Notifications 9
DMs
Calendar
Search
Class feed
MSE598ML: Machine Learning for MatSE
All categories
New post
This week
computation resource within U... #104
As professor discussed in class abo...
0 15 minutes ago
Clustering with DBSCAN/HDB... #103
I'm studying crystallization using a c...
0 a day ago
Using ML to predict Mechanic... #102
I am researching how changes in ini...
0 a day ago
Gaussian process regression #101
In GPR, can the prior functions be i...
0 a day ago
Coefficients of many-body ter... #100
I'm the person asking about the phy...
0 2 days ago
Possibility of general MD poten... #99
Professor talked about transferabili...
0 2 days ago
How to decide which part of th... #98
When using a pre-trained neural ne...
0 2 days ago
Exploring the Role of Neural Ne... #97
As I dive deeper into battery materi...
Rooms
Files
Grades
Settings
Collapse
Wenwen
Active
+1 online now

Jahid Emon 1524 asked a question 2 days ago
Visible to: Everyone

Resolved

MD vs MC #88

General

As far as I understand, there are two main types of molecular simulations: Molecular Dynamics (MD) and Monte Carlo (MC). How do we decide which method is more appropriate to use in a given situation?

0 0 86 31

Answers Comments

Answer this question

Student answers

Best

3

An instructor (Andre Schleife) endorsed this answer

Golam Mahmud Samdani 283 answered this question 2 days ago

MD is best for studying time-dependent properties (like diffusion, reactions) and non-equilibrium processes since it simulates atomic motion over time. MC, on the other hand, is used for equilibrium properties, statistical sampling, and phase transitions because it relies on random sampling of configurations. I would suggest to use MD for dynamics and MC for equilibrium studies.

3 replies Reply

Jahid Emon 1524 2 days ago

ok, got it.

Reply

Nicholas Saunders 26 2 days ago

Monte Carlo simulations actually can be used for non-equilibrium simulations, which is something that I do in my own research. A different distinction between MD and MC is that MD is deterministic and MC is stochastic.

Another important difference is that MD simulations give you dynamic information, by which I mean time-dependent properties. It's simple to get something like a diffusion coefficient. On the other hand, typical Metropolis Monte Carlo simulations do not tell you anything about time-dependent information, so something like a diffusion coefficient is not directly accessible.

Kinetic Monte Carlo simulations can re-introduce time-dependence into an MC simulation, but they come with their own drawbacks. For instance, the type of KMC simulation that I perform cannot use an interatomic potential as an input, so using ML beforehand would be more difficult. I would argue that MD simulations would be a more natural choice if you are interested in using ML potentials.

1 reply Reply

Jahid Emon 1524 2 days ago

but isn't we use same potential for both MC and MD?

1 reply Reply

Nicholas Saunders 26 2 days ago

For Metropolis MC, you can use an interatomic potential, but you lose the dynamic information that you have in MD. Unless there's some specific benefit to using the Metropolis MC simulations, I'd say you'd probably just want to use MD. The Metropolis MC doesn't really give you any information that MD wouldn't also give you.

For kinetic MC simulations, you have the dynamic information, but in the KMC that I perform you do not use an interatomic potential. My KMC simulations occur on a lattice, so you input the pairwise interactions between atoms. You could map the pairwise interactions from an ML potential, but there's not really much benefit here.

This is why I think that if someone is interested in using ML information for atomistic simulations, the first choice would be an MD simulation

Reply

James Tallman 59 2 days ago

There is some other nuances here in the difference between MD and MC

1. MD is also stochastic in some senses, namely in terms of how the temperature/pressure is controlled by the simulation. The timestep you choose also has an impact on the space you can explore.
2. MC is easy to do incorrectly and hard to do correctly, especially in large systems, because you have to code every single 'move type', i.e. moving single particles or moving clusters of particles together.
3. MD is actually pretty good at getting structural properties. I use MD to get structure of lipid membranes.

As Nicholas pointed out, there are many flavors of MC and MD. Hard particle monte carlo is one I find intuitive.

Reply

An instructor (Andre Schleife) endorsed this answer

1

An instructor (Andre Schleife) endorsed this answer

Wenwen LENG 59 answered this question 2 days ago

In one of my study on two proteins binding, we actually used both methods in sequence, which nicely illustrates their complementary strengths:

- We first used Monte Carlo simulations to broadly explore possible binding conformations. This was especially helpful because we didn't initially know the most likely binding region. MC allowed us to scan the configuration space quickly and identify high-probability interaction sites.
- Once we had identified a promising binding region, we then used Molecular Dynamics to simulate the local dynamics of the binding interface in detail. MD gave us information about structural fluctuations, interaction stability, and the kinetics of the binding process over time.

> MC helped us find where binding might occur.
> MD helped us understand how the interaction unfolds dynamically.

This kind of hybrid strategy is popular in biomolecular modeling and ligand binding studies.

Reply

Notifications 8
DMs
Calendar
Search
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Rooms
Files
Grades
Settings
Collapse

Wenwen
Active

Class feed

MSE598ML: Machine Learning for MatSE

All categories | New post

This week

computation resource within U... #104
As professor discussed in class abo...
0 18 minutes ago

Clustering with DBSCAN/HDB... #103
I'm studying crystallization using a c...
0 a day ago

Using ML to predict Mechanic... #102
I am researching how changes in ini...
0 a day ago
(repeated)

Gaussian process regression #101
In GPR, can the prior functions be i...
0 a day ago

Coefficients of many-body ter... #100
I'm the person asking about the phy...
0 2 days ago

Possibility of general MD poten... #99
Professor talked about transferabili...
0 2 days ago

How to decide which part of th... #98
When using a pre-trained neural ne...
0 2 days ago

Exploring the Role of Neural Ne... #97

+1 online now

Golam Mahmud Samdani 283 asked a question 2 days ago
Visible to: Everyone

Resolved

MD Model Prediction #87

General

I'm trying to understand how molecular dynamics simulations are used to train machine learning models for predicting interatomic potentials—how do we ensure the model captures long-range interactions and remains accurate across different phases or defect structures?

1 0 77 29

Answers | Comments

Answer this question

Student answers

Best

4

An instructor (Andre Schleife) endorsed this answer

Jahid Emon 1524 answered this question 2 days ago

To ensure machine-learned interatomic potentials trained on molecular dynamics data capture long-range interactions and remain accurate across phases or defects, researchers often include diverse training data that spans multiple configurations—equilibrium, strained, defected, and various phases. Models like MACE or NequIP can incorporate long-range physics through specialized architectures or explicit electrostatics. Additionally, techniques like active learning help adaptively sample regions where the model is uncertain, improving generalization to unseen structures.

Reply

2

An instructor (Andre Schleife) endorsed this answer

Wenwen LENG 59 answered this question 2 days ago

Traditional MLIPs (e.g., standard Deep Potential, MTP, GAP) typically rely on **local descriptors**, with only finite cutoff radius (e.g., 6–10 Å).I know that DeePMD group proposed an extension to their Deep Potential model by incorporating long-range electrostatics using a physically inspired approximation:

> The long-range Coulomb interaction between ions and valence electrons is modeled as interactions between **Gaussian charge distributions** placed at atomic and electronic centers.
>
> arXiv:2112.13327

This allows the model to:

- retain local symmetry-preserving neural networks,
- while adding an electrostatic baseline with distance-dependent interactions,
- and handle **heterogeneous charge distributions** more realistically.

In practice, if long-range interactions are important for your system, you'll need to:

- Use or extend a model that includes explicit electrostatics (e.g., DP-ZBL, DP-LR, PhysNet),
- Or combine MLIPs with **charge equilibration schemes** (QEq, HIPNN-QM9, etc.),
- And ensure your training set includes **diverse environments** (e.g., defects, interfaces, multiple phases) to promote generalizability.

Hope that helps.

1 reply Reply

Golam Mahmud Samdani 283 2 days ago

Thanks for explaining this nicely!

Reply

Notifications
DMs
Calendar
Search
Class feed
MSE598ML: Machine Learnin
All categories
This week
computation reso
As professor discu
21 minutes
Clustering with DB
I'm studying crystal
a day ago
Using ML to predi
I am researching ho
a day ago
Rooms
Files
Grades
Settings
Collapse
Wenwen
Active

Jahid's question in Machine Learning for MatSE

Jahid Emon 1524 asked a question 3 months ago
Visible to: Everyone

Resolved

Pseudo Force Field in MD Simulation #6

General

In molecular dynamics simulations, we typically use integration to calculate forces from atomic positions. What if we use machine learning to predict forces directly from atomic positions, bypassing integration? Would this speed up the simulation?

2 0 236 46

Answers Comments

Answer this question

Instructor answers

Best

Andre Schleife ⭐ answered this question 3 months ago

0

You're absolutely on the right track. I don't think you would skip integration, but people absolutely use ML to predict forces based on atomic positions. The advantage is that ML gives you relatively accurate forces much faster than comparable quantum-mechanical methods (like DFT).

Reply

Student answers

Best

Sheikh Mohd Ta-Seen Afrid 306 answered this question 3 months ago

1

Nice question! I actually have some experience working on this particular topic, and it's fascinating to explore the potential of machine learning in molecular dynamics. While bypassing integration to predict forces directly from atomic positions can indeed speed up simulations, the success of this approach depends heavily on the quality of the training data, the model's ability to generalize across different systems, and the computational cost of evaluating the machine learning model itself. If these challenges are addressed, it could revolutionize the field by enabling faster exploration of complex molecular systems.

Reply

Wenwen LENG 59 answered this question 3 months ago

1

Clarification on "Integration" in MD Simulations

I am not entirely sure whether I fully understand your confusion regarding **integration** in MD. In MD simulations, **integration typically refers to updating atomic positions and velocities over time based on forces**, not computing forces themselves.

- **Force Calculation**: In classical MD, forces are derived from empirical potentials (e.g., Lennard-Jones), whereas in **Ab Initio Molecular Dynamics (AIMD)**, they are computed from **first-principles (DFT-based) electronic structure calculations**, which is computationally expensive (scaling as **O(N³–N⁴)** with system size).
- **Time Integration**: Once the forces are known, numerical integration methods (e.g., **Verlet, Velocity-Verlet**) are used to propagate the atomic motion.

Thus, if your question concerns **bypassing force computation**, **machine learning methods can replace this step**. However, **time integration cannot be skipped**, as it is necessary to evolve atomic trajectories.

What Happens If We Use Machine Learning to Predict Forces?

the current machine learning, like **Deep Potential Molecular Dynamics** or **Neuroevolution Potential **, allows us to **directly predict forces from atomic positions without computing them from explicit potential functions**. This means:

- **Bypass the expensive force calculation step**, such as solving Schrödinger's equation in AIMD.
- **NOT bypass integration**, as numerical integration (like Verlet or Velocity-Verlet) is still needed to evolve atomic trajectories over time.

(*I am unsure if your question about integration pertains to force computation or numerical time integration in MD.*)

**Would This Speed Up the Simulation? **

✅ **Yes, it can significantly speed up simulations**, especially compared to AIMD, which require solving quantum mechanical equations for each step. Because:

- **DFT-based AIMD**: Computational cost scales as **O(N³–N⁴)** with the number of atoms.
- **ML-based MD (e.g., DeePMD)**: Once trained, the force evaluation scales **linearly** with system size, **O(N)**, similar to classical force fields.

🚀 **If your goal is to accelerate MD simulations by replacing traditional force fields with ML-based force predictions, then ML can help drastically reduce computational cost while maintaining near-DFT accuracy.**

2 replies Reply

Jahid Emon 1524 3 months ago • (edited)

My mistake; now I can recall the underlying principle of MD simulation, ie. solving Newton's 2nd law, to determine the trajectory of each atom using numerical integration. I appreciate your time and effort to give such an elaborate response. However, recent studies suggest that Machine Learning Interatomic Force Fields (MLIFs) are often limited to specific problems and require substantial computational resources for training. Given this high training cost, do MLIFs still outperform traditional force fields in terms of overall efficiency and accuracy?

1 reply Reply

James Tallman 59 3 months ago

Someone in my lab actually does a lot of work on this (statt lab). In their work, they are trying to get "effective interactions" of differently shaped colloids (as typically MD only solves for particles, but often ellipsoids, superballs, and other shapes are of interest).

https://ui.adsabs.harvard.edu/abs/2024JChPh.160x4901A/abstract

Is this a specific problem that required a lot of resources for training? Yes. But, it is still a very general solution to a wide number of problems. Also, the speed-up is hardware dependent but can be between 2x and 25x speed-up (comparing a naive implementation of the MLIFs vs a highly optimized MD simulation). This implies the possibility of significant more speed-up, while the accuracy is there.

I'm not sure if this completely answers your question, but I thought I could highlight work going on in my lab that was relevant.

Reply

Wenwen LENG 59 3 months ago

In most materials science studies, the training process for MLIFs does not necessarily impose an high computational cost. The crucial factor is not just the training itself but rather the preparation of a high-quality and accurate dataset for training the MLIFs. The most challenging aspect lies in generating reliable raw data, typically from DFT calculations, to ensure the robustness of the trained model.

One of the key advantages and diffuults of MLIFs over traditional force fields is their potential for transferability to unforeseen scenarios—those that are either difficult or entirely inaccessible to DFT calculations. While ML models can accurately reproduce known data, their true challenge lies in extrapolating beyond previously seen configurations, making generalization a critical research focus.

For instance, if one aims to develop a reliable interatomic potential for a **multicomponent alloy** in **solid-state electrolytes (SSEs)**, a straightforward approach would be to include data for all possible alloy compositions in the training dataset. In theory, this would allow the machine learning model to learn a comprehensive representation of the interatomic interactions. However, the challenge lies in the fact that constructing such a dataset is **both computationally expensive and technically difficult**. If such an extensive dataset is available, then the problem is largely solved.

The reality, however, is that for **simple systems**, obtaining sufficient high-quality data through first-principles calculations is relatively easy. But for **complex multicomponent systems**, we often have **only a limited amount of data**, which is far from sufficient to develop a robust force field. Ideally, if machine learning models could **leverage information from elemental systems and binary alloys**, and then **extrapolate or transfer knowledge** to predict the behavior of ternary, quaternary, and even more complex systems, that would be the most desirable outcome.

While **fitting data is straightforward**, addressing **the challenge of sparse data in complex systems**—even with **active learning techniques**—remains highly **computationally expensive**. This **cost** is particularly reflected in the **iterative active learning process**, which requires continuous exploration and refinement. However, despite these challenges, **this approach is not just theoretically feasible—it has already been successfully demonstrated in practical applications**.

Reply

Enock Bunan 16 answered this question a month ago

0

I think the answer is yes, but for the accuracy of the results, the ML must be trained on high-quality QM or experimental data.

Reply

Events See all
Today May, 8th
No events today
Your day is all clear!

Notifications 10
DMs
Calendar
Search

Class feed
MSE598ML: Machine Learning for MatSE

My posts
New post

Last week

How Transferable Are Equilibriu... #64
Hi everyone, Another question for m...
0 9 days ago

Discussion: Single vs Double Pr... #63
Hi everyone, I'd like to share an obse...
0 9 days ago

Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Rooms
Files
Grades
Settings
Collapse

Wenwen
Active

3 online now

Wenwen LENG 59 asked a question 9 days ago
Visible to: Everyone

Resolved

How Transferable Are Equilibrium-Trained ML Potentials to Nonequilibrium Regimes? #64

General

Hi everyone,

Another question for me. We often evaluate machine-learned interatomic potentials using benchmarks like **OC20** and **Matbench**. These datasets are widely adopted, but they're dominated by **relaxed or near-equilibrium structures**.

This leads to a question that's been bothering me for a while:

> **Can MLIPs trained (or pre-trained) on equilibrium data be trusted in nonequilibrium scenarios?**

Examples include:

- highly strained structures,
- high-energy defects or collisions,
- cascade events or fracture simulations.

Even for state-of-the-art models like **MACE**, **NequIP**, or **GemNet**, if the training distribution doesn't cover these regions, aren't we just *extrapolating blindly*?

I'm aware of two typical strategies to improve model performance in such cases:

- ✅ **Active learning / on-the-fly sampling**
- ✅ **Fine-tuning with domain-specific high-energy data**

But both essentially boil down to *adding more data*.

So here's my real question:

> **If we always need to add new data for each task, how "general" are general-purpose MLIPs in practice?**

Are pretrained models truly transferable, or just good warm starts?

Thanks!

0 0 82 31

Answers Comments

Answer this question

Student answers

Best

3

An instructor (**Andre Schleife**) endorsed this answer

Jahid Emon 1524 answered this **question** 8 days ago

That's a great question, and honestly, a lot of people working with MLIPs are wrestling with the same thing. You're right — most benchmark datasets like OC20 or Matbench are heavily skewed toward relaxed or near-equilibrium structures, which means MLIPs trained on them can struggle when pushed into high-strain, high-energy, or defect-heavy regimes. Even top models like MACE or NequIP will likely be extrapolating if those regions weren't in the training set. Active learning and fine-tuning definitely help, but as you said, they're just smarter ways of adding more data. So in practice, these "general-purpose" MLIPs often end up being more like solid starting points that still need tailoring for specific applications. Transferability sounds great in theory, but in high-stakes simulations like fracture or cascades, it's risky to rely on pretrained models alone without domain-specific validation.

1 reply Reply

Wenwen LENG 59 8 days ago

Thanks so much for your thoughtful response — that really clarified a lot!

As a follow-up: if I fine-tune or distill a general-purpose MLIP using my domain-specific data, I'd expect it to require **less training data** than training a model from scratch.

But in that case, wouldn't the resulting potential (after distillation) still be **structurally more complex** than a model trained only on my own dataset (especially when used in MD simulations), since it inherits a more unified and general representation of atomic environments?

Just wondering — is that added complexity a trade-off worth considering in practice?

To be clear, I'm not against distillation or transfer learning — I'm just trying to understand how much **computational cost or inference overhead** we're introducing when we use a distilled "general" model in production simulations.

Reply

Notifications 10
DMs
Calendar
Search

Class feed
MSE598ML: Machine Learning for MatSE
My posts
New post
Last week
How Transferable Are Equilibriu... #64
Hi everyone, Another question for m...
0 9 days ago
Discussion: Single vs Double Pr... #63
Hi everyone, I'd like to share an obse...
0 9 days ago

Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Class feed
Rooms
Files
Grades
Settings
Collapse

Wenwen
Active
+2 online now

Wenwen LENG 59 asked a question 9 days ago
Visible to: Everyone
Resolved

Discussion: Single vs Double Precision in ML-based Interatomic Potentials #63

General

Hi everyone,

I'd like to share an observation and raise a discussion regarding the use of **single (`float32`) vs double precision (`float64`)** in machine learning-based interatomic potentials — particularly in **DeepMD-kit**, though I suspect this issue may be more general.

🔍 Observation

During the **training phase** of DeepMD neural network models, we typically observe **little to no performance difference** between using `float64` and `float32`.

However, once the trained model is **deployed** — for example, in **LAMMPS for production-scale molecular dynamics simulations** — the **performance gap becomes substantial**, often with `float32` being significantly faster (e.g., 2x–8x speedup).

💬 Questions I'd like to discuss:

- **Why is `float32` significantly faster than `float64`**, especially on GPUs?
- **Why does this precision difference appear negligible during training**, but become prominent during inference?
- Is this performance difference specific to DeepMD-kit, or common to all **deep neural network-based MLIPs**, such as **MACE**, **NequIP**, or **SchNet**?
- **Is this difference inherent to NN-based inference**, or can it be mitigated through tools or model conversion?
- Do **non-deep-learning ML models** (e.g., moment tensor-based **MTP**, Gaussian process-based **GAP/GPUMD**) also suffer from performance or stability issues related to precision choice?

If you've encountered this issue or have experience converting models for lower-precision inference, I'd love to hear how you handled it.

What's your preferred trade-off between **accuracy** and **speed** in production-level MLIP simulations?

Thanks!

0 0 107 32

Answers Comments

Answer this question

Student answers

Best

2

Jahid Emon 1524 answered this question 8 days ago

Float32 is significantly faster than float64, especially on GPUs, because modern deep learning hardware (like NVIDIA Tensor Cores) is highly optimized for single precision. During training, the precision difference is less noticeable due to the dominance of complex operations and mixed-precision strategies, but during inference, float32's speed advantage becomes clear. This behavior isn't unique to DeepMD-kit—it also applies to other MLIP frameworks like MACE or NequIP. In contrast, classical ML models like GAP or MTP are often CPU-bound and use float64 more for stability than speed. A common practice is to train in float64 for accuracy, then convert the model to float32 for faster, large-scale simulations when numerical stability allows.

1 reply Reply

Wenwen LENG 59 8 days ago

Thanks a lot for your explanation!

Just to clarify — in your view, does **double precision** typically provide **better accuracy during training**?

Is that why you mentioned that *"a common practice is to train in float64 and convert to float32 for inference"*?

If so, it makes sense on why not just use single precision from the start on traning.

1 reply Reply

Jahid Emon 1524 8 days ago

in float32, the max of the bit is 128, and in float64 have 256, if I am not wrong. if you start with float32 there is a chance you get lower bound, e.g. 5.3258 if you use float64 you get 5.326 but if you use float32 then you get 5.325, as max bit size exceeds so truncated not round off.

1 reply Reply

Wenwen LENG 59 8 days ago

great

Reply