Numerical study on small cell training protocol for Pb system

YM Chan

1 Introduction

This report summarizes work conducted in Fall 2025 on the small cell training protocol, an active learning approach for developing machine-learned (ML) interatomic potential models for large atomic systems proposed by Meng et al. [1]. The objective is to train models that approximate density functional theory calculations by predicting atomic forces and system energies using reference DFT data.

The small cell training protocol addresses the high computational cost of generating DFT data for large systems. Training begins with small atomic cells and progressively increases system size until the target configuration is reached. This approach is effective because atomic behavior in large systems is largely governed by local interactions, allowing models trained on small cells to generalize to larger structures.

In this study, the small cell training protocol was implemented for a lead-based system. The following milestones were achieved:

1. Optimization of MLIP-3 hyperparameters for lead
2. Development of Python scripts to automate active learning between LAMMPS, MLIP-3, and VASP
3. Modification of the MLIP-3 source code to track the extrapolation factor γ
4. Execution of active learning on the [1,1,1] lead unit cell
5. Evaluation of model performance across iterations
6. Initial attempts to scale to the [1,1,2] system on the Pitagora HPC platform

Table 1: Nomenclature/ abbreviations

Name	Description
LAMMPS	Open-source Molecular Dynamic simulation software [2] `https://www.lammps.org/#gsc.tab=0`
VASP	Ab-initio Density Functional Theory simulation software [3] `https://www.vasp.at/`
MLIP-3	Open-source Machine learning model for interatomic potentials [4] `https://gitlab.com/ashapeev/mlip-3`
ML	Machine learning

Suggestions for future work:

1. Completing the implementation of small cell training protocol on HPC Pitagora and train on larger systems.
2. Perform more rigourous numerical study of error convergence at larger unit cells
3. Convergence and comparison study of MLIP-3 with other ML models, e.g. Deepmd.

2 MLIP-3 model hyperparameter selection

MLIP-3 is a ML method developed by Podryabinkin et al. based on a basis function set comprising of different orders of moment tensor products of the atom positions. The main model hyperparameter is the "MTP level", a measure of the number of basis functions to be included in the model. The higher the number the more higher order moments are included in the basis function set.

2.1 Training data description

In order to choose the optimal MTP level for the Pb system, I trained different MTP levels on the same training data set generated by Qiu Guo. The training data set consist of 4 different simulations, comprising of [2,2,2] and [3,3,3] unit cells, Pb unit cell = 4 atoms, so 32 and 108 atoms respectively. Once trained, the models are tested on 2 previously unseen datasets (1) Pb atoms in equilibrium configuration and (2) Pb atoms in non-equilibrium configurations.

Breakdown of training data

Table 2: Breakdown of training data distribution

Name	# configs
eq-32	1020
eq-108	1020
neq-32	1980
neq-108	1980

Results of MTP level versus model accuracy. Model accuracy is evaluated using a set of test configurations that are not included in the training data. For each test configuration, reference atomic forces

$$F_{\mathrm{ref}} = [F_x, F_y, F_z]$$

and total energy E_{ref} are computed using VASP. The trained MLIP-3 model is then used to predict the corresponding forces F_{ML} and energy E_{ML}. Model error is quantified by comparing the predicted values against the reference results. We define the error of the force vector as the root mean square (RMS) over all coordinates for every atom in the lattice:

$$\epsilon_F = \sqrt{}(\frac{1}{I}\sum_i ||F_{ref,i} - F_{ML,i}||_2) \tag{1}$$

and the error of the lattice energy is simply the absolute error:

$$\epsilon_E = |E_{ref} - E_{ML}| \tag{2}$$

Across a set of lattices, we take the average of error metrics

$$\bar{\epsilon}_F = \mathrm{Mean}(\epsilon_F^1, \epsilon_F^2, ...) \tag{3}$$

$$\bar{\epsilon}_E = \mathrm{Mean}(\epsilon_E^1, \epsilon_E^2, ...) \tag{4}$$

We decided to select MTP level=18 which shows stagnating accuracy.

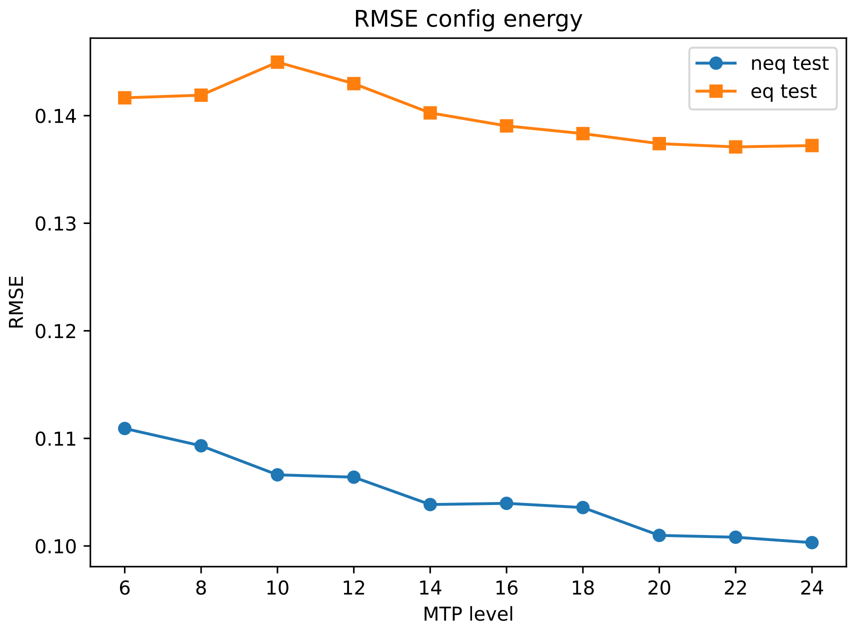

Figure 1: Convergence of RMSE energy with MTP level.

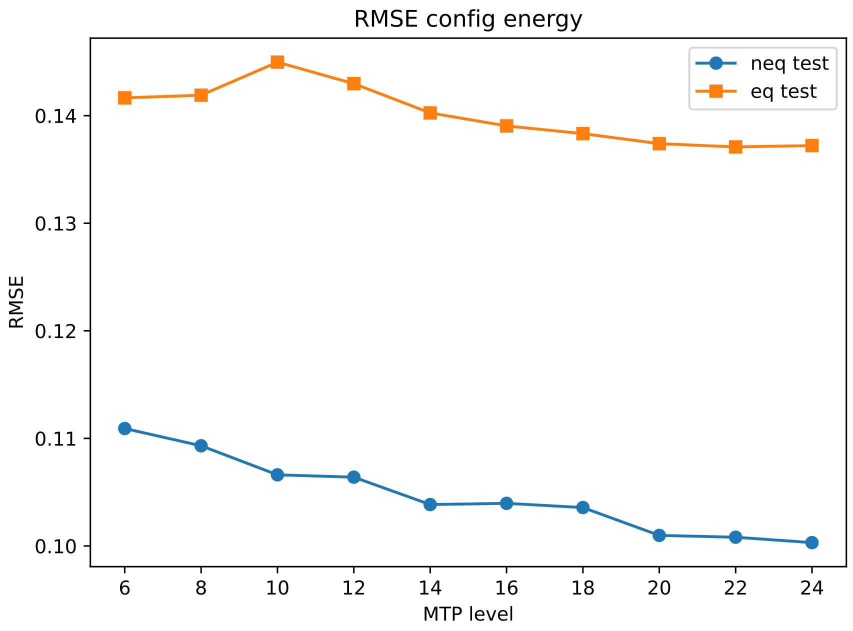

Figure 2: Convergence of RMSE forces with MTP level.

Summary of training times

Table 3: CPU-training times for various MTP levels.

MTP level	CPU-time
6	56 mins
12	6.7 hours
18	47 hours
24	496 hours
28	$>$ 48 days [killed]

3 Small cell training implementation

3.1 Description of MLIP-3/LAMMPS interface

A newer version, MLIP-4, was available but caused errors with the LAMMPS interface, so we continued using MLIP-3. MLIP-3 integrates with LAMMPS, and the input line specifies the trained model location along with additional parameters. (`threshold_save`, `threshold_break`,...):

```
pair_style mlip      load_from=<pot.almtp>
↪  extrapolation_control=true threshold_save=0.1
↪  threshold_break=10
↪  extrapolation_control:save_extrapolative_to=<preselected.cfg>
```

Description of MLIP-3 interface with LAMMPS

Table 4: Interface options between MLIP-3 and LAMMPS

Name	Description
`load_from:<file>`	Path to trained MLIP-3 model
`extrapolation_control`	Boolean. See below for logic control.
`threshold_save`	Value of γ_{save}. See below.
`threshold_break`	Value of γ_{break}. See below.
`save_extrapolative_to`	Path to save atomic configurations from LAMMPS with $\gamma > \gamma_{save}$ as a text file.

The MLIP-3 authors introduce an extrapolation factor γ, which is a scalar value quantifying how different a queried atomic configuration to the model is from the set of training data during model training. The value of $\gamma = 1$ is obtained when a queried atomic configuration matches exactly one of the training configurations. The authors suggest a value of $2 < \gamma < 10$ as "safe-extrapolation" and $\gamma > 10$ as "unsafe-extrapolation".

During a coupled LAMMPS-MLIP3 simulation, MLIP-3 will monitor the value of the extrapolation factor based on queried atomic configurations (γ_q). If γ_q exceeds the user-set parameters [γ_{save}, γ_{break}], MLIP-3 will perform the corresponding action:

- $\gamma_q > \gamma_{save}$: MLIP-3 continues LAMMPS-MLIP3 simulation, writing out the configuration to `preselected.cfg`.
- $\gamma_q > \gamma_{break}$: MLIP-3 ends LAMMPS-MLIP3 simulation after writing out the configuration to `preselected.cfg`.

Table 5: Podryabinkin et al. suggested values of [γ_{save}, γ_{break}] [4]

Parameter	Value
γ_{save}	2.1
γ_{break}	10.0

The small cell training protocol builds upon MLIP-3 extrapolation logic control to create new datasets using ab-initio code to simulate the configurations in `preselected.cfg`. We define the set of extrapolated configurations in `preselected.cfg` as $\mathcal{D}_{save}$ and the set of configurations to train the model as $\mathcal{D}$. For a given lattice configuration (e.g [1,1,2] unit cell) the active learning protocol is as follows

Algorithm 1 Active learning to reduce extrapolation grade

```
1: procedure ACTIVE LEARNING($L, \mathcal{D}$)
2:     Train model on $\mathcal{D}$
3:     while $\max \gamma > \gamma_{break}$ do
4:         WRITE LAMMPS INPUT(L)
5:         LAMMPS/MLIP-3 simulation(s) with extrapolation logic control to obtain $\mathcal{D}_{save}$
6:         for $C_i \in \mathcal{D}_{save}$ do
7:             Run VASP simulation on $C_i$
8:         end for
9:         Append new training configurations to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{save}$
10:        Retrain model on $\mathcal{D}$
11:    end while
12:    return $\mathcal{D}$
13: end procedure
```

The small cell training protocol applies active learning on progressively larger lattices. Let $\mathcal{L} = \{L_1, L_2, \dots\}$ with L_1 smaller than L_2. For practical simulations using LAMMPS/MLIP-3, temperature and pressure ranges $(\mathcal{T}, \mathcal{P})$ must also be specified. Following Meng et al., temperatures and pressures are sampled from a grid, and atomic positions are slightly perturbed from their lattice sites with length parameter a. The procedure is defined as:

Algorithm 2 Write LAMMPS input

```
1: procedure WRITE LAMMPS INPUT($L, (\mathcal{T}, \mathcal{P}), a$)
2:     for $(T, P) \in (\mathcal{T}, \mathcal{P})$ do
3:         for atom $i$ in lattice do
4:             Calculate "default" atom position in lattice $x_i, y_i, z_i \leftarrow f(L, a)$
5:             Generate random perturbations in coordinates $(\delta_x, \delta_y, \delta_z)$
6:             $x_i, y_i, z_i \leftarrow (x_i, y_i, z_i) + (\delta_x, \delta_y, \delta_z)$
7:         end for
8:         Write out LAMMPS file with atomic positions, $T$ and $P$
9:     end for
10: end procedure
```

The small cell protocol algorithm is therefore:

Algorithm 3 Small cell training protocol

```
1: procedure SMALL CELL PROTOCOL
2:     Dataset with initial training conditions $\mathcal{D}$
3:     for $L_i \in \mathcal{L}$ do
4:         ACTIVE LEARNING($L_i, \mathcal{D}$)
5:         $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{save}$
6:     end for
7: end procedure
```

3.2 Questions on active learning mechanism

To better understand the process of the coupled active learning mechanism, we decided to focus on the performance of Algorithm (1), answering several key questions

1. γ is used to determine if a simulation is "good". How does it actually correlate with error (i.e. the difference between ML model and VASP calculation)?
2. For a run of the active learning algorithm (Alg. 1) at a set lattice configuration at constant temperature and pressure, how many iterations i.e. step (3-11) are needed?
3. As $\mathcal{D}$ grows, two things are expected: (1) Max and average γ should decrease (2) actual error should decrease. Do we observe this decrease and how fast do they decrease?

3.3 Answers to active learning questions

3.3.1 γ is used to determine if a simulation is "good". How does it actually correlate with error (i.e. the difference between ML model and VASP calculation)?

The trained MLIP-3 model (MTP level = 18) was tested using a LAMMPS/MLIP-3 simulation. Configurations were generated with a wide range of γ values by setting a lower saving threshold, $\gamma_{\text{save}} = 0.1$, and configurations with γ uniformly distributed between 0.1 and 10 were then selected. In practice, however, the majority ($> 99\%$) of γ values are concentrated between 0.1 and 2.1, as shown by the distributions in the later sections.

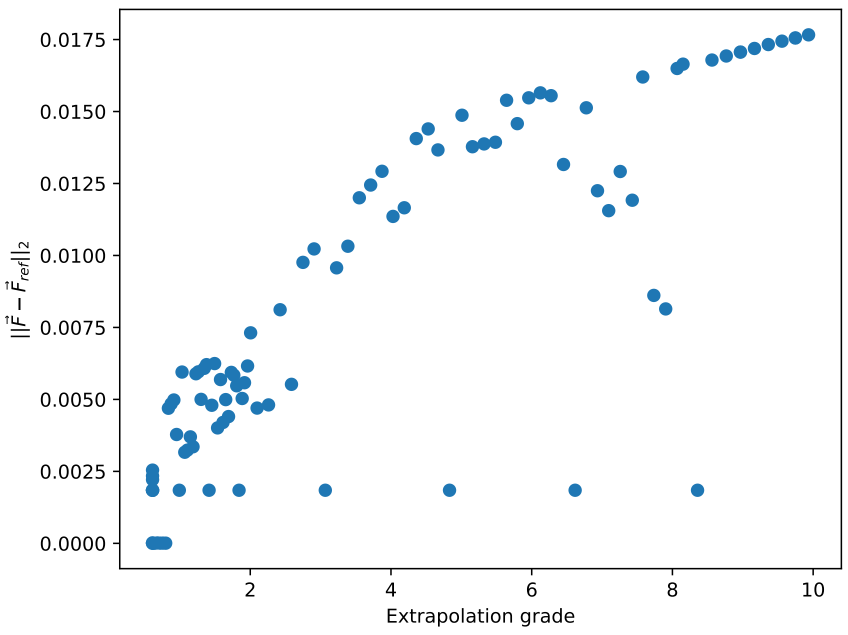

Figure 3: Relationship between ϵ_F (see Eqn 1) and γ

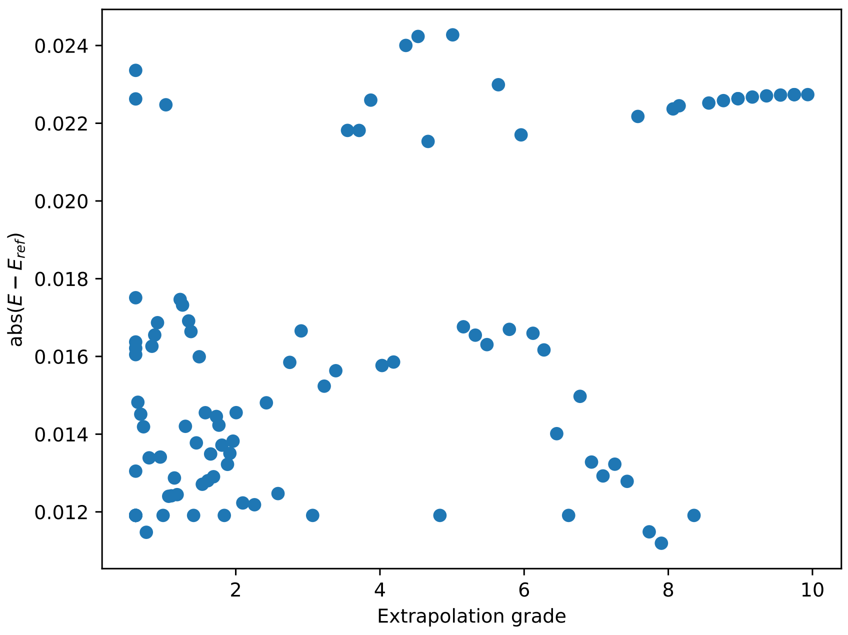

Figure 4: Relationship between ϵ_E (see Eqn 2) and γ

The correlation of ϵ_F is much stronger than that of ϵ_E. This may be because the energy reflects the entire lattice, so small random perturbations in atomic configurations tend to produce errors that partially cancel. In contrast, force errors are summed over all atoms, making them more sensitive to differences between training and testing configurations.

Conclusion: γ is strongly correlated with predicted force error, so reducing γ will most likely lead to lower predicted force error.

3.4 For a run of the active learning algorithm (Alg. 1) at a set lattice configuration at constant temperature and pressure, how many iterations i.e. step (3-11) are needed?

A python script was created to run the active learning algorithm (Alg 1). A [1,1,1] unit cell was used with set temperature and pressure at 300 K and 1 bar using the "fix npt/body command" in LAMMPS. The active learning iterations were carried out for 60 iterations with $\gamma_{break} = 10.0$. MLIP-3 $\gamma_{save} = 0.1$ was used to monitor the distributions of γ and to save almost all configuration queries between LAMMPS and MLIP-3, but only configurartions with $\gamma > 2.1$ were sent for VASP calculations to emulate $\gamma_{save} = 2.1$.

The figure below shows the number of VASP calculations per active learning iteration. The total number of VASP calculations over 60 iterations is 20308.

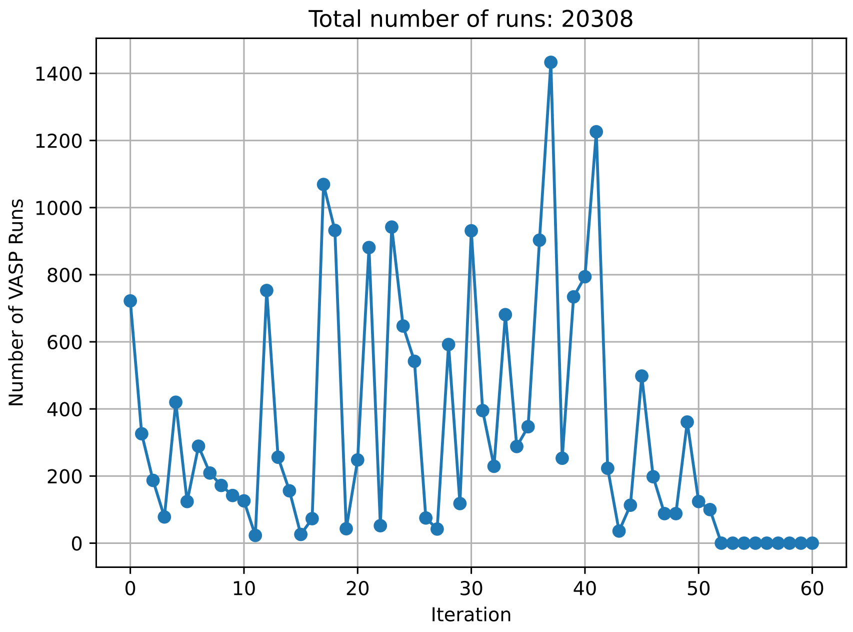

Figure 5: Number of VASP calculations against active learning iteration

Notice that there is a wide variation in the number of VASP calculations per iteration, ranging from 100 to 1400 VASP calculations. This variation makes it more difficult to implement this system in HPC framework (see below for recommendations).At iteration 52, maximum γ values are less than γ_{break} so the LAMMPS simulation is not prematurely terminated by MLIP-3.

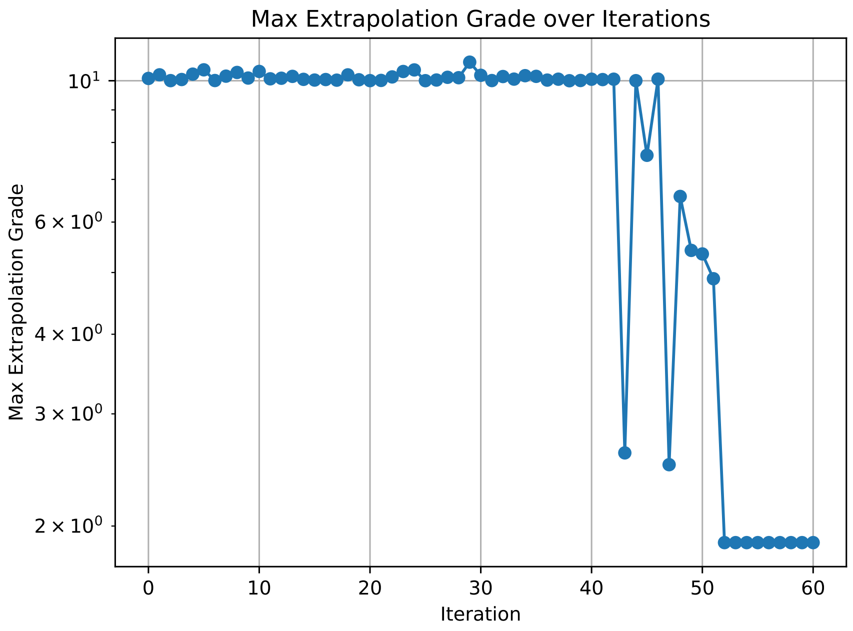

Figure 6: Maximum extrapolation grade against active learning iteration with $\gamma_{break} = 10.0$

3.5 As $\mathcal{D}$ grows, two things are expected: (1) Max and average γ should decrease (2) actual error should decrease. Do we observe this decrease and how fast do they decrease?

3.5.1 (1) Max and average γ should decrease

Max γ follows an unpredictable trend as shown in the previous section. In the following figure we show the mean and standard deviation of γ over active learning iterations, which show the mean decreasing from around 1.0 to around 0.5.

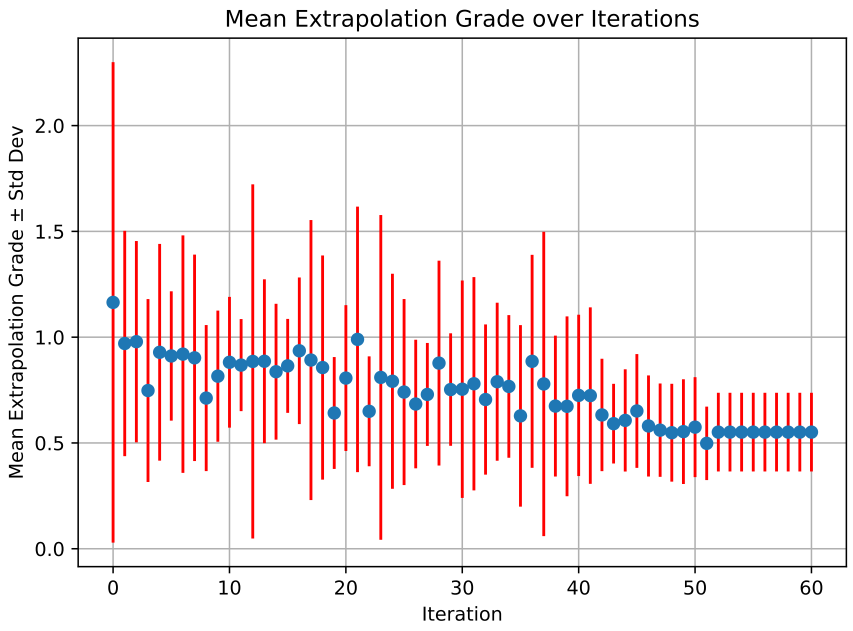

Figure 7: Mean and standard deviation of extrapolation grade against active learning iteration

The following figures show the histogram distribution of γ for queries between LAMMPS and MLIP-3 during a coupled simulation at an early iteration (iter = 1), mid-point (iter = 30), and final iteration (iter = 55). In all cases, over 99.9% of γ values are below 2.0, indicating that the model operates mostly within the interpolation or "safe-extrapolation" regime relative to the training data. At iterations 1 and 30, MLIP-3 prematurely terminates the LAMMPS simulations, resulting in ~20000 and ~50000 calls to MLIP-3, respectively. By iteration 55, with no premature termination, the number of calls increases to ~100000.

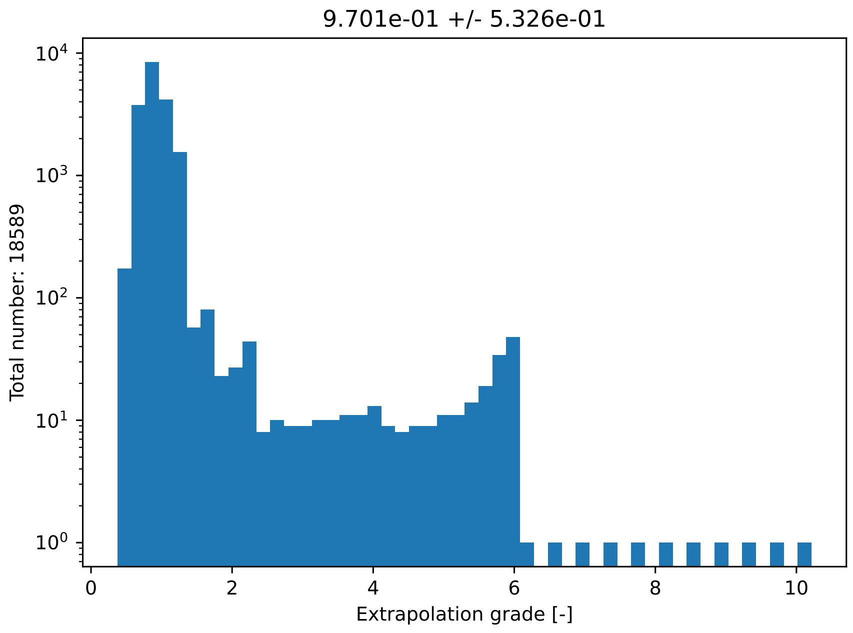

Figure 8: Distribution of γ at iteration 1

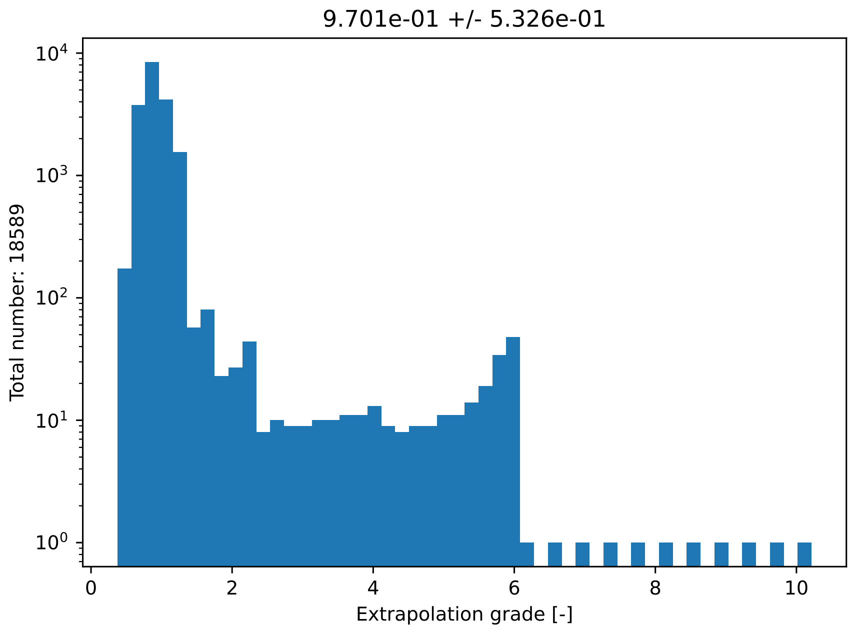

Figure 9: Distribution of γ at iteration 30

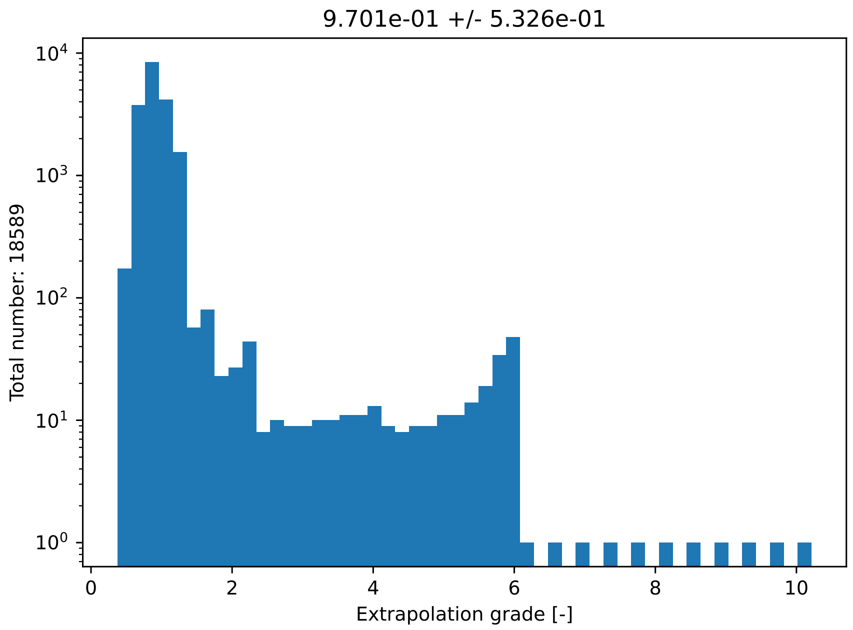

Figure 10: Distribution of γ at iteration 55

Conclusion: Mean γ decreases slowly over active learning iterations but with standard deviation of γ for each active learning iteration being much larger the overall trend. The maximum γ does not show any trend, but rather drops to small values very suddenly. The reason for this is unknown, possibly due to how LAMMPS simulation is carried out and its interaction with MLIP-3.

3.5.2 (2) actual error should decrease

In typical ML convergence studies, the mean prediction error decreases as the number of training points increases independently. In dynamic active learning, this is less straightforward because only configurations with “high” extrapolation factors are added to the training set, providing feedback from model queries. This selective sampling is expected to improve convergence. To examine this effect, 200 random configurations were drawn from all queries between LAMMPS and MLIP-3. These configurations were evaluated with VASP and compared to MLIP-3 predictions of atomic forces and energies, with errors ϵ_F and ϵ_E computed as defined in Equations (1) and (2), respectively.

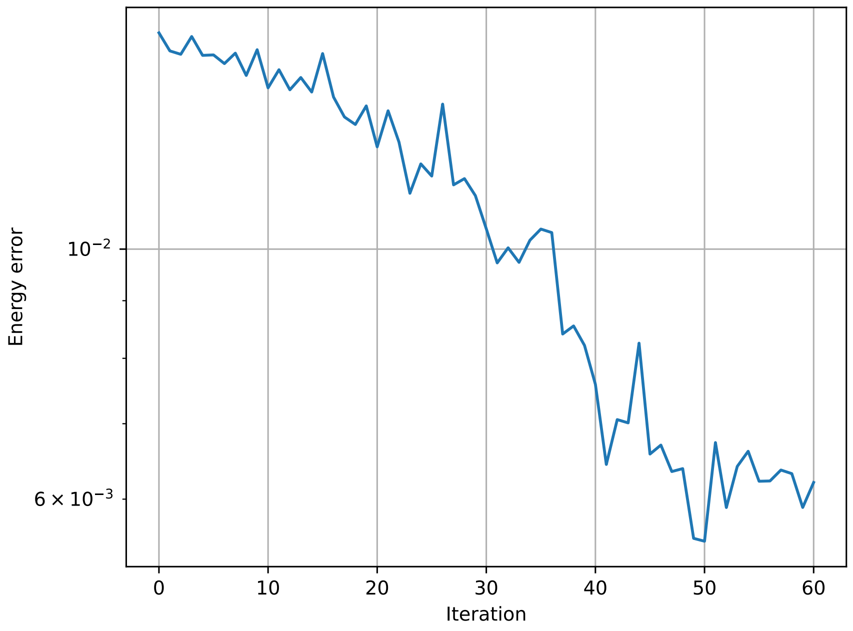

Figure 11: ϵ_E over iterations

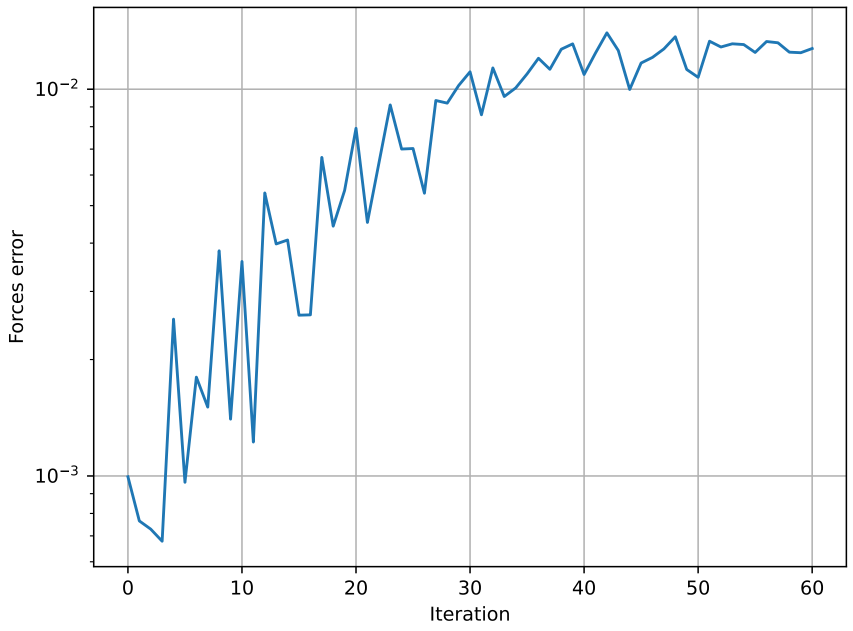

Figure 12: ϵ_F over iterations

The energy convergence plot behaves as expected, with errors decreasing over successive iterations. In contrast, the force error shows an unexpected increase despite the growing dataset, which should provide more information for learning. This anomaly is unlikely due to incorrect active learning implementation, as ϵ_E decreases, but may arise from improper VASP inputs or a potential bug in MLIP-3. Future work is suggested to reproduce and investigate this behavior. In summary, energy prediction errors decrease with active learning iterations, while force prediction errors surprisingly increase.

4 Summary and suggestions for future work

In this work we performed numerical investigation of the active learning step using MLIP-3's extrapolation control logic as one of the key components of the small cell training protocol. We tested on a lead based system with the [1,1,1] unit cell.

- Sensitivity study showed that MTP level=18 is good for lead.
- Total number of VASP calculations was ~20000.
- Max. γ decreases to < 2, bringing it within the range of 'safe extrapolation'.

- Energy error decreases but strangely, force error seems to increase.

Suggestions for future work

1. Completing the implementation of small cell training protocol on HPC Pitagora and train on larger systems.
2. Perform more rigourous numerical study of error convergence at larger unit cells
3. Convergence and comparison study of MLIP-3 with other ML models, e.g. Deepmd.

5 Code development and implementation

In this work, I wrote python files to script execution of algorithms 1 - 3, which are based on the python files of the small cell training group located at `https://github.com/RichardZJM/Small-Cell-MTP-Training` with significant changes. This section documents the codes created, how they should be used and where they are found in the folder "codes". A public github repo has been created for this purpose, found here: `https://github.com/YimengChankth/active-learning-ML-potential-KTH.git`

5.1 codes/active-learning

The main python file to handle the sequential code executions is: `codes/active_learning/main.py` which performs the following code execution (method: `lammps_phase`) for an active learning iteration

1. Create a new directory: `iter/<i>`
2. `generate_lamnps_file`: Write out new LAMMPS input file
3. Run LAMMPS
4. `sample_query_error`: Choose 200 random configurations with $0.1 < \gamma < 10$ and run VASP simulations (folder: `iter/<i>/vasp`) to perform iteration error analysis.
5. `run_new_batch_cfg_with_threshold`: Select all configurations with $2.1 < \gamma < 10$ and run VASP simulations (folder: `iter/<i>/vasp_threshold`) to generate new training data
6. `train_new_MLIP_model`: Append new training data and train new MLIP-3 model, saving the model parameters (file: `iter/<i>/updated_model/pot.almtp`)

5.2 codes/Small-Cell-MTP-Training

This is a forked version of the Small cell MTP training from `https://github.com/RichardZJM/Small-Cell-MTP-Training`. The main difference was to include a way to write FCC configuration, as the original only had BCC configurations specifically in the file `codes/Small-Cell-MTP-Training/SmallCellMTPTraining/io/writer.py`.

```
# look for the following line

def writeMDInput_fcc(fileName: str, jobProperties: dict):
  """Creates a MD Input for fcc e.g. Pb, needing:
  mdProperties = ["latticeParameter", "boxDimensions",
  ↪  "potFile", "temperature", "pressure","elements",
  ↪  "atomicWeights"]
  """
  properties = props.mdProperties
  if checkProperties(properties, jobProperties):
      raise Exception(
          "Properties missing: " + checkProperties(properties,
          ↪  jobProperties)
      )
  num_elem = len(jobProperties["elements"])
  # Calculate the number of unit cells in each dimension
  nx = int(np.ceil(jobProperties["boxDimensions"][0]))
  ny = int(np.ceil(jobProperties["boxDimensions"][1]))
  nz = int(np.ceil(jobProperties["boxDimensions"][2]))
  # Total number of atoms
  num_atoms = 4 * nx * ny * nz  # 4 atoms per unit cell in FCC
...
```

MLIP-3 and MLIP-3 LAMMPS interface are used “as-is”.

1. MLIP-3: `https://gitlab.com/ashapeev/mlip-3`
2. MLIP-3 LAMMPS interface `https://gitlab.com/ivannovikov/interface-lammps-mlip-3/`

5.3 codes/MTPlevel_sensitivity

This folder contains the python script used to test different MTP levels. See `main.py`.

The specific portion is

```
def train_mlip_model():
  # initial test for MLIP model and varying the levels

  MTP_levels = [8, 10, 14, 16, 20, 22, 28]

  # train phase
```

```
for mtp_count, mtp_value in enumerate(MTP_levels):
    mlip_model =
    ↪  mlipModel(savedir=f'mlip_ym_all_train/mtp_{mtp_value}')
    outcar_files, _ = get_all_training_outcars()
    mlip_model.training_outcars = outcar_files
    mlip_model.MTP_level = mtp_value
    shutil.copy(f'mlip_ym_all_train/mtp_6/train.cfg',
    ↪  mlip_model.savedir)
    mlip_model.train(mpi=16)

pass
```

5.4 Access to training data

As github is not meant to store large amounts of training data, this section points to where the training data may be found in Southpole.

5.4.1 MTP level sensitivity training and testing data

The training and testing data used for the MTP level sensitivity study contains equilibrium and non-equilibrium data sets for 32 and 108 atoms are currently with Qiu Guo. On Southpole the last updated location is at:

```
/home/qiuguo/pb-dpmodel/OUTCARS/
|-- 108eqOUTCARs/<outcar files>
|-- 108neqOUTCARs/<outcar files>
|-- 32eqOUTCARs/<outcar files>
|-- 32neqOUTCARs/<outcar files>
|-- pb_testdata/
|   |-- 108/
|   |   |-- eq/<outcar files>
|   |   `-- neq/<outcar files>
|   `-- 32/
|       |-- eq/<outcar files>
|       `-- neq/<outcar files>
```

The OUTCARS used for training are: [108eqOUTCARs, 108neqOUTCARs, 32eqOUTCARs, 32neqOUTCARs] and `pb_testdata` contains the testing data.

5.4.2 Active learning iteration data

At each active learning iteration from $i = \{0, 1, ...60\}$, the training data set is saved and stored in:

```
  /home/qiuguo/pb-dpmodel/mlip3_ym_test/small_cell_training/\
    test_iterative_training/iter/<i>/updated_model/train.cfg
```

And the model trained on the updated model is located in the same folder:

```
/home/qiuguo/pb-dpmodel/mlip3_ym_test/small_cell_training/\
  test_iterative_training/iter/<i>/updated_model/pot.almtp
```

The file size of train.cfg increases due to the continuous appending of new training configurations to $\mathcal{D}$. At iteration 1, the train.cfg file is 44MB and at iteration 60, the train.cfg file is 57MB. Iteration 1, i.e. the initial training data set consists of the 32 and 108 configurations used in the MTP level sensitivity study.

5.5 Difficulties of porting to HPC systems

I need to stress that the code development was on a private computer of the group (southpole) and not on supercomputing resources. This environment allowed complete freedom to run simulations continuously without concerns about job time limits, unexpected shutdowns, or queue-related cancellations. As a result, the scripting and workflow were designed around unrestricted, long-running execution.

Initial efforts to adapt the code for HPC systems were started on Pitagora but could not be completed. Porting the workflow to HPC environments is expected to be challenging, particularly due to incompatibilities with standard SLURM-based job submission practices. Typical HPC usage favors a small number of large jobs with predefined runtimes, often less than one day. In contrast, the current workflow relies on executing hundreds of small VASP calculations at reduced lattice sizes, which conflicts with common HPC scheduling policies.

Specifically, deploying the algorithm on HPC systems must address the following challenges:

1. The active learning process is inherently dynamic, and the total number of required VASP calculations is not known in advance.

2. Each active learning iteration involves hundreds of VASP calculations at smaller lattice sizes, potentially exceeding limits on the number of simultaneously submitted jobs.

3. After all VASP calculations in an iteration are completed, the machine learning model must be retrained. Triggering this step is nontrivial because the number of submitted VASP jobs is not predetermined.

4. A similar dependency exists for the LAMMPS simulations, which must be triggered only after model retraining has completed.

A potential solution is to reserve a large allocation of computing resources for a fixed time window and manage the entire workflow using a Python script, rather than relying on individual SLURM job submissions. This script would handle VASP calculations, machine learning interatomic potential training, and

LAMMPS simulations internally, while monitoring progress and recording completed tasks in a persistent checklist. If the reservation expires, the checklist can be used to resume the workflow from the previous state. Under this approach, the user would only need to submit a single job request periodically, for example every 24 hours, while the Python script manages all intermediate steps.

References

[1] Zijian Meng, Hao Sun, Edmanuel Torres, Christopher Maxwell, Ryan Eric Grant, and Laurent Karim Béland. Small-cell-based fast active learning of machine learning interatomic potentials. *Computational Materials Science*, 256:113919, 2025.

[2] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117:1–19, 1995.

[3] G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54(16):11169–11186, 1996.

[4] Evgeny Podryabinkin, Kamil Garifullin, Alexander Shapeev, and Ivan Novikov. Mlip-3: Active learning on atomic environments with moment tensor potentials. *The Journal of Chemical Physics*, 159(8):084112, 08 2023.