Supplementary Information

Modeled Forces

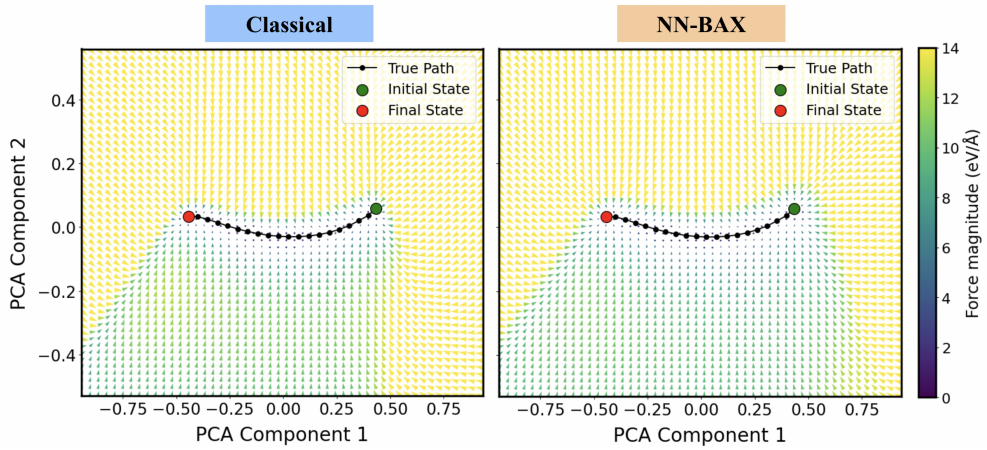

Fig. 7 Modeled Forces for LJ_7 system with 0 minima. The forces are 21-dimensional, so the first two principal components are displayed. The left "classical" plot shows the true forces. The right plot shows the modeled forces from NN-BAX, at the final converged iteration.

Computational Discussion

In order to calculate the time per DFT calculation NN-BAX would require to be faster than classical NEB, we run another test on the LJ_{38} 0 minima path to estimate the computational overhead of BAX. NN-BAX took 9.3 hours to run, for 30 BAX iterations, with 200 NEB steps, and with each model trained for 50 epochs. All of our code was run on a single NVIDIA Tesla A100 GPU. Since calls to the LJ potential are practically instant, this is a good approximation for solely the computational overhead cost of NN-BAX. Thus for a setting in which DFT is being used, for NN-BAX to achieve a wall-clock speedup relative to Classical NEB, each DFT simulation would have to be longer than 25.4 seconds. However our setup involves minimal implementation overhead, and can greatly be improved.

The two primary components of the overhead are model inference during NEB, and training the model on sampled points. In our implementation these components took similar times, with training taking 48% of the time. Due to the default ASE NEB implementation, images are evaluated sequentially in our NEB runs. Taking advantage of GPU parallelism for model inference would cut the NEB run time down by a factor of 20. For the training component, we retrain the model from scratch at each iteration, with a training budget of 50 epochs. Strategies to reduce this include using a warm start, where we may learn in some fraction of those iterations, like 10 epochs. Taking into account a potential factor of 20 speedup for model inference during NEB and a

factor of 5 speedup for training results in a time of 3.1 seconds per DFT simulation, in order to achieve a speedup relative to classical NEB. We note that we can achieve an even greater speedup by using a smaller EquiformerV2 model (see below).

Transition State Predictions

Minima of potential energy surfaces have a zero gradient in all dimensions. Transition states are a minimum in all dimensions except for one, and correspond to a maximum of potential energy along a reaction path. In three dimensions for example, a saddle point satisfies this constraint. A transition state is often of particular interest because it dictates the activation energy of the reaction and corresponds to bonds breaking and new bonds forming [4]. Transition states are helpful in understanding the kinetic properties of chemical systems, such as reaction rate. In Figure 8 we display the predicted structures for the highest energy transition state of each reaction. We observe the NN-BAX structures qualitatively match the classical NEB structures.

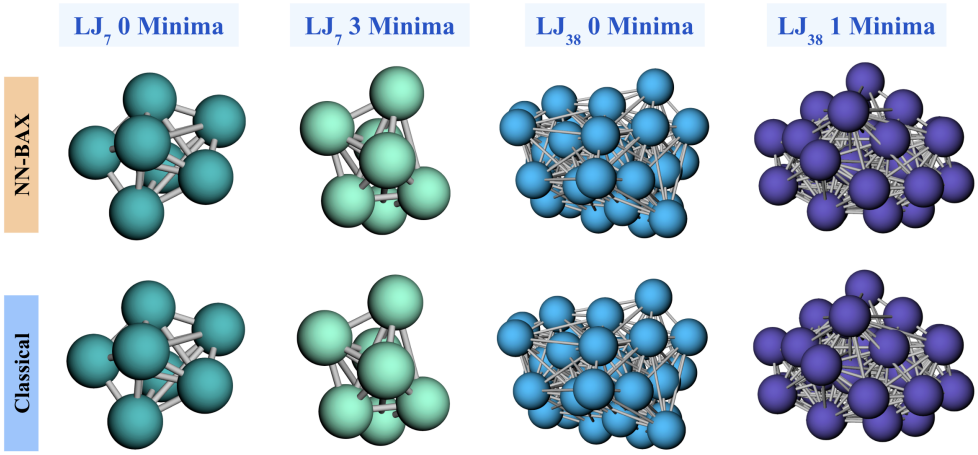

Fig. 8 Transition state predictions. The NN-BAX predictions and the classical ground truth atomic arrangements are displayed.

NN-BAX Convergence

In Figure 9 the MAE_i convergence parameter for the four LJ paths we study is plotted. We observe that it is stable, and reliably falls under the threshold of 0.1 when NN-BAX is outputting the correct path. We note that in the Foundation-BAX study, lower MAE_i thresholds are required. Specifically, we use $\text{MAE}_i = [0.1, 0.1, 0.025, 0.05, 0.025]$ for paths from points A through F respectively, observing that lower convergence thresholds suit the later paths better. We also observe slightly higher patience values of [4, 2, 2, 3, 3] which suit Foundation-BAX better. For the EAM study $\text{MAE}_i = 0.001$ was found to be optimal, likely because the EquiformerV2 model was already more

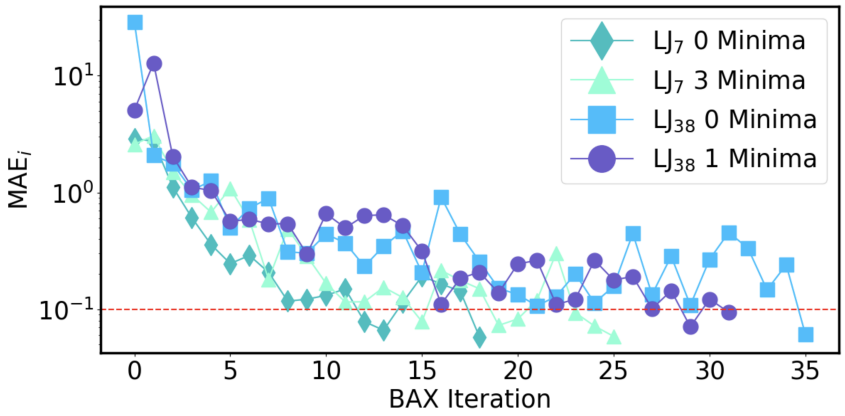

Fig. 9 MAE_i convergence metric versus BAX iteration. The red line denotes the convergence threshold of 0.1

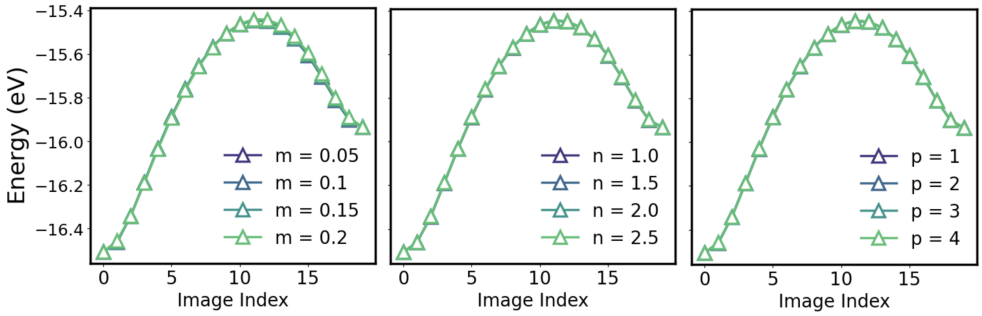

Fig. 10 Probing robustness of NN-BAX convergence hyperparameters for the LJ_7 system with 0 minima. For each plot we vary the one of the default values of $m = 0.1$, $n = 2$, and $p = 2$ and plot the resulting energy profile.

accurate before any fine-tuning. In Figure 10 we comprehensively test the robustness of each of the convergence hyperparameters m, n, and p, observing that NN-BAX returns the correct energy profile for various values of these hyperparameters.

Ablation Study

We perform an ablation study in order to assess the strength of our acquisition strategy (sampling randomly from the algorithm output). We contrast our method with a random sampling method, which samples points around the linear interpolation between the initial and final state. δ. Specifically, the random sampling method involves first randomly picking a image along the linear interpolation, then varying all atomic coordinates according to a uniform distribution $\in [-\delta, +\delta]$. We compare two methods of random sampling, one where we simply set $\delta = 0.2$, and one where we use the true path to compute some optimal δ, given by the distance the transition state is from the linear interpolation. We run the random sampling method for 125 BAX iterations

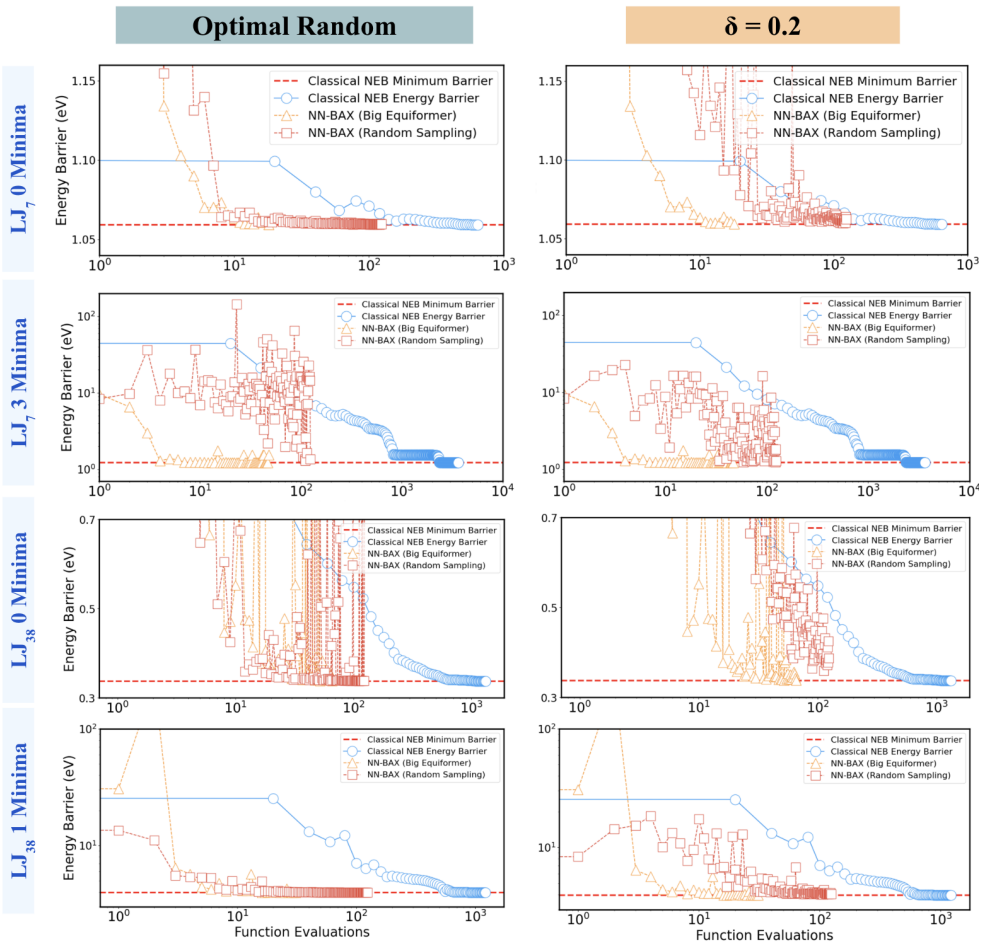

Fig. 11 Ablation study with acquisition method, comparing NN-BAX against random sampling method, keeping the fine-tuning procedure the same. We analyze runs with an optimal sampling variation δ and with $\delta = 0.2$.

and plot the results. We observe that for $\delta = 0.2$, the random sampling method consistently performs worse, roughly an order of magnitude. For the optimal sampling method we observe better performance, with the performance being on par with NN-BAX for some paths, but still being slower and unable to find the true path for others. However this optimal sampling method is unrealistic as it assumes knowledge of the true path transition state, which is not known prior to running NEB. The results are shown in Figure 11.

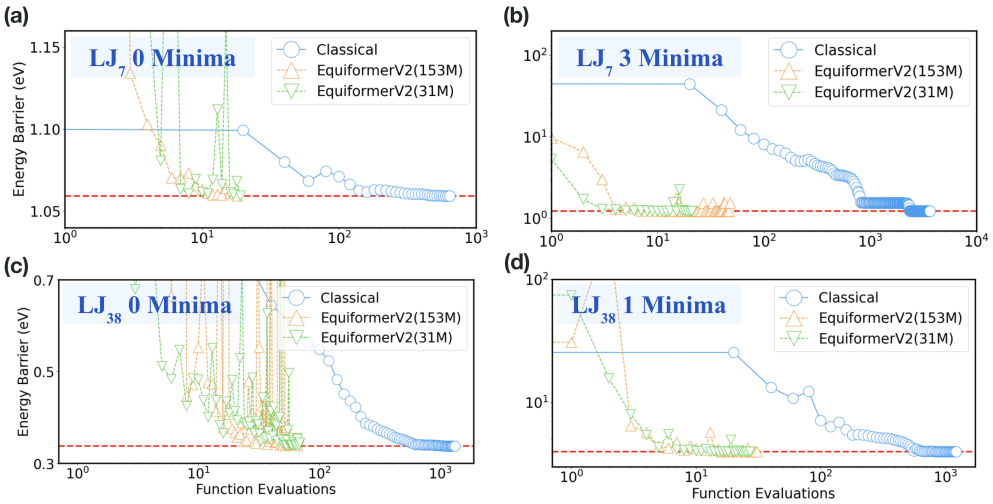

Fig. 12 Smaller EquiformerV2 comparison. The EquiformerV2 model with 31M parameters is compared against the one with 153M parameters.

Smaller EquiformerV2

In Figure 12 we compare a smaller EquiformerV2 model (31M params) against the model we used (153M params). We observe generally similar performance. Furthermore, the smaller EquiformerV2 has a faster inference and training time. Specifically, we observe a 1.6× speedup factor for training, and a 3.8× speedup factor for inference.