Supplementary Information: High-performance training and inference for deep equivariant interatomic potentials

Chuin Wei Tan,[1] Marc L. Descoteaux,[1] Mit Kotak,[2] Gabriel de Miranda Nascimento,[3] Seán R. Kavanagh,[4] Laura Zichi,[1] Menghang Wang,[1] Aadit Saluja,[1] Yizhong R. Hu,[1] Tess Smidt,[5] Anders Johansson,[6] William C. Witt,[1] Boris Kozinsky,[1,7] and Albert Musaelian[1,8]

[1]*John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA*

[2]*Center for Computational Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA*

[3]*Department of Materials Science and Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA*

[4]*Center for the Environment, Harvard University, Cambridge, MA, USA*

[5]*Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, USA*

[6]*Sandia National Laboratories, Albuquerque, NM, USA*

[7]*Robert Bosch LLC Research and Technology Center, Watertown, MA, USA*

[8]*Mirian Technologies Inc., Boston, MA, USA*

CONTENTS

I. SPICE 2 ALLEGRO MODEL TRAINING

A. Mixed-Precision Architecture

The Allegro model balances the performance benefits of lower-precision arithmetic with the need for numerical accuracy by employing a mix of double- and single-precision operations. Specifically, the initial embedding of edge vectors between a central atom and its neighbors to a radial-chemical and spherical harmonics basis is performed largely in double-precision before casting the resulting embedded features to single-precision. Subsequent operations in the Allegro layers, including the multilayer perceptrons (MLPs) acting on scalar features and the Clebsch–Gordan tensor products acting on tensor features, are carried out in single precision. The learned representations are then passed through a readout MLP, also in single precision, and finally cast back to double precision when producing the energy prediction. The backward pass used to compute forces and other derivative quantities follows this sequence in reverse, preserving the same precision transitions.

B. Allegro Model Hyperparameters

All three models used the Bessel function Radial embedding with 8 Bessel features, untrainable Bessel frequencies, and a polynomial cutoff with $p = 6$. The two-body multilayer perceptron (MLP) used an embedding dimension of 32, with hidden layers of depth 2, and width 64, using the SiLU nonlinearity. The output dimension of the scalar embedding was 128. All models used the full set O(3) parities, had tensor product path-channel coupling enabled, and had weights initialized for forward normalization. The readout MLP had depth 1, width 128, and no nonlinearity.

The small and medium models used two Allegro layers, 64 scalar features, 64 tensor features, and an Allegro MLP with depth 2, width 256, and SiLU nonlinearity, while the large model used three Allegro layers, 128 scalar features, 128 tensor features, and an Allegro MLP with depth 3, width 512, and SiLU nonlinearity. The small model used $\ell_{\text{max}} = 2$, while the medium and large models used $\ell_{\text{max}} = 3$.

The repulsive Ziegler-Biersack-Littmark (ZBL) potential [1] was applied in addition to the Allegro models' base predictions. The per-atom energy shifts for each elemental species were set to the isolated atom energy of the most common charge state in the dataset. The per-atom energy scales were set to the root mean square force magnitude for each elemental species in the training dataset.

C. Training Hyperparameters and Procedure

The three Allegro models trained on the SPICE 2 dataset used similar training strategies with a few differences. The batch size was set to 256. We employed the AdamW optimizer [2] with a weight decay of 10^{-6}. An exponential moving average (EMA) of the model weights was used for validation, testing, and inference, with an EMA decay factor of 0.999. The loss was a 1:1 weighting of the per-atom-energy and force mean squared errors. The `ReduceLROnPlateau` learning rate scheduler was used with a reduction factor of 0.8, patience of 10 epochs, a threshold of 10^{-6}, and minimum learning rate of 10^{-7}. It tracked the weighted sum of the validation per-atom-energy and force loss. Initially the learning rate was set to 0.002. Training was set to stop if the average of validation root mean square force and per-atom-energy errors did not improve by at least 10^{-9} after 20 epochs for the large model and 200 epochs for the small and medium models. The models were set to train for a maximum of 1000 epochs, or three and a half days.

After this first stage, a model was initialized from the "best" checkpoint, and trained with the same settings, but a weighting of 25:1 for per-atom-energy and force loss, and an initial learning rate of 0.001. The small and medium models were trained for another 24 hours, while the large model was trained for 42 more hours. The large model was then initialized from the "best" checkpoint from the extended training process with the same loss weighting and initial learning rate of 0.001, and trained for another three and a half days.

The SPICE 2 models were trained using the Frontier supercomputer at the Oak Ridge Leadership Computing Facility. Training was performed using PyTorch v2.6.0, ROCM 6.2.4, `nequip` version 0.7.0, `allegro` version 0.4.0. The small and medium models were trained using a Distributed Data Parallel paradigm across 4 nodes (32 ranks), and the large model was trained across 8 nodes (64 ranks).

D. SPICE 2 and TorsionNet 500 Dataset Details

Experiments were performed using the SPICE 2 dataset [3], version 2.0.1, accessed from https://doi.org/10.5281/zenodo.10975225. The dataset was randomly split into train, validation, and test partitions in a 90:5:5 ratio. The SPICE 2 dataset spans multiple charge states for many of the included atomic species and of the system's total charge. We chose to train only on atomic configurations where the total charge of the system is zero. Non-conforming atomic configurations were removed from these splits, leaving 1,834,949 of the original 2,008,126 atomic configurations. Additionally, to avoid geometric degeneracy stemming from the locality of the Allegro potential, 192 atomic configurations with isolated atoms beyond the model cutoff r_{max} were removed, mostly from the ion-pairs subset. Altogether, the models were trained, validated, and tested on 1,651,281, 91,737 and 91,739 atomic configurations, respectively.

The TorsionNet 500 benchmark, computed at the SPICE level of theory, and the SPICE 2 test set dimers split were downloaded using the tools provided at https://github.com/openmm/spice-dataset [3–5]. Consistent with the approach of Eastman *et al.* [5], the four atomic configurations with force magnitudes greater than 1 Ha/Bohr were removed from the dimers split of the SPICE 2 test set. The other three splits of the SPICE 2 test dataset were accessed from https://doi.org/10.5281/zenodo.11455132. The SPICE 2 test set with no further alterations was used for evaluating the models (7,976 atomic configurations) in addition to a version where only neutral systems were kept (6,448 atomic configurations), and a version where only systems containing atoms of neutral formal charge were kept (5,998 atomic configurations). The TorsionNet 500 benchmark was used in full when evaluating the models (12,000 atomic configurations).

Table 1 further compares the results for the TorsionNet 500 benchmark obtained in this work (as shown in Table 2 of the original manuscript) with the models introduced by Eastman *et al.* [5] and Kovács *et al.* [6], along with their most relevant model hyperparameters. We note that, due to the models architecture being different, hyperparameter values are not directly comparable between each other, although they still provide a sense of model complexity. An important note is that values reported by Kovács *et al.* [6] are from models trained with version 1 of the SPICE dataset, whereas Eastman *et al.* [5] and this work uses version 2.

Table 1. **Comparison of model hyperparameters and torsion barrier height MAEs.**

N_{layers} and N_{features} stand for number of layers and number of features, respectively. Values marked by (*) represent numbers read from a plot instead of reported on text. $l_{\max}$ numbers reported for Kovács *et al.* [6] represent the maximum value of l used to build atom messages (*max L* parameter reported in their manuscript [6]).

			TorsionNet 500	Hyperparameters			
ref.	Architecture	Version	Barrier MAE [meV]	r_{cut} (Å)	$l_{\max}$	N_{layers}	N_{features}
Eastman *et al.* [5]	TensorNet [7]	small	17.5	10	2	1	128
		medium	13.5	10	2	1	128
		large	11.0	10	2	2	128
Kovács *et al.* [6]	MACE [8]	small	23.1*	4.5	0	2	96
		medium	13.2*	5.0	1	2	128
		large	10.5*	5	2	2	192
This work	Allegro [9]	small	22.75	6	2	2	64
		medium	15.42	6	3	2	64
		large	11.37	6	3	3	128

II. ADDITIONAL SPICE 2 MODEL ACCURACY AND PERFORMANCE BENCHMARKING

On the held out 5% of the SPICE 2 dataset we reserved for testing, we report energy and force errors on different subsets.

Table 2: Additional model accuracy metrics

Split Name	**Model**	**Per Atom Energy (meV)**		**Forces (meV/Å)**	
		MAE	**RMSE**	**MAE**	**RMSE**
SPICE Amino Acid Ligand	small	0.9	1.6	20.4	40.1
	medium	0.7	1.4	16.8	33.9
	large	0.5	1.0	12.3	26.2

Continued on next page

Table 2 – continued from previous page

Split Name	Model	Per Atom Energy (meV)		Forces (meV/Å)	
		MAE	**RMSE**	**MAE**	**RMSE**
SPICE DES370 K Monomers	small	0.8	1.1	14.8	22.5
	medium	0.7	1.0	11.7	17.6
	large	0.4	0.5	7.7	11.7
SPICE DES370 K Dimers	small	0.9	1.5	15.1	29.0
	medium	0.8	1.2	12.4	24.6
	large	0.5	0.9	8.6	16.1
SPICE Dipeptides	small	0.8	1.2	25.0	40.6
	medium	0.8	1.2	20.5	34.2
	large	0.5	0.8	14.7	24.5
SPICE Ion Pairs	small	69.4	209.1	190.7	511.8
	medium	24.4	44.8	126.9	310.4
	large	30.7	71.7	122.7	420.5
SPICE PubChem	small	1.4	2.8	32.8	69.1
	medium	1.1	2.4	26.4	60.6
	large	0.6	1.7	18.1	51.6
SPICE Solvated Amino Acids	small	1.5	1.8	39.8	62.3
	medium	1.4	1.7	34.3	52.6
	large	1.1	1.3	27.1	41.0
SPICE Solvated PubChem	small	1.6	2.3	43.7	79.0
	medium	1.4	1.9	37.1	66.8
	large	1.1	1.4	29.0	50.7
SPICE Water Clusters	small	1.8	2.4	33.5	45.6
	medium	1.5	2.0	30.0	40.6
	large	1.1	1.6	23.6	31.8

III. INFERENCE DETAILS

All inference experiments were evaluated using molecular dynamics simulations performed in LAMMPS [10]. The inference experiments with AMD MI250X GPUs performed on OLCF Frontier used PyTorch 2.6.0, ROCM 6.2.4, LAMMPS version “stable 29Aug2024, Update 1”. The inference experiments with NVIDIA A100 40GB and 80GB GPUs performed on NERSC Perlmutter used Pytorch 2.6.0, CUDA 12.4, LAMMPS version “2Apr2025”. The inference experiments with NVIDIA A100 80GB GPUs and NVIDIA H100 GPUs performed on Harvard FASRC Cannon used Pytorch 2.6.0, CUDA 12.4, LAMMPS version “4Feb2025 - Development”.

The single GPU small molecule inference speed experiments were performed on molecules from the SPICE 2 train and test datasets with PubChem substance IDs 135001062 and 103939106, and pentapeptide sequence Trp-Gly-Tyr-Lys-Pro. Results were averaged over three trials performed in separate runs with different seeds for the initial velocities. The initial structures were prepared such that the LAMMPS simulation box was fixed with a buffer of 5 Å from the maximum and minimum x, y, or z positions of the molecule. There were no periodic boundary conditions. The structures were energy-minimized with the conjugate gradient descent algorithm using an `etol` of 10^{-7}, `ftol` of 10^{-7}, `maxiter` 100, and `maxeval` 1000. The velocities were initialized at 300 K and the system was run for 20000 steps in the NVT ensemble with fix momentum applied every 10 steps to remove linear and angular momentum. The timestep was set to 0.5 fs. The momentum and thermostat fixes were removed, and the per-step evaluation time was measured over a subsequent 20000 step NVE MD simulation. We note that the 100-atom pentapeptide molecule is a charged molecule. This breaks the neutral-system assumption of the model but does not affect the inference speed.

The single rank water inference experiments were performed on systems ranging in size from 24 to 5184 atoms with all three models and across three GPU architectures, using the Frontier AMD MI250X GPUs, the NERSC Perlmutter NVIDIA A100 (80 GB) GPUs, and the Harvard FASRC NVIDIA H100 GPUs. The water boxes were fixed at a density of 1 $\mathrm{g/cm^3}$, with the temperature held at 300K with a Langevin thermostat. The measurements were performed with a timestep of 1 fs.

The strong scaling experiments on the DHFR and cellulose systems from the Amber20 benchmark were performed on the Frontier and Perlmutter supercomputers using the same software setups as described in the preceding paragraphs. On Perlmutter, the 40GB A100s were requested. The systems were energy-minimized with the conjugate gradient descent algorithm using an `etol` of 10^{-7}, `ftol` of 10^{-7}, `maxiter` 100, and `maxeval` 1000. The timestep was set to 0.5 fs. Velocities were initialized at 300 K and the system was equilibrated in the NVT ensemble for 1000 steps or a maximum of three minutes. The performance was measured with the profiling done by LAMMPS over a subsequent 1000 step or maximum three-minute molecular dynamics simulation.

IV. ANALYSIS OF DDP STRATEGY: NUMERICS

We analyze the potential for numerical deviations in using the custom DDP scheme over standard PyTorch DDP through empirical verification that the numerical deviations that exist are comparable to the numerical deviations associated with the intrinsic nondeterminism of GPU execution and floating-point rounding effects. We performed experiments comparing the training loss trajectory across different training runs using 64 MPI ranks (8 nodes of 8 AMD MI250X GCDs) to investigate the numerical deviations between different DDP configurations, including

- custom DDP without compilation (baseline),
- standard DDP (which can only be performed without compilation in our framework),
- custom DDP with compilation,
- custom DDP without compilation, using the same seed as the baseline, and
- custom DDP without compilation, using a different seed from the baseline.

The results to follow are presented with reference to the "custom DDP without compilation" baseline. These experiments were performed using ROCm 6.4, PyTorch version 2.9.1, `nequip` version 0.7.0, and `allegro` version 0.4.0.

As shown in Fig. 1 and Table 3, we observe that the loss trajectories monitored every 100 steps over the first four epochs (about 16,100 training steps) are generally similar across different training configurations. We make three key observations.

1. The numerical deviations between custom DDP without compilation (the baseline) and standard DDP are comparable to the deviations observed when rerunning the baseline configuration twice with the same seed. These differences likely reflect the intrinsic nondeterminism of GPU execution and floating-point rounding effects.
2. Compared to the smaller deviations in observation 1, the numerical deviation between runs with different seeds under custom DDP without compilation are much larger. These deviations reflect the more drastic change in terms of the loss trajectory since the model weights were initialized differently in weight space.

3. Compared to the smaller deviations in observation 1, the numerical deviation between the uncompiled baseline and custom DDP with compilation is notably larger as well. These deviations reflect how kernel fusion and other compiler optimizations can change the numerics of model operations.

Table 3. Train loss differences across the first four epochs of training the medium model, logged every 100 steps, on 8 nodes of 8 AMD M250X GCDs for different training settings. All differences are with respect to a run using the custom DDP implementation without compilation. All runs do not use model compilation except when specifically noted.

Custom DDP compared against:	Train Step Loss MAE	Train Step Loss RMSE
Standard DDP	0.00073	0.00156
Custom DDP with compilation	0.00280	0.0264
Custom DDP Rerun	0.00079	0.00175
Custom DDP New Seed	0.00410	0.01569

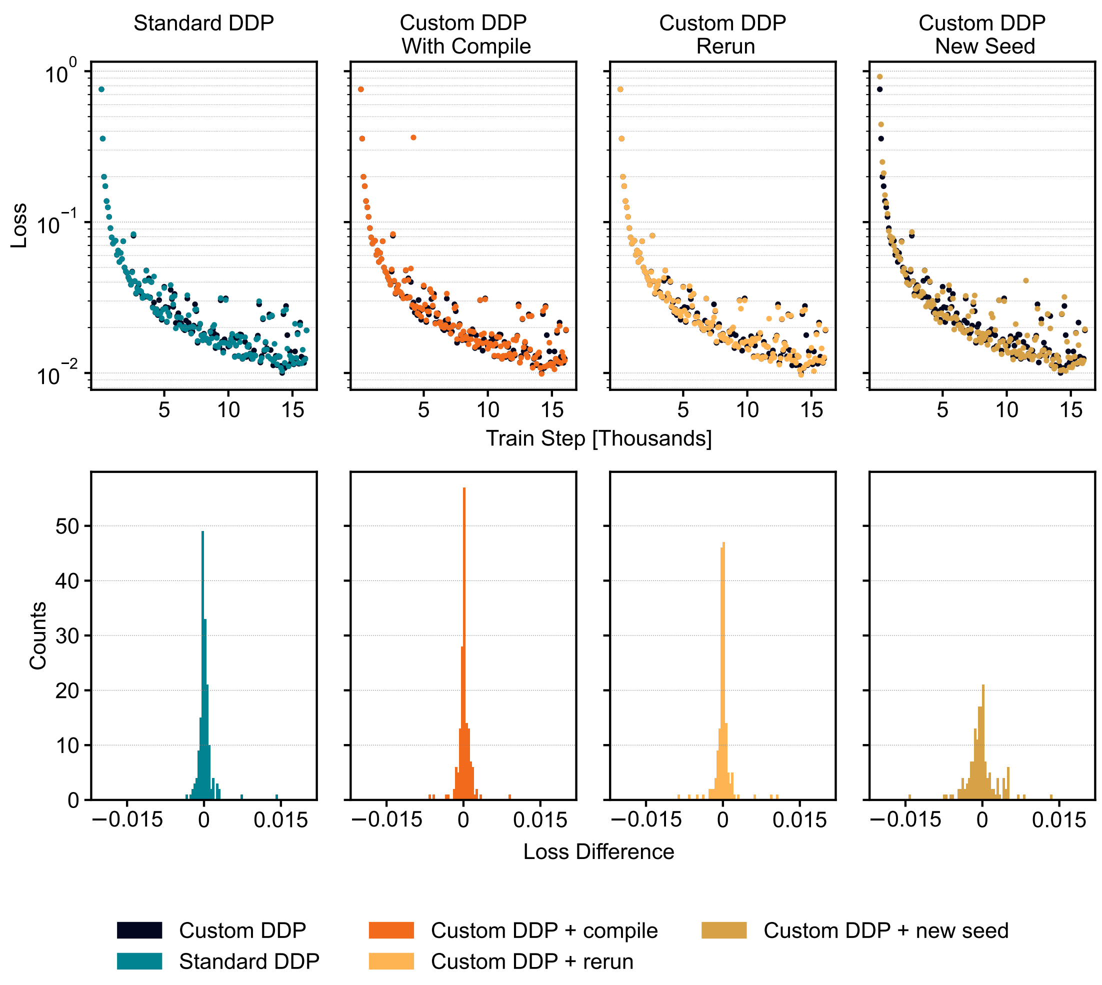

Figure 1. **Training step loss differences using various training configurations and DDP implementations with an Allegro model using the medium model's hyperparameters.** From left to right, the leftmost column compares against PyTorch Lightning's DDPStrategy, next is a comparison against our custom DDP strategy and model compilation, followed by our custom DDP strategy run a second time, and finally our custom DDP strategy with a different model initialization seed. All differences are presented with respect to a run using the custom DDP implementation without model compilation, this reference run is plotted in every column for comparison. All runs do not use model compilation except when specifically noted. For clarity, we fix the histogram boundaries at ±0.020, which excludes one point in the "Custom DDP With Compile" column, and five points in the "Custom DDP New Seed" column.

V. ANALYSIS OF DDP STRATEGY: SPEED

To assess whether omitting gradient bucketing could become a bottleneck for very large models, we conducted additional experiments using both uncompiled and compiled configurations. The uncompiled runs allow a direct comparison between the standard and custom DDP implementations to isolate the impact of communication behavior, while the compiled run quantifies the overall performance benefit of compilation when combined with the custom DDP setup.

We trained message-passing NequIP models on MPTrj [11] using OpenEquivariance tensor-product kernels [12] and TensorFloat32 precision on two nodes of four A100 (80 GB) GPUs (eight GPUs total), with a per-GPU batch size of 80 corresponding to an effective batch size of 640 atomic configurations. These experiments were performed using:

- PyTorch version 2.7.1,
- `nequip` version 0.15.0, and
- `openequivariance` commit `dbc26dfa388218307fc3cae6f74d326e9b101083`.

To probe the scaling of gradient communication in a controlled way, we deliberately increased the dimensionality of the atom type embedding from 32 to 128. Each chemical element (89 in total) has its own learned embedding vector, so increasing the embedding dimensionality directly increases the number of trainable parameters associated with these weights. This change raised the total parameter count from 32.1 M to 114 M while leaving the computational workload per step nearly unchanged. The use of embedding lookups for atom type features means that the computational cost depends only on the number of atoms processed per batch, not on the total number of stored embedding vectors. As a result, the modification selectively amplifies the amount of gradient data exchanged between GPUs without affecting the core compute cost, providing a more controlled way to assess communication overhead. For context, the largest model on the Matbench Discovery benchmark contains roughly 80–90 M parameters, so our tests reach and slightly exceed that scale.

Table 4 summarizes the results for both uncompiled and compiled configurations. Comparing the uncompiled runs, using the custom DDP implementation without gradient bucketing slightly increases epoch time relative to the standard overlapping DDP, as expected from the lack of communication–computation overlap. When compilation is enabled, the overall training time is reduced by about 8 % relative to the baseline, indicating that the performance gains from TorchInductor's kernel fusion outweigh the modest communication overhead introduced by the custom

DDP design. The improvement remains modest for NequIP because the relatively large batch sizes used for efficient GPU throughput make the tensor-product convolution the dominant computational bottleneck. OpenEquivariance already provides optimized kernels for this operation, and compilation primarily fuses smaller miscellaneous operations that are not performance limiting.

These results indicate that the communication overhead introduced by the custom DDP is minor relative to the total computational cost, and that the overall setup remains efficient even for models approaching the largest practical scales. Should communication become a bottleneck for substantially larger models, gradient bucketing or overlapping communication could be reintroduced.

Table 4. Walltime per training epoch for three configurations: (1) standard DDP without compilation (baseline), (2) custom DDP without compilation, and (3) custom DDP with compilation, evaluated for large NequIP models with 32.1 M and 114 M parameters. The relative percentage differences in walltime for configurations (2) and (3) are reported in parentheses, relative to the baseline (1).

No. of Parameters	32.1 M	114 M
Standard DDP without compilation	35 mins 58 seconds	40 mins 2 seconds
Custom DDP without compilation	36 mins 14 seconds (+ 0.7 %)	42 mins 5 seconds (+ 5 %)
Custom DDP with compilation	32 mins 52 seconds (- 8.5 %)	36 mins 56 seconds (- 7.5 %)

[1] J. F. Ziegler and J. P. Biersack, in *Treatise on heavy-ion science: volume 6: astrophysics, chemistry, and condensed matter* (Springer, 1985) pp. 93–129.

[2] I. Loshchilov and F. Hutter, arXiv preprint arXiv:1711.05101 (2017).

[3] P. Eastman, P. K. Behara, D. L. Dotson, R. Galvelis, J. E. Herr, J. T. Horton, Y. Mao, J. D. Chodera, B. P. Pritchard, Y. Wang, *et al.*, Scientific Data **10**, 11 (2023).

[4] B. K. Rai, V. Sresht, Q. Yang, R. Unwalla, M. Tu, A. M. Mathiowetz, and G. A. Bakken, Journal of Chemical Information and Modeling **62**, 785 (2022).

[5] P. Eastman, B. P. Pritchard, J. D. Chodera, and T. E. Markland, Journal of chemical theory and computation **20**, 8583 (2024).

[6] D. P. Kovács, J. H. Moore, N. J. Browning, I. Batatia, J. T. Horton, Y. Pu, V. Kapil, W. C. Witt, I.-B. Magdău, D. J. Cole, *et al.*, Journal of the American Chemical Society (2025).

[7] G. Simeon and G. De Fabritiis, Advances in Neural Information Processing Systems **36**, 37334 (2023).

[8] I. Batatia, D. P. Kovacs, G. Simm, C. Ortner, and G. Csányi, Advances in Neural Information Processing Systems **35**, 11423 (2022).

[9] A. Musaelian, S. Batzner, A. Johansson, L. Sun, C. J. Owen, M. Kornbluth, and B. Kozinsky, Nature Communications **14**, 579 (2023).

[10] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, Comput. Phys. Commun. **271**, 108171 (2022).

[11] B. Deng, P. Zhong, K. Jun, J. Riebesell, K. Han, C. J. Bartel, and G. Ceder, Nature Machine Intelligence **5**, 1031 (2023).

[12] V. Bharadwaj, A. Glover, A. Buluç, and J. Demmel, in *2025 Proceedings of the Conference on Applied and Computational Discrete Algorithms (ACDA)* (SIAM, 2025) pp. 32–46.