Fine-Tuning CHGNet on a Foundational Materials Dataset

Christopher Akiki, Eamon Bartlett, Ian Harreschou, and Jongkyeong Kim

Abstract

Density functional theory (DFT) has been the go-to methodology for performing electronic state calculations on solid state structures for the previous three decades. Its utilization has revolutionized computational modeling with its versatility and accuracy. However, these calculations are computationally expensive and represent a major bottleneck in accelerating the rate of materials discovery. Machine learning interatomic potentials (MLIPs) have been introduced with a promise to slash these "run-time" costs, instead shifting the burden of compute power to be upfront.

Several pre-trained MLIPs are currently available to the public, and this work focuses on one such MLIP, the Crystal Graph Hamiltonian Neural Network (CHGNet). We developed a custom fine-tuning pipeline that selectively trains the final convolutional layer of CHGNet, allowing it to adapt to a new dataset while retaining previously-learned chemical and physical information. The highest performing fine-tuned model achieved a mean absolute error (MAE) of ~100 meV/atom on test data, sitting higher than the pre-trained CHGNet's baseline error of 33 meV/atom. Further testing revealed that our model underpredicts DFT energies due to inherited biases from the pre-trained layers, which we addressed with a post hoc linear correction as described in MP2020Compatibility. While the fine-tuned model did not outperform the original, the project reveals key limitations in current transfer learning strategies for MLIPs and highlights the importance of dataset compatibility, architectural flexibility, and energy correction.

BACKGROUND

Accurate modeling of chemical systems is key to understanding complex materials, as many structure-property relationships span challenging time and length scales. While many first-principle, post-Hartree-Fock methods can capture these effects (to varying degrees), their poor scaling limits their use to small systems. Although suffering from many of these same drawbacks, DFT has been the gold-standard in performing electronic structure calculations since major improvements in methodologies circa the 1990s. Through educated choice of exchange correlation functional, electronic properties for various chemistries can be determined with high accuracy, though this is easier said than done.[2]

In recent years, data-driven methods have become an increasingly popular part of the materials science discovery toolkit, utilizing the vast amounts of experimental observations available in literature.[10] Machine learning interatomic potentials train various model architectures on DFT calculations to generate a robust predictor with comparable accuracy, but with speed that enables rapid, repeated deployment. This is particularly powerful in molecular dynamics (MD) simulations, which rely on classical models for interactions as in the case of the popular CHARMM or AMBER force fields.[3,4] The more intensive variant of MD, *ab-initio* molecular dynamics, uses DFT to calculate these interatomic forces at each timestep, again sacrificing speed for accuracy.[13] Another downside to these force fields is the inability to capture changes in molecular connectivity, as is the case in bond formation or breaking.[12] MLIPs–on the other hand–require large amounts of high-quality data to be effective, and training serves as a massive upfront barrier, but they are very generalized predictors. Once trained successfully, future property predictions are near-instantaneous, making their insertion into MD frameworks extremely powerful for studying phenomena that span large time and length scales.[11]

A number of pre-trained MLIPs are available to the public, all with widely varying input data, architectures, and particular methodologies. This project was focused on using one such MLIP, CHGNet, to determine if predicted ground state energies of specified groups of materials could be improved by additional model fine-tuning with unseen data.[6]

DATA

The data used in this work is the MatPES-PBE-2025.1 dataset, made available by collaborations between the Materials Virtual Lab and Materials Project.[1,7] It differs significantly from other popular training sets, namely Materials Project's own structural relaxation data, MPtrj, by prioritizing accuracy in DFT-computed potential energy surfaces, methodology compatibility and most importantly, a wide cohesion energy distribution. A major limitation of MPtrj is that nearly all structures in the set are those in equilibrium or energetic minima, thus any trained MLIP is forced to make out-of-distribution predictions when passed far-from-equilibrium structures. Overall, these (and similar) bodies of data can suffer from both systematic and non-systematic issues.

The authors of MatPES combat these by standardizing the DFT functionals used, employing stringent convergence criteria for single-point energy calculations, and by including both equilibrium and non-equilibrium structures as part of the set. The result is a much more information-dense collection that has shown improved performance and reliability in some cases, while being smaller overall and thus more forgiving for model training. The records in MatPES-PBE-2025.1 used here

amount to ~400,000 structures coming from a combination of MD simulations performed at 300K and directly from the Materials Project. The single-point energy calculated with a PBE functional is reported, though more accurate data via the r^2SCAN functional is available as well. There are at least 6,000 structures for each element besides unstable radioactive elements, rare earth metals, and noble gases, providing great coverage of the periodic table and technologically relevant oxides.

Specific subsets of this dataset used for fine-tuning in this work include any oxygen-containing compounds, transition metal oxides (Sc, Ti, V, Cr, ...), halides (F, Cl, Br, I), nitrides (containing N), sulfides (containing S), intermetallics (H, B, P, Se, ...), 'all' systems, or a random sample thereof. These categories contain roughly 120k, 85k, 45k, 45k, 20k, and 200k structures, respectively. The partitions are not mutually exclusive, and there is some overlap among them.

METHODS

Various helper functions and scripts were created to facilitate the extraction of structures and features for a user-defined chemical system from the raw MatPES.gz file (see *Supplemental Material*).

Data Pre-Processing The *data_preprocessing.py* module unzips and reads the raw MatPES JSON file. Entries from the dataset are filtered down by the user's input of a desired chemical system (as described in the *Data* section) via a *Filter* class. Optionally, it allows one to filter by a range of band gap values, or by crystal system (e.g. cubic, orthorhombic, *etc*). A second included class, *DataExtractor*, searches for and returns the corresponding structures, energy-per-atom, forces, magnetic moments, and stresses from the MatPES data for all candidate compounds. Finally, the *EnergyCorrector* class applies appropriate energy adjustments based on chemical composition to all DFT energies, as outlined in the Pymatgen MP2020Compatibility corrections.[8]

Graph Generation The *generate_graphs.py* module uses tools provided within the CHGNet Python package to generate atom and bond graphs from the filtered data. Pymatgen structures and target features are extracted, and these structures get passed into CHGNet's *CrystalGraphConverter* class. The module also creates a dictionary that maps each graph to its corresponding target features using a unique structural ID assigned in the dataset. User chosen atom- and bond-cutoff radii dictate how much raw information is retained from the atomic structure after the featurization process. The default values used in training are 5 Å and 3 Å, respectively.[7] These choices ultimately reflect a trade-off between the descriptive power of the generated features, and the computational complexity. It is important to note that using this module is optional as CHGNet has capabilities to automatically generate graphs under the hood when prompted to initiate training. However, the downside is that these graphs are then stored in the memory which can quickly become a problem when dealing with large enough datasets. This module fixes that problem by pre-creating the graphs making it so the training loop is accessing them from the hard drive rather than from RAM.

Consolidating the Workflow The entire preprocessing, graph creation, and fine-tuning workflow is handled by the *finetuning_launcher.py* file. The desired material system, model learning rate, number of training epochs, and cutoffs are set by the user for each training loop, where each run uses a

train/validation/test split of 90%/5%/5%. The pre-trained CHGNet model is loaded, and all but the weights of the final convolutional layer for atomic features are frozen in place. This approach ensures that much of the chemical and physical information learned during pre-training is retained, while providing an avenue for the model to ingest and fit to new data. Only the energies and forces from the MatPES dataset were used as training features, though CHGNet's pre-training would allow all 4 quantities to be computed if desired. Finally, fine-tuned models are saved to a directory with identifying information in the title, such as material system chosen, number of epochs, learning rates, and atom/bond graph cutoffs.

Random Forest Regressor The final module, *regressor.py*, is used to train and perform regression with a Random Forest ensemble model, implemented via Scikit-Learn.[9] The *CHGNetEnergyRegressor* class provides a pipeline for extracting the CHGNet-generated crystal embeddings that contain abstract features related to atom, bond, and angle information for an atomic structure and using them in a *RandomForestRegressor*. The pre-trained CHGNet model is loaded, and input compounds are passed through the architecture, generating all the embeddings. These features, along with the structures' per atom energy, are funneled into a data frame, then partitioned into a test and training set (20%/80% split by default). Optionally, a grid-search cross-validation is performed to identify an optimal set of hyperparameters for the forest, those being the number of estimators, maximum estimator depth, minimum samples needed for splits, and minimum samples allowed in leaf nodes. This tuning process helps ensure generality of the resultant ensemble model for the given dataset. The MAE and R^2 metrics are returned. Also included in the *CHGNetEnergyRegressor* custom class is the ability to generate parity plots and prediction residual histograms. The idea behind extracting the CHGNet embeddings and running them through a *RandomForestRegressor* is to evaluate the quality of the MatPES data irrespective of the deep learning model.

RESULTS

Fine-Tuning To understand how well the fine-tuned neural network performs, we look at the performance of the pre-trained CHGNet model in comparison. Specifically, since these fine-tuned models do not train on stresses or magnetic moments, the relevant benchmarks are energy and force MAEs without magmoms, amounting to 33 meV/atom and 79 meV/atom, respectively (see *Table 1*).

	Energy (meV/atom)	Force (meV/Å)	Stress (GPa)	Magmom (μ_B)
With mag	30	77	0.348	0.032
No mag	33	79	0.351	N/A

Table 1: *MAEs of CHGNet for energy, force, stress and magnetic moments predictions.*

We ran multiple fine-tuning iterations with different materials subsets of MatPES and different training parameters. We will only present the results for the transition metal oxides because of their versatile technological applications. The first model we fine-tuned was done with a maximum epoch count of 5, and a learning rate of 1e-2. In this initial iteration of fine-tuning we did not correct the

energies of the MatPES dataset (using the *EnergyCorrector* class discussed in the methods section). This yielded a very poor performance on the test set. Indeed, we got a test set energy MAE of 604 meV/atom. In contrast, the best fine-tuned we were able to achieve was obtained with 5 epochs and a learning rate of 1e-3, with the important inclusion of corrected energies. The test set energy MAE obtained for this model was 98 meV/atom, showing a significant improvement from the model trained on the uncorrected energies. To confirm that this wasn't simply an artifact of the lower learning rate and is instead due to the energy corrections themselves, we ran another model without correcting the MatPES energies but with 10 epochs and a learning rate of 1e-4 (doubling the epochs and using a two order of magnitude smaller learning rate). This resulted in a test set energy MAE of 125 meV/atom. While a large gain from the original test set energy MAE of 604 meV/atom, it still underperforms the model trained with energy corrections, despite using half the epoch count and a learning rate an order of magnitude larger (see *Table 2*). Because of these results, we feel confident in asserting that the energy correction is a necessary step in adapting MatPES to CHGNet.

Model	Epochs	Learning Rate	MAE	Energy Corr
bestE_epoch3_e98_f193_sNA_mNA.pth.tar	5	1e-2	98	True
bestE_epoch2_e604_f452_sNA_mNA.pth.tar	5	1e-3	604	False
bestE_epoch7_e125_f195_sNA_mNA.pth.tar	10	1e-4	125	False

Table 2: *Summary of the training results for the transition metal oxide subclass of MatPES. Models in which the energy correction was applied are denoted by a True/False in the final column.*

There are a number of potential reasons these performed worse than the pre-trained network. The most glaring one being that our tuning procedure would benefit from a greater number of epochs in standard training runs. We were not able to run anything larger than 10 cycles due to time and computational limitations. Even utilizing supercomputer resources, these models took upwards of 12-24 hours to fine-tune, depending on the specific subset of material types used.

Another explanation, and perhaps a more important reason these models underperformed, is that we only allowed the last atom convolutional layer to vary, leaving the remainder of the parameter weights and biases learned from the original training static. This means that the neural network was rigid, and any significant uptake of new information is expected to take a considerable amount of time. Additionally, MPtrj contains a mix of GGA and GGA+U energies whereas MatPES only contains GGA energies, which can also add another layer of error and complexity. We expect that retraining CHGNet from scratch–solely on MatPES–would have given much better results.

The next step was to apply our fine-tuned model to ~40 random transition metal oxide structures queried from the Materials Project to predict their ground state energies. As an initial test, we wanted to see how well our fine-tuned model would do on near equilibrium structures. To that end, we ran two DFT calculations on all the structures; a structural relaxation calculation to ensure convergence to an equilibrium position, followed by a static energy calculation to compute the energy

of the ground state. All calculations were performed with a GGA functional, matching the information contained in MatPES (see *Figure 1*).

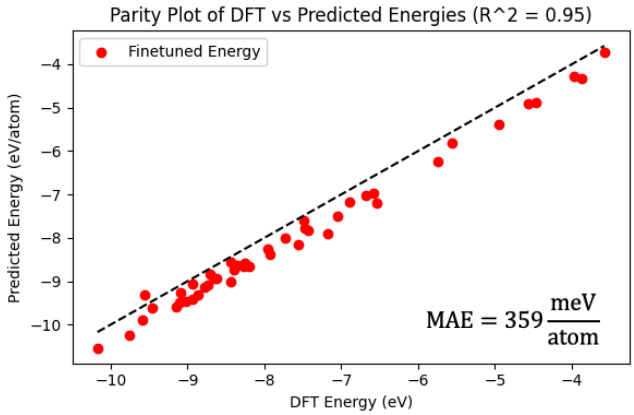

Figure 1: *Parity plot of predicted energies vs DFT energies. This was done on equilibrium structures.*

The first thing to notice is that the MAE is significantly higher than the test MAE of the model. This is however misleading because the fine tuned model is consistently under-predicting the energy of the structures. This problem was addressed by Deng et al.[5] In this paper, the authors highlight a consistent softening of the potential energy surface by uMLIP's. They posit that this is due to the biased sampling toward near equilibrium structures in the training datasets of these pre-trained models. This is certainly true for the MPtrj dataset CHGNet was pre-trained on. Because the majority of the layers of our model still contain the biases of the pre-training dataset, we observe this systematic softening. The authors suggest that a simple linear correction to the energy should be enough to account for the softening. Figure 2 shows a clear improvement in the energy prediction of the structures: an energy MAE of 120 meV/atom is much closer to the test set MAE of 98 meV/atom.

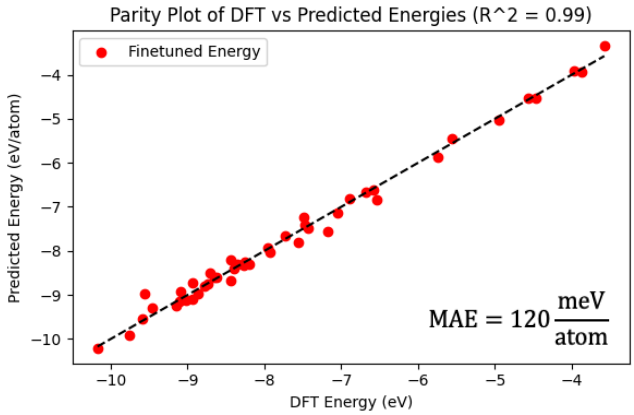

Figure 2: *Parity plot of the predicted energies by the best fine-tuned model vs DFT energies after systematic softening energy correction. This was done on equilibrium structures.*

The next thing to test was how well our fine-tuned model could predict the energies of non-equilibrium structures. To that end we queried the same ~40 structures from MaterialsProject and introduced a random translational perturbation of 0.5 Å before performing a single DFT static calculation. After applying the systematic softening correction, most of our structures lie on the parity

line with an energy MAE of 134 meV/atom (see *Figure 3*). As expected, our model's performance is worse on non-equilibrium structures compared to equilibrium structures. However, the important takeaway is that the performance decrease amounts to only a ~10% difference in meV/atom. This is an important result because it solidifies the claim of MatPES that a smaller dataset with a wider energy distribution is better than a large dataset with a narrow energy distribution around equilibrium.

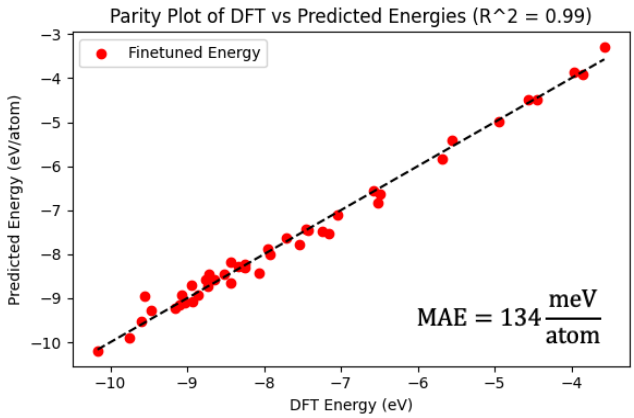

Figure 3: *Parity plot of the predicted energies by the best fine-tuned model vs DFT energies after systematic softening energy correction. This was done on non-equilibrium structures.*

RandomForestRegressor As explained in the methods section, the crystal embeddings of the pre-trained CHGNet model were passed through a random forest to predict the energies (*Figure 4*). Clearly, there is a high concentration of yellow dots (zero absolute residual). In fact, the test energy MAE was 65 meV/atom. This highlights the quality of the MatPES features. Indeed, the features are rich enough for the random forest to be able to overfit and memorize the energy mapping. The much higher test error confirms that a shallow machine learning model like a random forest is not enough to efficiently capture the complexities of the energy surface.

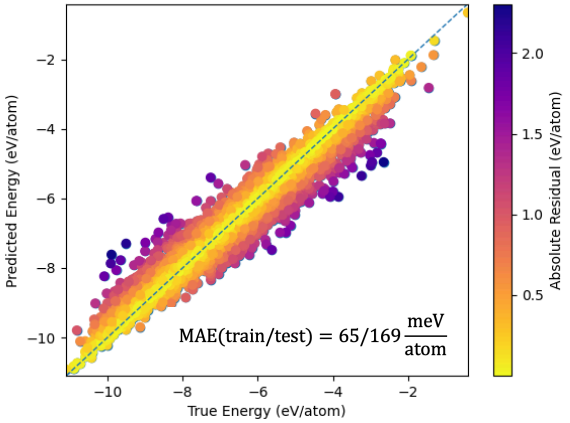

Figure 4: *Parity plot of the energy predicted from CHGNet embeddings vs DFT*

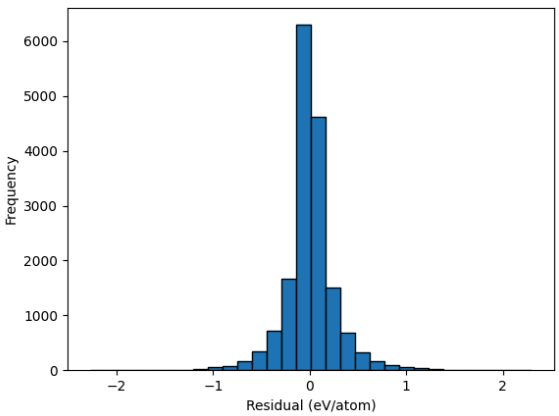

Figure 5: *Histogram of the residual distribution of the Random Forest regressor.*

Another useful visualization is a histogram of how the residuals (actual minus predicted values) are distributed (see *Figure 5*). The majority of the residuals are centered around zero. Another

interesting detail is that, despite the high concentration around zero, there are significant tails to the distribution. These likely correspond to high-energy, low symmetry, far-from-equilibrium structures: regions of MatPES where CHGNet lacked training data. This finding reinforces our claim that unfreezing more layers for fine-tuning would help CHGNet rid itself of its inherent bias towards equilibrium structures.

DISCUSSION

This project set out to evaluate whether fine-tuning a pre-trained model like CHGNet on a more diverse and orthogonal dataset (MatPES) could improve energy prediction performance for transition metal oxides. While our best fine-tuned model did not outperform the original CHGNet in terms of test MAE, the process revealed meaningful insights into the challenges and opportunities associated with adapting pre-trained MLIPs.

First, we demonstrated that including a compositional energy correction to the MatPES dataset based on the MP2020Compatibility correction was critical. Models trained on uncorrected data consistently underperformed, even with more aggressive training parameters such as epochs and learning rates. This highlights the importance of data compatibility and preprocessing when adapting off-the-shelf models to new datasets

Second, our results show that simply unfreezing the last convolutional layer limited the flexibility of the model to meaningfully internalize the new energy distribution MatPES was offering. The apparent systematic softening we observed further supports this idea. Even after we applied the linear corrections to account for this bias, residual errors remained. The Random Forest regressor trained on the feature embeddings of CHGNet showed low residuals for many structures, suggesting that MatPES is rich in predictive features. However, the tails of the residual histogram further cement our point that only unfreezing the last layer of the neural network did not give the model the freedom it needed to overwrite the biases from the pre-training dataset.

These findings motivated some recommendations for future work. The lowest hanging fruit to explore in the future would be to run our fine-tuning with a much higher epoch number and lower learning rate. This would be a computational expensive endeavor, likely requiring training on GPUs. Another avenue to possibly explore is to retrain CHGNet from scratch on MatPES data. Having a dataset like MatPES which is significantly different from the original dataset, simply fine-tuning might not be the correct approach. Again, such a feat would require GPU training. A solution to mitigate the computational cost is a staged approach to training. Instead of training from scratch on the entirety of MatPES, we would train on a smaller representative sample of MatPES, then fine-tune the last few layers of the newly trained model on the rest of the dataset.

Overall, while the fine-tuning did not surpass the pre-trained model's performance, it successfully exposed key limitations in current workflows and demonstrated the potential of more targeted model adaptation strategies in materials machine learning.

REFERENCES

[1] **Jain A et al.** Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. APL Mater. 1, 011002 (2013).

[2] **Becke AD.** Perspective: Fifty years of density functional theory in chemical physics. J. Chem. Phys. 140, 18A301 (2014).

[3] **Brooks BR et al.** CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. J. Comp. Chem. 4, 2, 187-217 (1983)

[4] **Cornell WD et al.** A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. J. Am. Chem. Soc. 117, 19, 5179–5197 (1995)

[5] **Deng B et al.** Systematic softening in universal machine learning interatomic potentials. npj Comput. Mater. 11, 9 (2025).

[6] **Deng B et al.** CHGNet as a pretrained universal neural network potential for charge-informed atomistic modelling. Nat. Mach. Intell. 5, 1031–1041 (2023).

[7] **Kaplan AD et al.** A foundational potential energy surface dataset for materials. arXiv (2025).

[8] **Ong SP et al.** Python Materials Genomics (pymatgen): A Robust, Open-Source Python Library for Materials Analysis. Comp. Mat. Sci. 68, 314-319 (2013)

[9] **Pedregosa F et al.** Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011).

[10] **Schmidt J, Marques MRG, Botti S, Marques MAL.** Recent advances and applications of machine learning in solid-state materials science. npj Comput. Mater. 5, 83 (2019).

[11] **Wang G et al.** Machine learning interatomic potential: Bridge the gap between small-scale models and realistic device-scale simulations. iScience 27, 109673 (2024).

[12] **Tuckerman ME, Martyna GJ.** Understanding modern molecular dynamics: techniques and applications. J. Phys. Chem. B 104, 159–178 (2000).

[13] **Paquet E, Viktor HL.** Computational methods for ab initio molecular dynamics. Adv. Chem. 1–14 (2018).