", but file appears to be "demo.ipynb"

Response:

Thank you for your feedback. We apologize for the inconsistency in the README file. The correct filename is indeed "demo.ipynb" (with a lowercase 'd'). We have updated the README to accurately reflect this information. You can find our code repository at https://github.com/TaoyongCui/TAIP-codes.

Reviewer #2 (Remarks on code availability):

The code is reasonably accessible, but documentation could be better.

1. docstrings for, at least, top level functions that are meant to be called by the user.

Response:

Thank you for your feedback regarding the documentation. We have added comprehensive docstrings for all top-level functions intended for user interaction.

2. typos in the demo.ipynb, e.g. "python train_new.py" but the script is actually "train.py" according to the text above.

Response:

Thank you for pointing out the typographical errors in demo.ipynb. We have corrected the reference to the script name from "python train_new.py" to "python train.py" and ensure that the code repository has no other errors.

Response to Reviewer #1

Reviewer #1 (Remarks to the Author):

The code base provided does not reproduce the results in the paper both for testing and training and furthermore contains several bugs:

Response:

We sincerely apologize for the issues with the uploaded code. The primary cause of these problems was related to the code version. The checkpoint we uploaded was also not the correct version. We have conducted a thorough review and provided the correct version after careful verification. All the experimental results that appeared in our paper are reproducible. In the following, please find our point-to-point response.

Testing:

- Running 'python test.py --config config.yaml' (using checkpoint: './checkpoint/water_taip.pt', dataset:'./processed/newliquid_shifted_ev.pt' as provided) does not work, since the checkpoint was produced with dimensions 43 in the decoder, but TAIP/TAIP_test.py generates dimensions 17. After fixing the dimensions on line 67 and 251 in TAIP/TAIP_test.py, the model loads.

- The code still does not run due to errors in the naming of the dataset (namely an inconsistent mixture of test_dataset/test_loader and train_dataset/train_loader). After fixing these in test.py, the script runs through.

- However, the online repo suggests the energy MAE should be 0.2, and the force MACE 0.04. Yet the script returns an energy MAE of 34652(!) and a force MAE of 1.1(!), thus produces totally unusable predictions. Either the provided model is wrong, or the provided code.

Response:

We have updated the code, checkpoints, and notebooks on GitHub. Running the updated notebooks can now reproduce the results:

- The notebook '*Demo_water_TAIP.ipynb*' includes data preprocessing, PaiNN-TAIP training on the periodic testcases (liquid water dataset), testing on the liquid water/ice test dataset, and corresponding molecular dynamics simulations.

- The notebook '*Demo_water_baseline.ipynb*' includes PaiNN baseline training on the periodic testcases (liquid water dataset), testing on the liquid water/ice

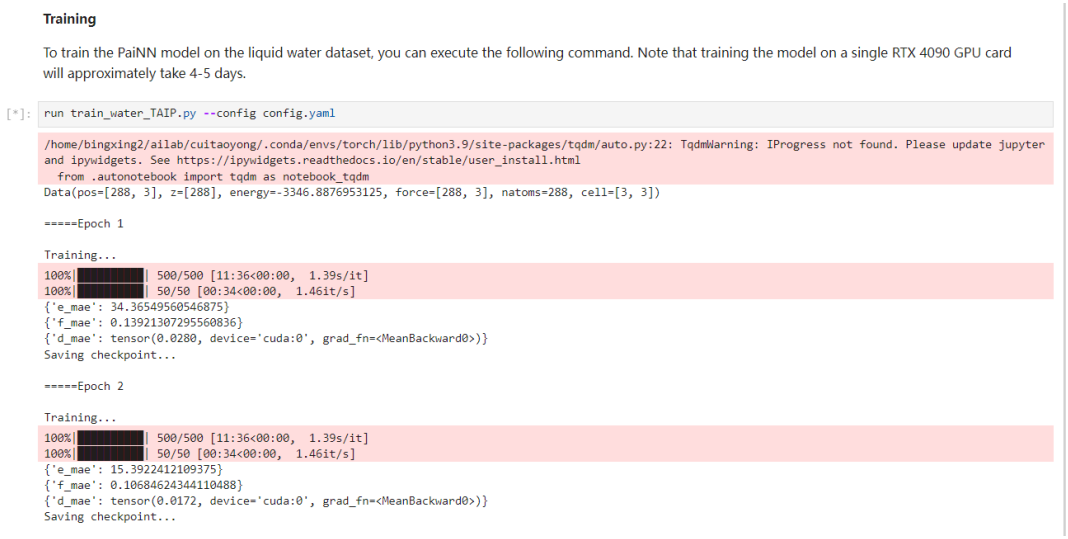
test dataset, and corresponding molecular dynamics simulations.

- The notebook '*Demo_md17_TAIP.ipynb*' includes PaiNN-TAIP training on the non-periodic testcases (aspirin dataset), testing on the aspirin test dataset.

- The notebook '*Demo_md17_baseline.ipynb*' includes PaiNN baseline training on the non-periodic testcases (aspirin dataset), testing on the aspirin test dataset.

- The notebook '*MD_simulations.ipynb*' includes molecular dynamics simulations, as well as comparisons between the PaiNN-TAIP and the PaiNN baseline model.

Figures R1–R3 provide examples of the code execution in the file *'Demo_water_TAIP.ipynb'*, demonstrating both training and testing of PaiNN-TAIP model on the liquid water dataset. Figures R4–R7 show the corresponding Mean Absolute Error (MAE) for each test sample in the files *'./processed/water_test.pt'* and *'./processed/ice_test.pt'*.

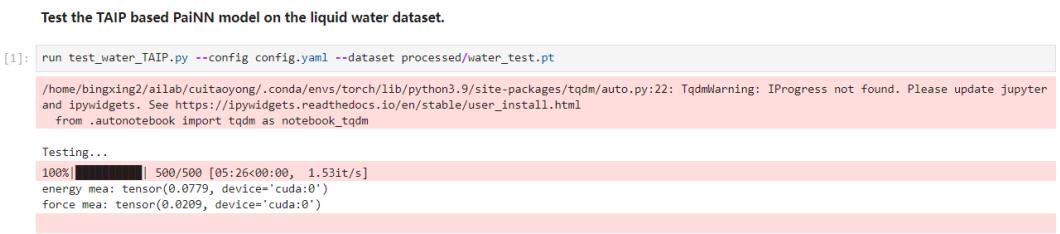


Training

To train the PaiNN model on the liquid water dataset, you can execute the following command. Note that training the model on a single RTX 4090 GPU card will approximately take 4-5 days.

```
[*]: run train_water_TAIP.py --config config.yaml

/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Data(pos=[288, 3], z=[288], energy=-3346.8876953125, force=[288, 3], natoms=288, cell=[3, 3])

=====Epoch 1

Training...
100%|██████████| 500/500 [11:36<00:00,  1.39s/it]
100%|██████████| 50/50 [00:34<00:00,  1.46it/s]
{'e_mae': 34.36549560546875}
{'f_mae': 0.13921307295560836}
{'d_mae': tensor(0.0280, device='cuda:0', grad_fn=<MeanBackward0>)}
Saving checkpoint...

=====Epoch 2

Training...
100%|██████████| 500/500 [11:36<00:00,  1.39s/it]
100%|██████████| 50/50 [00:34<00:00,  1.46it/s]
{'e_mae': 15.3922412109375}
{'f_mae': 0.10684624344110488}
{'d_mae': tensor(0.0172, device='cuda:0', grad_fn=<MeanBackward0>)}
Saving checkpoint...
```

Figure R1: Train the PaiNN-TAIP models on the liquid water training dataset in the file 'Demo_water_TAIP.ipynb'



Test the TAIP based PaiNN model on the liquid water dataset.

```
[1]: run test_water_TAIP.py --config config.yaml --dataset processed/water_test.pt

/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

Testing...
100%|██████████| 500/500 [05:26<00:00,  1.53it/s]
energy mea: tensor(0.0779, device='cuda:0')
force mea: tensor(0.0209, device='cuda:0')
```

Figure R2: Test the PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the liquid water test dataset (file './processed/water_test.pt')

Test the TAIP based PaiNN model on the ice dataset

```
[3]: run test_water_TAIP.py --config config.yaml --dataset processed/ice_test.pt

Testing...
100%|██████████| 500/500 [05:27<00:00,  1.53it/s]
energy mea: tensor(0.0804, device='cuda:0')
force mea: tensor(0.0171, device='cuda:0')
```

Figure R3: Test the PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the ice test dataset (file './processed/ice_test.pt')

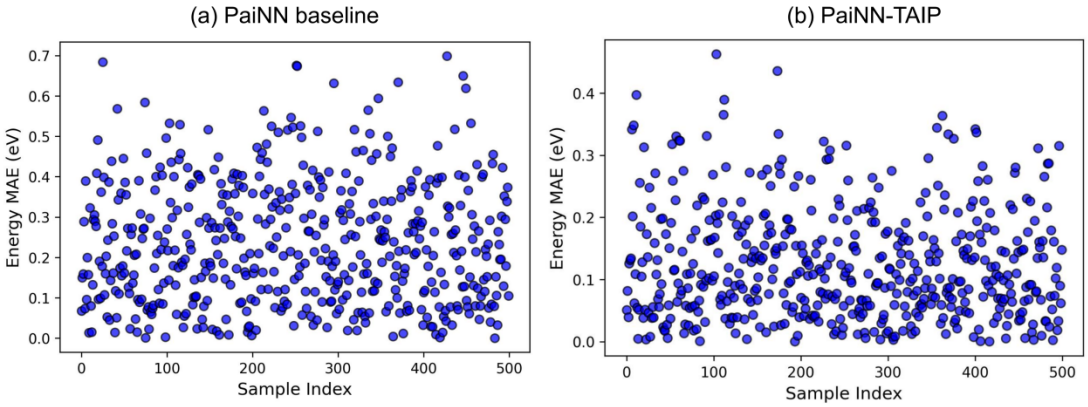


Figure R4: Energy MAE of PaiNN baseline (checkpoint file './checkpoint/water_base.pt') and PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the liquid water test dataset (file './processed/water_test.pt')

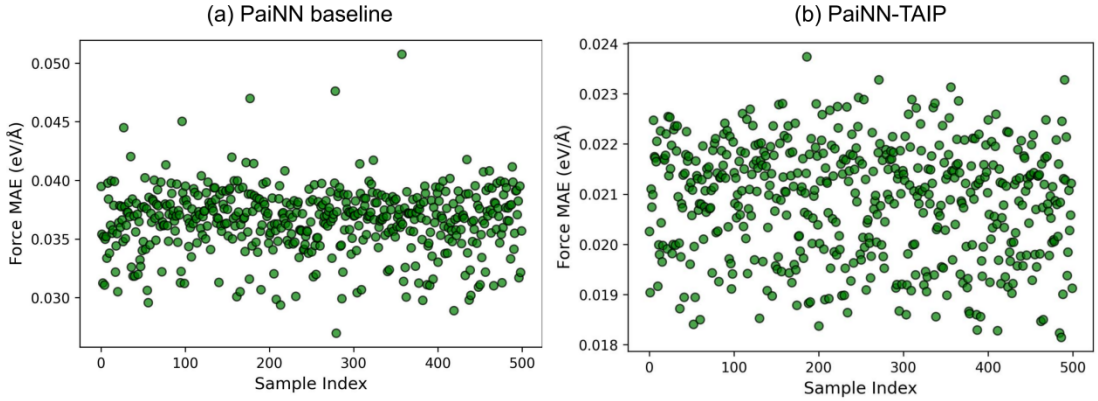


Figure R5: Force MAE of PaiNN baseline (checkpoint file './checkpoint/water_base.pt') and PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the liquid water test dataset (file './processed/water_test.pt')

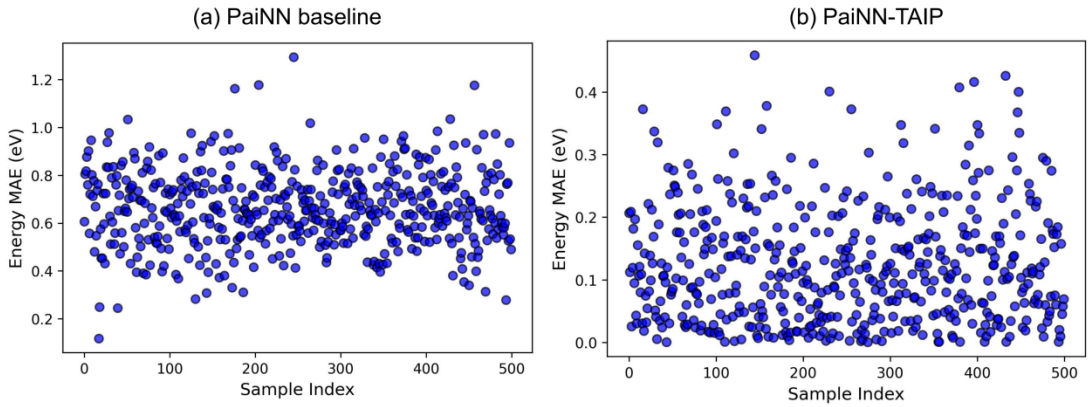


Figure R6: Energy MAE of PaiNN baseline (checkpoint file './checkpoint/water_base.pt') and PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the ice test dataset (file './processed/ice_test.pt')

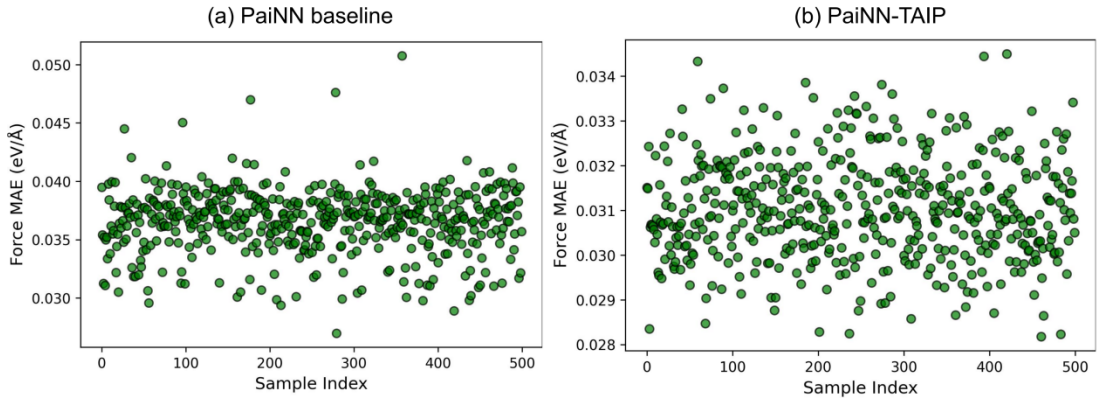


Figure R7: Force MAE of PaiNN baseline (checkpoint file './checkpoint/water_base.pt') and PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the ice test dataset (file './processed/ice_test.pt')

Training:

- Running 'python train.py --config config.yaml ' (using dataset:'./processed/newliquid_shifted_ev.pt' as provided) produces some output, which, however, unfortunately is entirely unusable, since on line 253 in train.py, the target forces of all atoms in the batch are set to the target forces of the first atom(!). I.e., all target forces are entirely wrong. Removing the line allows the training to happen on the correct target forces. I am therefore convinced that the presented results were not generated with the provided code.

Response:

The version of our data processing code was incorrect, leading to the force data including an additional dimension, [1, N, 3]. Consequently, force[0] contained the forces for all atoms. However, this did not match the uploaded dataset, rendering the

code unusable. We have since replaced it with the correct version.

- The variable P, which seems to be lambda in the manuscript (the scaling between the energy and force loss) is set to 1000 in the script, while the manuscript reports 100. Running the training with P=100 leads to a very poor convergence of the force MAE (about 0.4 after 50 epochs), so I guess the 100 is wrong. Running with P=1000 gives a better force MAE (about 0.04 after 50 epochs). The run with P=100 became furthermore unstable after 51 epochs, producing NaNs in the backward pass.

Response:

The variable P corresponds to lambda in the manuscript. We sincerely apologize for mistakenly writing P=100 instead of P=1000 and we have corrected it. In Table 1 and Table 2 of the revised manuscript, the reported TAIP results are obtained with P=1000, while the baselines are with P=100. For fair comparisons, we have retrained all baseline models with P=1000, and the additional results have been included in the updated tables. All the models are trained independently five times. Our conclusion that TAIP outperforms baseline models remains the same.

- However, in both cases the energy MAE does not converge, but jumps wildly between 1 to 30 even for late epochs for both values of P, and is thus not come close to the reported values.

Response:

It is normal that the energy MAE fluctuates during the early stages of training, but it will converge as training progresses. We saved the checkpoint with the minimum validation loss and use it for testing. To demonstrate the repeatability, we have retrained the model on the liquid water dataset (using the code command in Figure R1), and the loss curves throughout the training process are displayed in Figures R8-R10. The training process can be reproduced using the provided notebook 'Demo_water_TAIP.ipynb', and all log outputs are documented in the file 'water_training.out'.

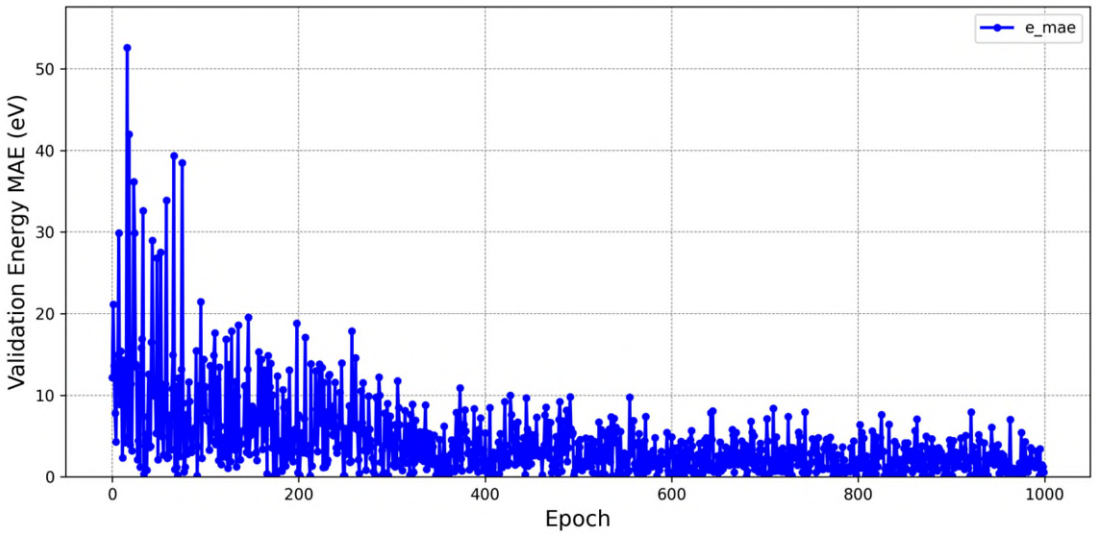


Figure R8: The Validation Energy MAE Curve for the PaiNN-TAIP Model.

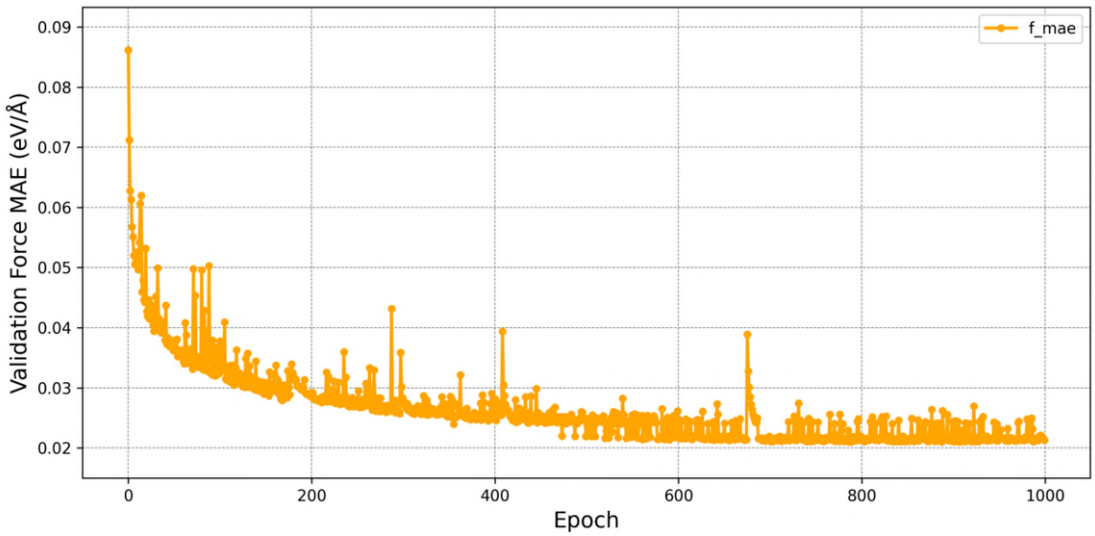


Figure R9: The Validation Force MAE Curve for the PaiNN-TAIP Model.

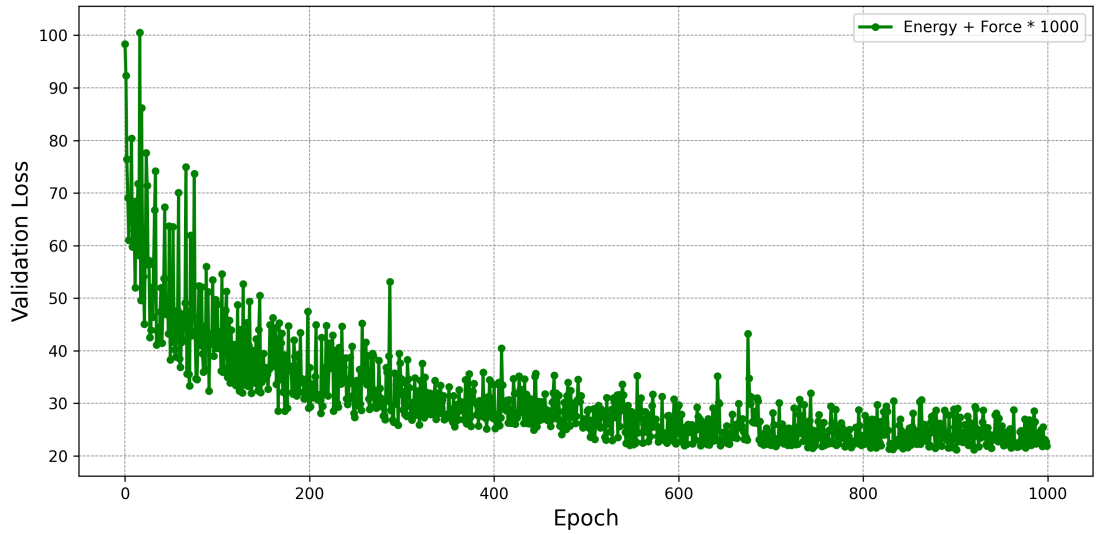


Figure R10: Validation Loss Curve for the PaiNN-TAIP Model.

- The periodic boundary conditions are hard-coded into many of the parts of the architecture, so running any of the non-periodic testcases is impossible without large changes to the code in TAIP/PaiNN.py, and train.py. I was able to run it after several changes, but wonder whether this code was actually ever used to run any of the non-periodic benchmarks.

Response:

To reproduce our results on both periodic and non-periodic benchmarks, we have provided separate codes for each, respectively. For non-periodic benchmarks, the Aspirin testcase can be directly trained and tested using the notebook 'Demo_md17_TAIP.ipynb'. Figures R11-R12 illustrate the training and testing processes within this notebook. Figures R13-R14 illustrate the MAE of the PaiNN baseline (checkpoint file './checkpoint/Baseline_md17_aspirin.pt') and PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_MD17_aspirin.pt') for each test sample point in aspirin test dataset.

Training

To train the PaiNN model on the Aspirin dataset, you can execute the following command. Note that training the model on a single RTX 4090 GPU card will approximately take 4 days.

```
[*]: run train_md17_TAIP.py --config config_md17.yaml
```

```
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

=====Epoch 1

Training...
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/torch_geometric/data/in_memory_dataset.py:300: UserWarning: It is not rec
ommended to directly access the internal storage format `data` of an 'InMemoryDataset'. If you are absolutely certain what you are doing, access the int
ernal storage via `InMemoryDataset._data` instead to suppress this warning. Alternatively, you can access stacked individual attributes of every graph v
ia `dataset.{attr_name}`.
  warnings.warn(msg)
100%|██████████| 500/500 [04:02<00:00,  2.06it/s]
100%|██████████| 32/32 [00:16<00:00,  1.89it/s]
{'e_mae': 2.4033590257167816}
{'f_mae': 5.007408767938614}
{'d_mae': tensor(0.1244, device='cuda:0', grad_fn=<MeanBackward0>)}
Saving checkpoint...

=====Epoch 2

Training...
100%|██████████| 500/500 [03:56<00:00,  2.11it/s]
100%|██████████| 32/32 [00:17<00:00,  1.87it/s]
{'e_mae': 2.076125882565975}
{'f_mae': 4.1111727729439735}
{'d_mae': tensor(0.0358, device='cuda:0', grad_fn=<MeanBackward0>)}
Saving checkpoint...
```

Figure R11: Train the PaiNN-TAIP models on the aspirin dataset in the file 'Demo_md17_TAIP.ipynb'

Test the TAIP based PaiNN model on the Aspirin test dataset.

```
[1]: run test_md17_TAIP.py --config config_md17.yaml
```

```
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

Testing...
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/torch_geometric/data/in_memory_dataset.py:300: UserWarning: It is not rec
ommended to directly access the internal storage format `data` of an 'InMemoryDataset'. If you are absolutely certain what you are doing, access the int
ernal storage via `InMemoryDataset._data` instead to suppress this warning. Alternatively, you can access stacked individual attributes of every graph v
ia `dataset.{attr_name}`.
  warnings.warn(msg)
100%|██████████| 1000/1000 [09:15<00:00,  1.80it/s]
energy mea: tensor(0.1758, device='cuda:1')
force mea: tensor(0.3461, device='cuda:1')
```

Figure R12: Test the PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_MD17_aspirin.pt') on the aspirin test dataset in the file 'Demo_md17_TAIP.ipynb'

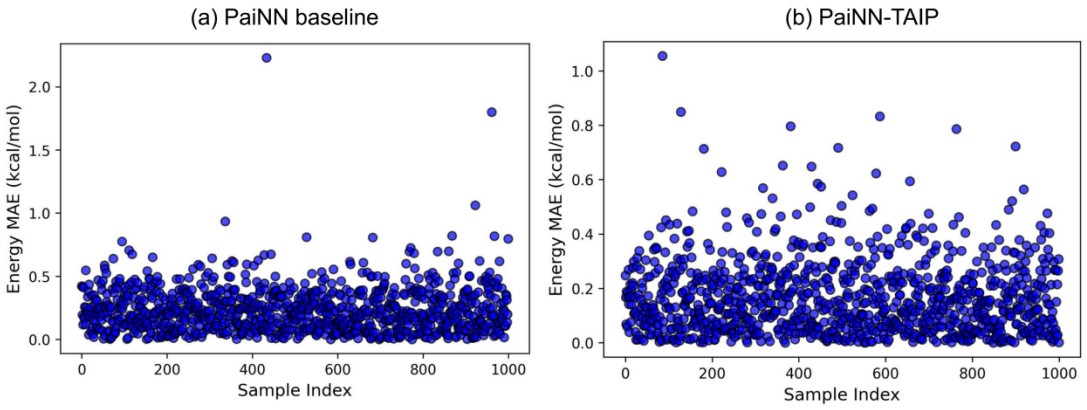


Figure R13: Energy MAE of PaiNN baseline model (checkpoint file './checkpoint/Baseline_md17_aspirin.pt') and PaiNN-TAIP model (checkpoint file

'./checkpoint/TAIP_MD17_aspirin.pt') on the aspirin dataset

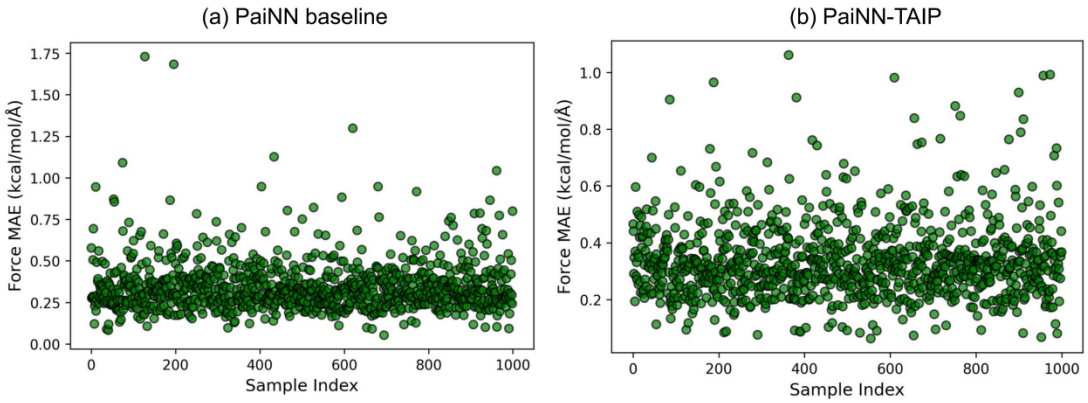


Figure R14: Force MAE of PaiNN baseline model (checkpoint file './checkpoint/Baseline_md17_aspirin.pt') and PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_MD17_aspirin.pt') on the aspirin dataset

- It is impossible to test or train baselines. The demo suggests to comment out lines 190-194 (no file mentioned) - I looked through all files and in none of them lines 190-194 would toggle a switch to the baseline model. The code needs to be able to both train and test baseline models, especially since a custom implementation of PaiNN is used.

Response:

In the previous revision, we added docstrings to the code as per the reviewer's suggestion. However, we overlooked updating the line numbers accordingly. We have provided separate codes for baseline and TAIP, respectively. Now, without any code changes, you can run the baseline models using "Demo_md17_baseline. ipynb" and "Demo_water_baseline.ipynb". Figure R15-R18 illustrates the process of training and testing the PaiNN baseline model on the water and md17 datasets using notebook 'Demo_water_baseline.ipynb' and 'Demo_md17_baseline.ipynb'. All the loss curves throughout the training process are displayed in Figures R19-R24. MAE distributions of baseline models are displayed in Figures R4-R7 (a) and Figures R13-R14 (a).

Training

To train the PaiNN model on the liquid water dataset, you can execute the following command. Note that training the model on a single RTX 4090 GPU card will approximately take 2 days.

```
[*]: run train_water_baseline.py --config config.yaml
```

```
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
Data(pos=[288, 3], z=[288], energy=-3346.8876953125, force=[288, 3], natoms=288, cell=[3, 3])

=====Epoch 1

Training...
100%|██████████| 500/500 [05:16<00:00,  1.58it/s]
100%|██████████| 50/50 [00:21<00:00,  2.29it/s]
{'e_mae': 15.750009765625}
{'f_mae': 0.1359633380174637}
{'d_mae': tensor(0.0479, device='cuda:0', grad_fn=<MeanBackward0>)}
Saving checkpoint...

=====Epoch 2

Training...
100%|██████████| 500/500 [05:21<00:00,  1.55it/s]
100%|██████████| 50/50 [00:22<00:00,  2.27it/s]
{'e_mae': 6.42786865234375}
{'f_mae': 0.09483041971921921}
{'d_mae': tensor(0.0252, device='cuda:0', grad_fn=<MeanBackward0>)}
Saving checkpoint...
```

Figure R15: Train the PaiNN baseline models on the liquid water dataset in the file 'Demo_water_baseline.ipynb'

Test the baseline PaiNN model on the liquid water dataset.

```
[1]: run test_water_baseline.py --config config.yaml --dataset processed/water_test.pt
```

```
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

Testing...
100%|██████████| 500/500 [01:14<00:00,  6.73it/s]
energy mea: tensor(0.1197, device='cuda:0')
force mea: tensor(0.0366, device='cuda:0')
```

Test the baseline PaiNN model on the ice dataset

```
[2]: run test_water_baseline.py --config config.yaml --dataset processed/ice_test.pt
```

```
Testing...
100%|██████████| 500/500 [01:13<00:00,  6.80it/s]
energy mea: tensor(0.5408, device='cuda:0')
force mea: tensor(0.0310, device='cuda:0')
```

Figure R16: Test the PaiNN baseline model (checkpoint file './checkpoint/water_base.pt') on the liquid water/ice test dataset in the file 'Demo_water_baseline.ipynb'

Training

To train the PaiNN model on the Aspirin dataset, you can execute the following command. Note that training the model on a single RTX 4090 GPU card will approximately take 2 days.

```
[*]: run train_md17_baseline.py --config config_md17.yaml
```

```
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

=====Epoch 1

Training...
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/torch_geometric/data/in_memory_dataset.py:300: UserWarning: It is not rec
ommended to directly access the internal storage format `data` of an 'InMemoryDataset'. If you are absolutely certain what you are doing, access the int
ernal storage via `InMemoryDataset._data` instead to suppress this warning. Alternatively, you can access stacked individual attributes of every graph v
ia `dataset.{attr_name}`.
  warnings.warn(msg)
100%|██████████| 500/500 [01:29<00:00,  5.57it/s]
100%|██████████| 500/500 [01:00<00:00,  8.21it/s]
{'e_mae': 13.860221015930176}
{'f_mae': 4.552771507740021}
Saving checkpoint...

=====Epoch 2

Training...
100%|██████████| 500/500 [01:29<00:00,  5.60it/s]
100%|██████████| 500/500 [01:00<00:00,  8.21it/s]
{'e_mae': 1.6906887362599372}
{'f_mae': 3.213884108066559}
Saving checkpoint...
```

Figure R17: Train the PaiNN baseline models on the aspirin dataset in the file 'Demo_md17_baseline.ipynb'

Test the baseline PaiNN model on the Aspirin test dataset.

```
[1]: run test_md17_baseline.py --config config_md17.yaml
```

```
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm

Testing...
/home/bingxing2/ailab/cuitaoyong/.conda/envs/torch/lib/python3.9/site-packages/torch_geometric/data/in_memory_dataset.py:300: UserWarning: It is not rec
ommended to directly access the internal storage format `data` of an 'InMemoryDataset'. If you are absolutely certain what you are doing, access the int
ernal storage via `InMemoryDataset._data` instead to suppress this warning. Alternatively, you can access stacked individual attributes of every graph v
ia `dataset.{attr_name}`.
  warnings.warn(msg)
100%|██████████| 1000/1000 [02:49<00:00,  5.91it/s]
energy mea: tensor(0.2355, device='cuda:0')
force mea: tensor(0.3469, device='cuda:0')
```

Figure R18: Test the PaiNN baseline model (checkpoint file './checkpoint/Baseline_md17_aspirin.pt') on the aspirin test dataset in the file 'Demo_md17_baseline.ipynb'

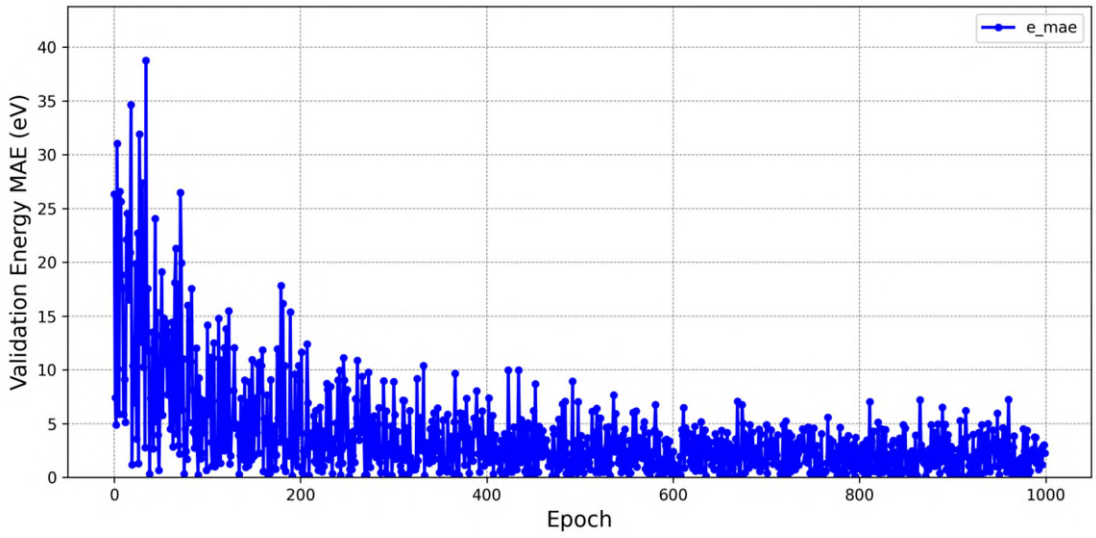


Figure R19: Validation Energy MAE Curve for the PaiNN Baseline Model on the Liquid Water Dataset.

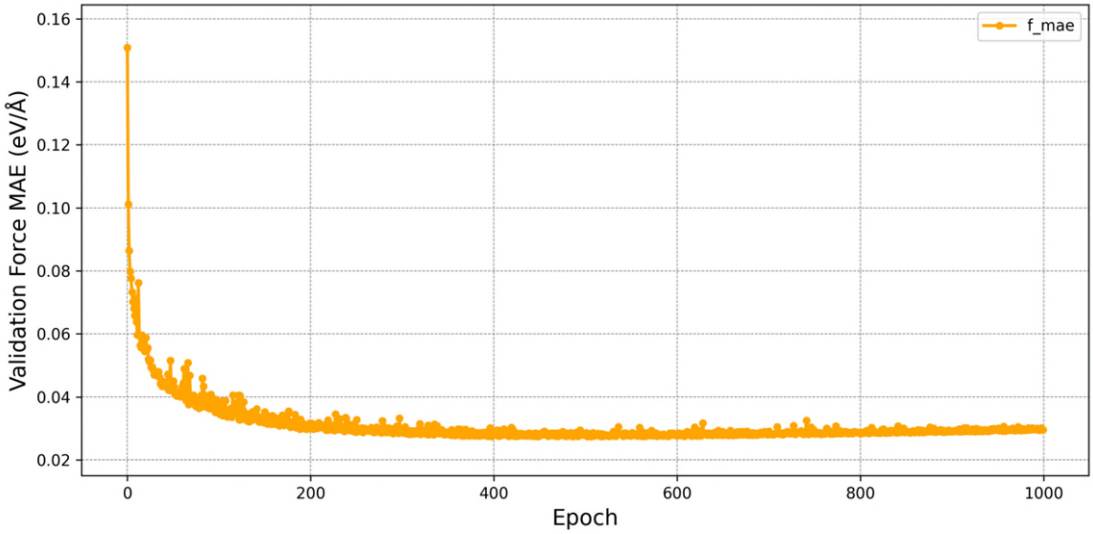


Figure R20: Validation Force MAE Curve for the PaiNN Baseline Model on the Liquid Water Dataset.

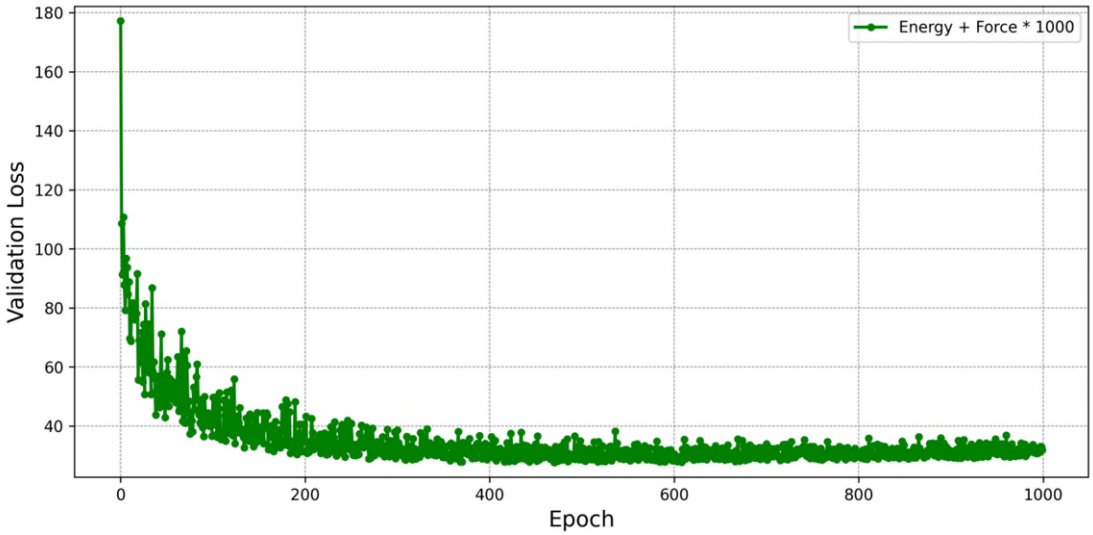


Figure R21: Validation Loss Curve for the PaiNN Baseline Model on the Liquid Water Dataset.

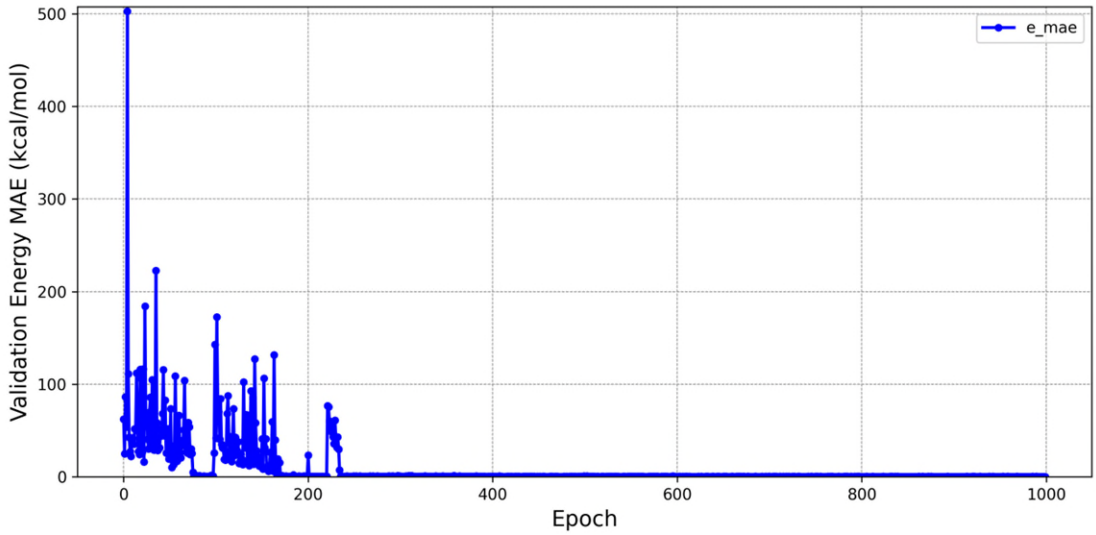


Figure R22: Validation Energy MAE Curve for the PaiNN Baseline Model on the Aspirin Dataset.

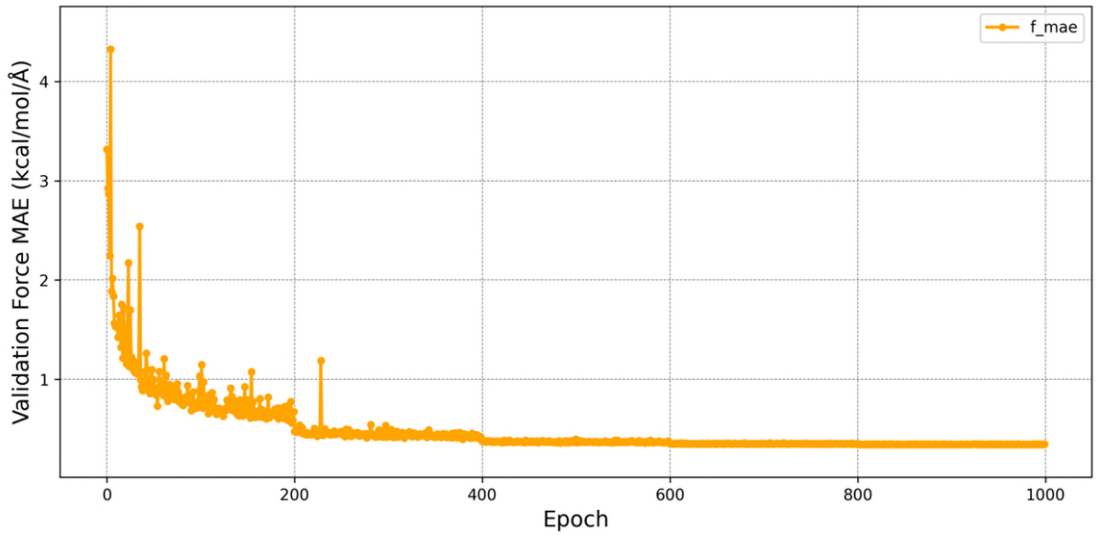


Figure R23: Validation Force MAE Curve for the PaiNN Baseline Model on the Aspirin Dataset.

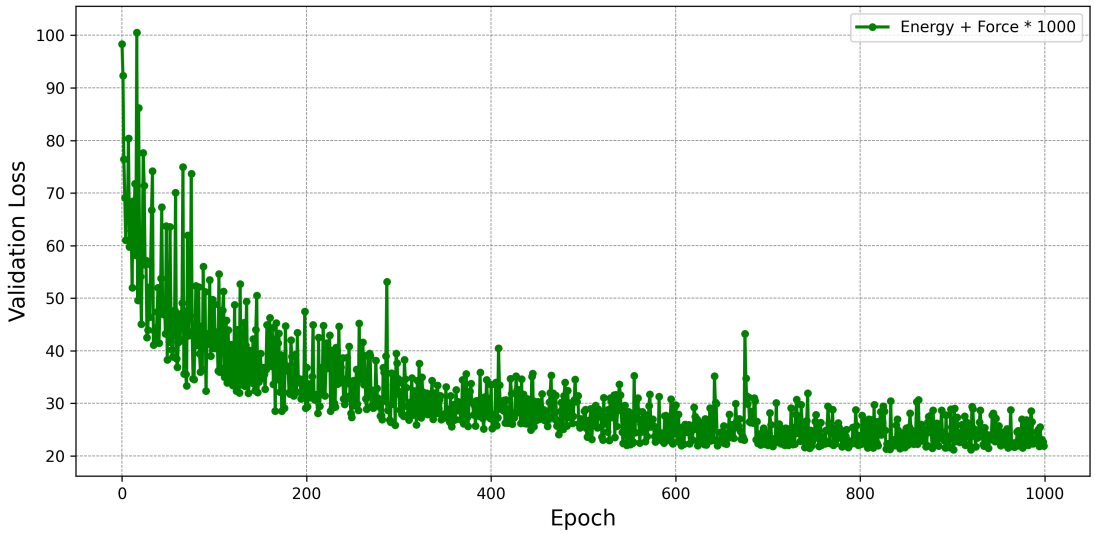


Figure R24: Validation Loss Curve for the PaiNN Baseline Model on the Aspirin Dataset.

Data leakage:

- In the code for the initial manuscript, the test set seems to be the same as the validation set (datapoints 1000-1300 for liquid water), which causes data leakage, because the validation set is used to select the best-performing model. In the code for the revised version, the test and validation sets are now distinct, but due to the above issues, I doubt whether this version of the code was actually used for the paper. Further, in the git commit history it is visible that at some point there was a shuffling of the dataset before splitting into train/val/test splits in train.py, but not in test.py, leading to training datapoints being put to the test dataset(!). Now, this only happens in older commits and not the newest ones, so there is a chance that this did not affect the outcomes of the paper, but I wanted to raise it nevertheless.

Response:

Sorry for the confusion, but we guarantee that there is no data leakage issue in our experiments. The liquid water configurations were sampled every 10 ps from a classical MD trajectory, as described in the manuscript. Then, we performed DFT calculation and collected about 1700 well converged results. In all experiments related to liquid water, we use datapoints [:1000] for training, [1000:1100] for validation, and [1100:1600] for testing, and now the three datasets are uploaded separately. We have also visualized the energy distribution of the validation and test sets, shown in Figure R25-26.

Specifically, for the results in our paper, we trained on the uploaded training dataset, saved the checkpoint with the minimum validation loss, and evaluated on the test set. We have provided a log file (water_training.out) of the training process.

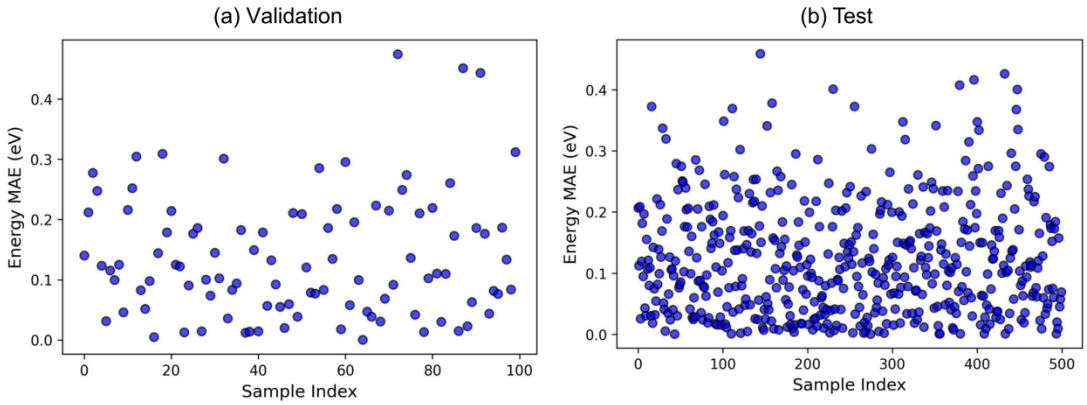


Figure 25: Energy MAE of PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the liquid water validation and test dataset (file './processed/water_valid.pt' and './processed/water_test.pt')

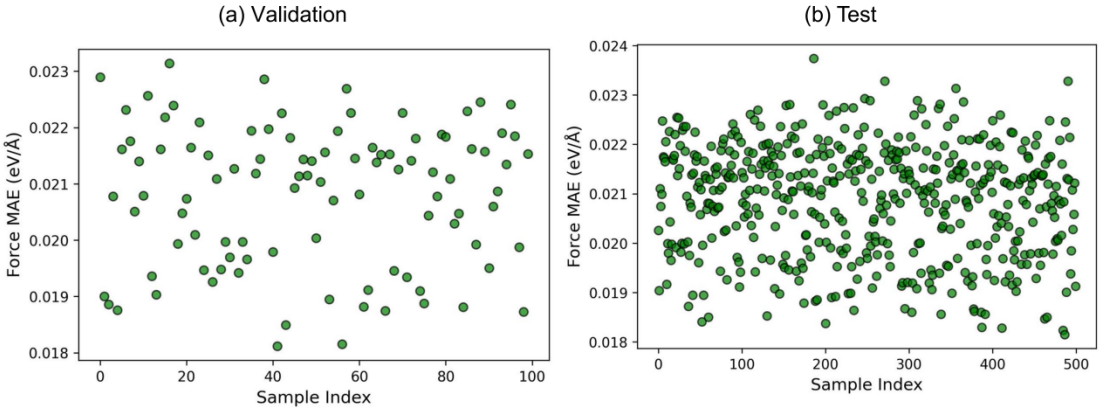


Figure 26: Force MAE of PaiNN-TAIP model (checkpoint file './checkpoint/TAIP_water.pt') on the liquid water validation and test dataset (file './processed/water_valid.pt' and './processed/water_test.pt')

The reproducibility of the manuscript is therefore currently impossible, and I suggest the authors correct their code, make sure the code reproduces the reported results, and provide working code and instructions for readers to also reproduce these results.

Response:

We sincerely apologize again for the issues with reproducing our results. We have updated the code, checkpoints, and notebooks on GitHub, and we believe that all the aforementioned issues are solved. Additionally, we have provided a notebook "MD_simulations.ipynb" for reproducing the MD simulation results.

Figures R27-R28 provide a comparison of simulation stability between TAIP and baseline models. We conducted 20 independent simulations for PaiNN baseline and PaiNN-TAIP. The PaiNN-TAIP simulations were consistently stable over 100,000 steps. For the baseline PaiNN, only two testcases exceeded 10,000 steps, and neither of them reached 100,000 steps due to temperature anomalies.

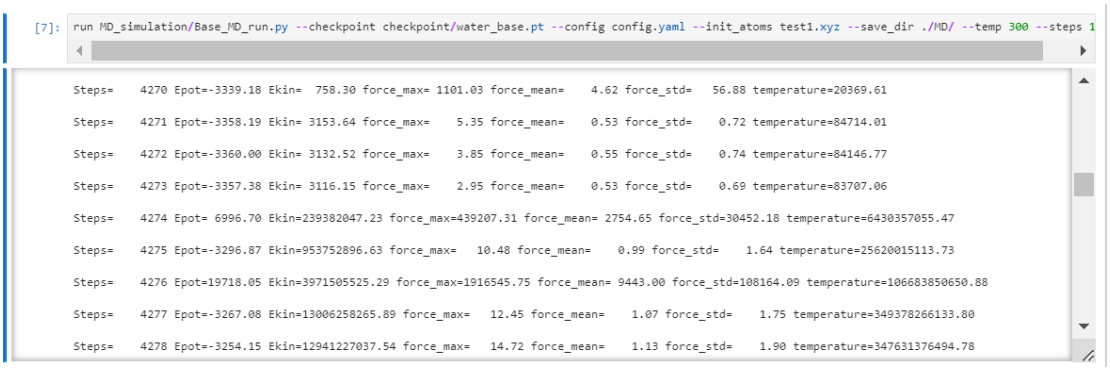


Figure R27(a): Simulation on ice example (test1.xyz) of Baseline PaiNN (checkpoint file './checkpoint/water_base.pt).

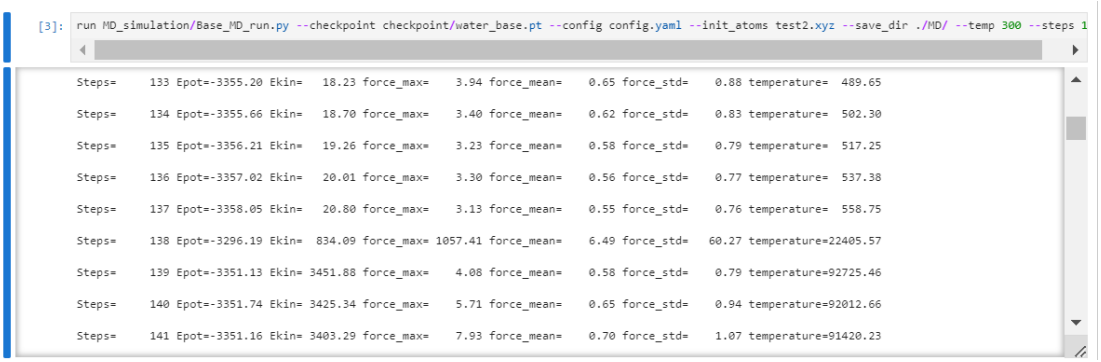


Figure R27(b): Simulation on ice example (test2.xyz) of Baseline PaiNN (checkpoint file './checkpoint/water_base.pt')

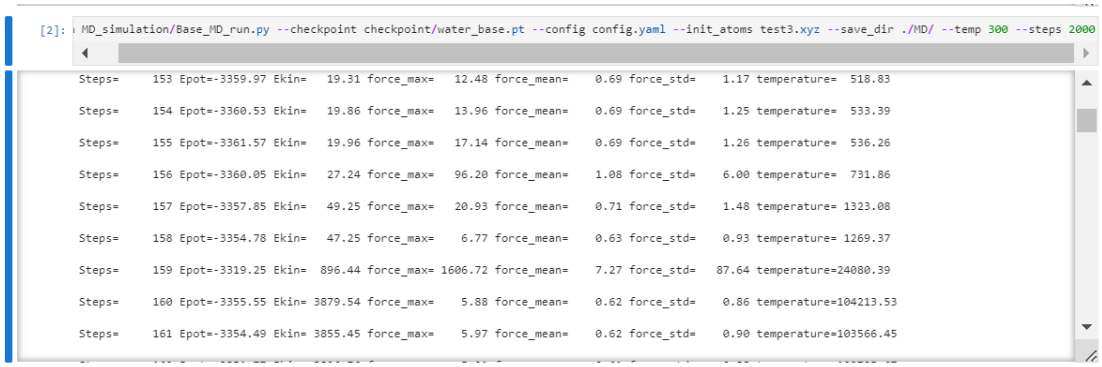


Figure 27(c): Simulation on ice example (test3.xyz) of Baseline PaiNN (checkpoint file './checkpoint/water_base.pt')

```
[ ]: run MD_simulation/Base_MD_run.py --checkpoint checkpoint/water_base.pt --config config.yaml --init_atoms test.xyz --save_dir ./MD/ --temp 300 --steps 10

Steps=    5593 Epot=-3362.00 Ekin=   12.33 force_max=    4.52 force_mean=    0.54 force_std=    0.75 temperature=  331.35
Steps=    5594 Epot=-3362.47 Ekin=   12.75 force_max=    5.65 force_mean=    0.55 force_std=    0.78 temperature=  342.49
Steps=    5595 Epot=-3335.93 Ekin= 1230.96 force_max= 1208.04 force_mean=    5.76 force_std=   71.54 temperature=33066.56
Steps=    5596 Epot=-3358.58 Ekin= 5075.05 force_max=    5.26 force_mean=    0.54 force_std=    0.76 temperature=136327.68
Steps=    5597 Epot=-3354.31 Ekin= 5048.49 force_max=    3.82 force_mean=    0.51 force_std=    0.69 temperature=135614.14
Steps=    5598 Epot=-3357.90 Ekin= 5018.27 force_max=    4.61 force_mean=    0.52 force_std=    0.72 temperature=134802.41
Steps=    5599 Epot=-3359.51 Ekin= 5005.95 force_max=   13.92 force_mean=    0.57 force_std=    0.97 temperature=134471.41
Steps=    5600 Epot=-3356.84 Ekin= 5004.34 force_max=    4.03 force_mean=    0.51 force_std=    0.71 temperature=134428.12
Steps=    5601 Epot=-3357.22 Ekin= 4986.75 force_max=    4.77 force_mean=    0.51 force_std=    0.71 temperature=133955.68
```

Figure R27(d): Simulation on liquid water example (test.xyz) of Baseline PaiNN (checkpoint file './checkpoint/water_base.pt')

```
[8]: run MD_simulation/MD_run.py --checkpoint checkpoint/TAIP_water.pt --config config.yaml --init_atoms test1.xyz --save_dir ./MD/ --temp 300 --steps 10000

Steps=    9980 Epot=-3236.28 Ekin=   11.07 force_max=    3.17 force_mean=    0.53 force_std=    0.69 temperature=  297.30
Steps=    9981 Epot=-3235.64 Ekin=   10.97 force_max=    2.91 force_mean=    0.52 force_std=    0.69 temperature=  294.72
Steps=    9982 Epot=-3234.52 Ekin=   10.87 force_max=    2.84 force_mean=    0.52 force_std=    0.69 temperature=  292.09
Steps=    9983 Epot=-3234.47 Ekin=   10.78 force_max=    2.60 force_mean=    0.51 force_std=    0.69 temperature=  289.66
Steps=    9984 Epot=-3234.42 Ekin=   10.73 force_max=    2.42 force_mean=    0.51 force_std=    0.69 temperature=  288.13
Steps=    9985 Epot=-3234.63 Ekin=   10.73 force_max=    2.69 force_mean=    0.51 force_std=    0.69 temperature=  288.11
Steps=    9986 Epot=-3234.96 Ekin=   10.78 force_max=    2.91 force_mean=    0.51 force_std=    0.68 temperature=  289.58
Steps=    9987 Epot=-3235.33 Ekin=   10.86 force_max=    2.80 force_mean=    0.50 force_std=    0.67 temperature=  291.67
Steps=    9988 Epot=-3235.08 Ekin=   10.91 force_max=    2.69 force_mean=    0.50 force_std=    0.66 temperature=  293.11
```

Figure R28(a): Simulation on ice example (test1.xyz) of PaiNN-TAIP (checkpoint file './checkpoint/TAIP_water.pt')

```
[4]: run MD_simulation/MD_run.py --checkpoint checkpoint/TAIP_water.pt --config config.yaml --init_atoms test2.xyz --save_dir ./MD --temp 300 --steps 10000

Steps=    9993 Epot=-3359.11 Ekin=   11.17 force_max=    3.50 force_mean=    0.54 force_std=    0.71 temperature=  299.94
Steps=    9994 Epot=-3358.68 Ekin=   11.12 force_max=    3.11 force_mean=    0.54 force_std=    0.72 temperature=  298.69
Steps=    9995 Epot=-3358.21 Ekin=   11.10 force_max=    3.37 force_mean=    0.53 force_std=    0.72 temperature=  298.25
Steps=    9996 Epot=-3358.20 Ekin=   11.14 force_max=    3.33 force_mean=    0.52 force_std=    0.70 temperature=  299.27
Steps=    9997 Epot=-3358.10 Ekin=   11.23 force_max=    2.84 force_mean=    0.51 force_std=    0.68 temperature=  301.57
Steps=    9998 Epot=-3357.98 Ekin=   11.32 force_max=    2.76 force_mean=    0.50 force_std=    0.65 temperature=  304.02
Steps=    9999 Epot=-3358.44 Ekin=   11.36 force_max=    2.75 force_mean=    0.49 force_std=    0.64 temperature=  305.18
Steps=   10000 Epot=-3358.66 Ekin=   11.33 force_max=    2.55 force_mean=    0.48 force_std=    0.64 temperature=  304.22
Steps=   10001 Epot=-3358.38 Ekin=   11.22 force_max=    2.42 force_mean=    0.49 force_std=    0.65 temperature=  301.36
```

Figure R28(b): Simulation on ice example (test2.xyz) of PaiNN-TAIP (checkpoint file './checkpoint/TAIP_water.pt')

```
[8]: run MD_simulation/MD_run.py --checkpoint checkpoint/TAIP_water.pt --config config.yaml --init_atoms test1.xyz --save_dir ./MD/ --temp 300 --steps 10000

Steps=    9993 Epot=-3233.40 Ekin=   10.74 force_max=    2.94 force_mean=    0.54 force_std=    0.72 temperature=  288.43
Steps=    9994 Epot=-3234.71 Ekin=   10.80 force_max=    3.16 force_mean=    0.53 force_std=    0.71 temperature=  290.24
Steps=    9995 Epot=-3233.86 Ekin=   10.89 force_max=    3.18 force_mean=    0.51 force_std=    0.69 temperature=  292.52
Steps=    9996 Epot=-3234.11 Ekin=   10.95 force_max=    3.00 force_mean=    0.50 force_std=    0.68 temperature=  294.14
Steps=    9997 Epot=-3234.06 Ekin=   10.96 force_max=    2.66 force_mean=    0.51 force_std=    0.68 temperature=  294.38
Steps=    9998 Epot=-3233.84 Ekin=   10.92 force_max=    2.58 force_mean=    0.51 force_std=    0.68 temperature=  293.45
Steps=    9999 Epot=-3233.26 Ekin=   10.88 force_max=    2.59 force_mean=    0.52 force_std=    0.70 temperature=  292.25
Steps=   10000 Epot=-3234.33 Ekin=   10.87 force_max=    2.93 force_mean=    0.52 force_std=    0.70 temperature=  291.95
Steps=   10001 Epot=-3235.15 Ekin=   10.91 force_max=    3.00 force_mean=    0.52 force_std=    0.70 temperature=  293.20
```

Figure R28(c): Stability Simulation on ice example (test3.xyz) of PaiNN-TAIP (checkpoint file './checkpoint/TAIP_water.pt')

```
[ ]: run MD_simulation/MD_run.py --checkpoint checkpoint/TAIP_water.pt --config config.yaml --init_atoms test.xyz --save_dir ./MD/ --temp 300 --steps 10000

Steps=    9993 Epot=-3338.38 Ekin=   11.29 force_max=    2.36 force_mean=    0.48 force_std=    0.64 temperature=  303.14
Steps=    9994 Epot=-3338.76 Ekin=   11.33 force_max=    2.37 force_mean=    0.47 force_std=    0.64 temperature=  304.30
Steps=    9995 Epot=-3338.58 Ekin=   11.44 force_max=    2.36 force_mean=    0.46 force_std=    0.61 temperature=  307.34
Steps=    9996 Epot=-3339.11 Ekin=   11.56 force_max=    2.21 force_mean=    0.44 force_std=    0.59 temperature=  310.52
Steps=    9997 Epot=-3339.43 Ekin=   11.62 force_max=    2.29 force_mean=    0.43 force_std=    0.57 temperature=  312.02
Steps=    9998 Epot=-3339.50 Ekin=   11.57 force_max=    2.22 force_mean=    0.43 force_std=    0.57 temperature=  310.75
Steps=    9999 Epot=-3339.03 Ekin=   11.42 force_max=    2.31 force_mean=    0.45 force_std=    0.59 temperature=  306.76
Steps=   10000 Epot=-3339.05 Ekin=   11.21 force_max=    2.33 force_mean=    0.47 force_std=    0.61 temperature=  301.16
Steps=   10001 Epot=-3338.75 Ekin=   11.00 force_max=    2.27 force_mean=    0.48 force_std=    0.63 temperature=  295.58
```

Figure R28(d): Simulation on liquid water example (test.xyz) of PaiNN-TAIP (checkpoint file './checkpoint/TAIP_water.pt')