Supplementary Information (SI) for Nanoscale.
This journal is © The Royal Society of Chemistry 2024

Harnessing Machine Learning for Efficient Large-Scale Interatomic Potential for Sildenafil and pharmaceuticals containing H, C, N, O, and S

E. Nikidis[1,2], N. Kyriakopoulos[1,2], R. Tohid[3], K. Kachrimanis[4,2] and J. Kioseoglou[1,2,*]

[1]Physics Department, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece
[2]Center for Interdisciplinary Research & Innovation, Aristotle University of Thessaloniki, Greece
[3]Center of Computation and Technology, Louisiana State University, 70803 Baton Rouge, USA
[4]Pharmaceutical Technology Department, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

Supplementary Information

All training process was performed with the Allegro code available from Materials Intelligence Group, Harvard University on the projects GitHub repository https://github.com/mir-group/allegro at the time available under release v.0.2.0. (currently unavailable). Allegro is not a standalone software but a an extension package for the NequIP code available at https://github.com/mir-group/nequip, the used version is a user forked version https://github.com/Hongyu-yu/nequip of the original code, on the para_stress branch, with v.0.6.0., under the git commit 6ca00ac. Also, for the training PyTorch was used with version 1.11.0+cu113 (cuda enabled), and Python with v.3.9.18..

The crystal lattice relaxations and elastic constant calculations and melting temperature calculation were run with the LAMMPS code available at https://github.com/lammps/lammps.git under the git commit 6a8ca34 modified pre compilation with the user forked pair_allegro stress branch code available at https://github.com/Hongyu-yu/pair_allegro, git commit b20f966.

The elastic constant calculation was performed using one of the example scripts in LAMMPS repository available at https://github.com/lammps/lammps/blob/develop/examples/ELASTIC/in.elastic in the development branch, under the git commit ae2a7e2.

The Density Functional Theory (DFT) calculations we perform for the test compound (Iso)Sildenafil were done using VASP 6.1.1 · and the PBE GGA (PAW_PBE S 06Sep2000) approximation for the exchange-correlation energy. The standard PAW VASP pseudopotentials were used. The process required three different steps. The first relaxation of the sample with ISIF =3. With ISIF=3 there is calculation of the forces and stress tensor. Also, all the possible degrees of freedom that are allowed to change during the relaxation process (ionic positions, cell volume, cell shape) are activated and used. The second relaxation of the sample with ISIF =1. Relaxation using forces and only the trace of the stress tensor and the degrees of freedom that change are only the positions of the ions. Finally, the stress calculation with ISIF =3 and IBRION= 6. The cutoff energy of the plane-wave basis was 500 eV. The partial occupancies were treated using the Gaussian smearing (ISMEAR=0) with a width of 0.05 eV. One k point was used at gamma and the elastic constants were determined with 0.1Å deformation. Ovito basic 3.8.5 was used to visualize the molecules.

For the training process of the large 1GB in size and 721,662 different molecules we used the Nvidia A100 GPU. The final potential training took 14 days of continuous training in a single GPU without including the hours of fine tuning the dataset in previous attempts. During that period, the academic community, encompassing a broad spectrum of disciplines beyond AI, grappled with adapting to evolving tools, hardware requirements, and the broader changes necessitated by advancements in AI research. Securing access to high-performance GPUs for extended durations proved challenging due to the stringent time constraints imposed by academic clusters. However, with AI now in its "golden age," academic hardware capabilities have advanced sufficiently to accommodate longer training periods, rendering a 14-day duration no longer an exceptional undertaking. For the case of DFT simulations that were executed using VASP on 320 CPU cores (Sandy Bridge - Intel(R) Xeon(R) E5-4650v2), 10 computation hours were required for the calculations of elastic constants. This amount of time compared to the elastic constant LAMMPS calculation which averaged around 3 hours using 64 CPU cores (Intel(R) Xeon(R) Platinum 8358) is a serious improvement . The real benefit of this approach is that the interatomic potential is transferable to other molecules that consist of the same atom pairs, that are

even more complex in size, shape and interactions where DFT couldn't handle them with a reasonable amount of CPU cores in a reasonable amount of time.

In an effort to be as transparent as possible with the training process the specific details for configuring and successfully obtaining a potential should be analyzed. The hyperparameters are parameters that define the model architecture and the learning process.

In more detail :

- **r_max:** This is the radial cutoff for the atomic environment. It determines how far out from an atom the model will consider when determining the atomic environment. The value is typically chosen based on the typical bond lengths in the system being studied. The value we used is 4.5.
- **avg_num_neighbors:** This is used to normalize the sum of the atomic environment. If set to auto, the system will automatically compute it.
- **BesselBasis_trainable:** This determines whether the roots of the Bessel function, which are used in the radial basis function, are trainable parameters. If set to true, the model can adjust these values during training to better fit the data.
- **PolynomialCutoff_p[1]:** This is the p-parameter in the envelope function. It's used in the cutoff function that ensures the atomic environment smoothly goes to zero at the cutoff radius. The value is typically chosen based on the smoothness required. Higher value makes it sharper. The sharper cutoff might be useful in situations where you want the model to strictly limit the influence of atoms beyond a certain distance (r_max)This could be the case if atoms beyond this distance have negligible impact on the properties you're trying to predict, and you want to reduce noise from these distant atoms. On the other hand, a smoother cutoff (lower PolynomialCutoff_p) might be useful when atoms beyond the cutoff distance still have some influence on the properties you're trying to predict. A smoother cutoff allows this influence to taper off more gradually, which might lead to more accurate predictions. We experimented with this value and the observations we got is that the value 6 we used to include enough interactions beyond the 4.5 of the radius without including too many interactions, making the model training heavy.
- **l_max:** This is the maximum order of spherical harmonics used in the atomic environment. Higher values can capture more complex environments but are more computationally expensive. According to the developers number 3 used for training this potential is the most accurate order possible and the slowest to compute. During trials or tests setting up the model I would advise using the baseline option which is 1.
- **parity**: This determines whether to include E(3)-symmetry / parity. Different settings can capture different levels of symmetry in the atomic environment. The options o3_full, o3_restricted, and so3 refer to different types of symmetry groups.
- **o3_full:** This option includes full O(3) symmetry, which is the group of all rotations in three dimensions.
- **o3_restricted:** This option includes a restricted version of O(3) symmetry.
- **so3:** This option includes SO(3) symmetry. The value used during training is the full O(3) symmetry.
- **num_layers:** This is the number of tensor product layers in the model. More layers can capture more complex relationships but are more computationally expensive. According to the developer 1-3 is usually the best option , larger is more accurate but slower. The increase in accuracy was worth the extended training time.
- **env_embed_multiplicity:** This is the number of features in the atomic environment. More features can capture more complex environments but are more computationally expensive. According to the developer the options are 1, 4, 8, 16, 64, 128. Since the dataset was large 64 was the most optimal choice since it took the most reasonable time for each step of the training process.

- **embed_initial_edge:** This determines whether to embed the initial edge in the atomic environment. If true, the model includes information about the initial state of each bond in the atomic environment. When embed_initial_edge is set to true, the model includes information about the initial edge in its representation of the molecular system. This can help the model capture important structural information, especially in systems where the initial edge has a significant impact on the properties being predicted.
- **two_body_latent_mlp_latent_dimensions:** These are the dimensions of the hidden layers in the 2-body embedding MLP. Larger dimensions can capture more complex relationships but are more computationally expensive. The value used on this was default by the developer [128, 256, 512, 1024].
- **latent_mlp_latent_dimensions:** These are the dimensions of the hidden layers in the latent MLP. Larger dimensions can capture more complex relationships but are more computationally expensive. The value used on this was default by the developer [1024, 1024, 1024]
- **latent_resnet:** This determines whether to use a ResNet-style update in the scalar latent space. If true, the model includes skip connections that can help with training deep networks. The value used was 'true'.
- **edge_eng_mlp_latent_dimensions:** These are the dimensions of the hidden layers in the per-edge energy MLP. Larger dimensions can capture more complex relationships but are more computationally expensive. The value used on this was default by the developer [128]
- **n_train:** This is the number of training samples to use. More samples can lead to a better trained model but are more computationally expensive.The value used on this was 300.000
- **n_val:** This is the number of validation samples to use. More samples can provide a better estimate of the model's performance but are more computationally expensive.The value used on this was 10.000
- **batch_size:** This is the number of samples to process at once during training. Larger batch sizes can be more computationally efficient but may lead to less stable training. Greatly depends on the amount of VRAM the GPUs have available. Our earliest test with the nVidia Tesla V100 16 GB variant were only possible with batch size 1. Due to a hardware upgrade to nVidia A100 which at the point of training was state of the art batch size up to 5. In general, this parameter requires trial testing depending on the hardware specification of the system.
- **max_epochs:**This parameter sets the maximum number of training iterations, or "epochs", that the model will undergo. Each epoch is a complete pass through the entire training dataset. Setting a higher number for max_epochs allow the model more opportunities to learn from the data, potentially leading to a better trained model. However, more epochs also require more computational resources and time. In this case, max_epochs is set to 100,000. According to the developer, this large number is chosen to ensure that other early stopping criteria will likely be met before reaching the maximum number of epochs, effectively using max_epochs as a fail-safe.The value used was 100.000
- **Learning_rate:** This parameter sets the step size for the optimizer during the training of the model. It controls how much the model's parameters are adjusted at each training step. The choice of learning rate can significantly impact the model's performance and training time. A high learning rate allows the model to learn quickly, but it may overshoot the optimal solution. On the other hand, a low learning rate ensures that the model makes careful, small adjustments, but it may slow down the training process and the model might get stuck in a suboptimal solution. In this case, the learning rate has been set to 0.001. This is a decrease from the default value of 0.002, based on the developer's recommendation that values of 0.002 and 0.0005 tend to work best. Lowering the learning rate further could potentially increase the accuracy of the model, as it allows the model to fine-tune its parameters. However, it would also mean that each training

step results in a smaller update to the model's parameters, which could significantly slow down the training process and get it stuck in a suboptimal solution. The chance of that increases as this value is decreased and also depends on the dataset.

Additionally, there is the loss function. Which is used as a weighted loss function with different weights for forces and total energy. The loss function measures the difference between the model's predictions and the actual values. It's what the model tries to minimize during training. When the model is predicting multiple quantities (like forces and total energy in this case), the loss function is often a weighted sum of the individual losses for each quantity. The loss_coeffs parameter sets these weights. In your case, it's set to 10 for forces and 1 for total energy. This means that the model will put 10 times more emphasis on accurately predicting forces than on predicting total energy. Also, in the yaml it's specified that the Mean Squared Error (MSE) loss function should be used for total energy, and that this loss should be calculated on a per-atom basis. Since the tests to evaluate the performance of the potential were designed around elastic properties, so the dynamics of the compound, how it moves or changes shape over time, then accurately predicting the forces theoretically is more important. If the goal was the stability of the compound, its potential conformations, or its interactions with other molecules, then accurately predicting the total energy might be more important.

The model uses a train-validation split specified by `train_val_split` (random or sequential). The number of validation samples is specified by `n_val`. The model uses early stopping based on validation loss (`validation_loss` in `early_stopping_patiences`) to prevent overfitting. Overfitting is prevented using several techniques:

- Early stopping: Training is stopped if there's no improvement on validation loss for a specified number of epochs `**early_stopping_patiences**`.
- Exponential moving average of the weights `use_ema`: This helps to smooth out the weights and prevent overfitting.
- Learning rate scheduler `**lr_scheduler_name**`: The learning rate is reduced if there's no improvement for a specified number of epochs (`lr_scheduler_patience`), which can help to prevent overfitting.
- Weight decay in the optimizer `**weight_decay**` in `**optimizer_params**`: This adds a regularization term to the loss function, which can help to prevent overfitting.

As mentioned above in the hyperparameters section the model was trained on a dataset of 300,000 samples `n_train: 300000` and validated on a separate set of 10,000 samples `n_val: 10000`. These validation samples were not used during the training process, so they represent "unseen" data for the model. The validation set helped us monitor the model's performance on data it had not been trained on, allowing us to ensure that the model was learning to generalize from the training data, rather than simply memorizing it. This process of validation during training helped tuning the model's parameters for optimal performance.

References

[1] Stocker, S., Gasteiger, J., Becker, F., Günnemann, S., & Margraf, J. T. (2022). How robust are modern graph neural network potentials in long and hot molecular dynamics simulations? In Machine Learning: Science and Technology (Vol. 3, Issue 4, p. 045010). IOP Publishing. https://doi.org/10.1088/2632-2153/ac9955