Implementation and Comparison of Machine Learning Interaction Potentials for TiO_2 Simulations

Kiyan Bates
PHYS 310
University of British Columbia
April 22, 2025

1 Introduction

Density Functional Theory (DFT) has become an essential tool in computational chemistry and materials science due to its ability to provide accurate quantum mechanical calculations of energies and atomic forces. Despite its predictive power, DFT suffers from significant computational limitations, it has poor scaling with system size, around $O(N^3)$, restricting its practical applications to small systems. As a result, exploring more large-scale phenomena is often too expensive to be practical in most cases.

To overcome this issue, researchers have increasingly turned to machine learning interaction potentials (MLIPs). MLIPs, such as the Behler–Parrinello artificial neural network (ANN) potentials, learn from a set of DFT-computed reference structures and energies to predict atomic interactions. By encoding local atomic environments through invariant descriptors (e.g., radial and angular symmetry functions like G2 and G4), MLIPs should ideally be able to get near-DFT accuracy at a fraction of the computational cost.

A notable implementation of ANN-based MLIPs was demonstrated by Artrith and Urban (2016), who achieved exceptional predictive accuracy for various phases of TiO_2. Their approach used carefully optimized descriptors and ANN architectures (using aenet) to accurately model energies, achieving errors lower than their target of approximately 5 meV per atom. Despite the promising outcomes reported, reproducing such high-accuracy results outside specialized research contexts is still difficult.

In this study, I initially aimed to reproduce the results demonstrated by Artrith et al. However, despite many attempts using other neural network implementations (e.g., TensorFlow-based models with various optimization and regularization strategies) or even regressor models, achieving comparable accuracy proved difficult. Significant deviations magnitudes higher than those of Artrith et al. highlighted critical subtle issues/differences in dataset preparation, feature scaling, model architecture tuning, training procedures, etc... that substantially impact the final accuracy.

Consequently, this paper adopts a slightly broadened objective: rather than strict reproduction, it will compare performances of the different MLIP architectures and training methodologies using TiO_2 data I used, in addition to comparing the achieved results to the

results of the paper. I will also attempt to highlight potential reasons that I was not able to achieve a comparably good performance.

1.1 Machine Learning Interaction Potentials (MLIPs) and the Behler–Parrinello Framework

MLIPs address the computational limits of DFT by learning relationships between atomic structures and energies from DFT data. The Behler–Parrinello method splits total energy into atomic contributions predicted by artificial neural networks (ANNs). It uses symmetry functions (G2 and G4) as input to ensure invariance to translations, rotations, and atom permutations. This framework is effective for systems like TiO_2 with complex chemical bonding and structural diversity.

1.2 Symmetry Functions: G2 and G4

Symmetry functions describe local atomic environments in an invariant way. The radial symmetry function (G2) describes pairwise distances within a cutoff radius, capturing radial distributions. The angular symmetry function (G4) accounts for angles and distances among groups of three atoms, capturing angular environments. Together, G2 and G4 provide a detailed representation of atomic environments suitable for accurate ML predictions. Here is the definition for the radial function G2 centered at atom i:

$$G_i^2 = \sum_{j \neq i} e^{-\eta(R_{ij} - R_s)^2} \cdot f_c(R_{ij})$$

Where a smooth truncation is achieved using a cutoff function:

$$f_c(R) = \begin{cases} 0.5 \left[\cos\left(\dfrac{\pi R}{R_c} \right) + 1 \right], & R \leq R_c \\ 0, & R > R_c \end{cases}$$

Here is the definition of the angular three bodied function centered at atom i:

$$G_i^4 = 2^{1-\zeta} \sum_{j \neq i} \sum_{k \neq i,j} (1 + \lambda \cos\theta_{ijk})^{\zeta} \cdot \exp\left(-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)\right) \cdot f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk})$$

where R_{ij}, R_{ik}, R_{jk} are the distances between atoms i, j,and k. θ_{ijk} is the angle defined by the three atoms, and ζ, λ, and η are parameters

1.3 What is Titanium Dioxide (TiO_2)?

Titanium dioxide (TiO_2) is a common material that shows up in stuff like sunscreen, paint, and even food coloring. TiO_2 has different stable polymorphs each with different structure and properties:

- **Rutile** (space group: $P4_2/mnm$)

- **Anatase** (space group: $I4_1/amd$)
- **Brookite** (space group: $Pbca$)

There are also other polymorphs, two high pressure polymorphs are:

- **Columbite** — α-PbO_2 structure (space group: $Pbcn$)
- **Baddeleyite** — ZrO_2 structure (space group: $P2_1/c$)

2 Methods

2.1 Dataset and Reference Energies

TiO_2 structures and DFT energies obtained from atomistic.net, including rutile, anatase, brookite, and selected high-pressure polymorphs (columbite and baddeleyite). Bulk data in the form of xsf files. Created a custom parser using the Atomic Simulation Environment (ASE) library. After descriptor calculations, the data is stores in a pandas dataframe with one row per atom. Dataframe contains structure id, species labels, atomic positions, per-atom forces, total structure energy, and the computed symmetry function components. Different groupings of energy likely correspond to different TiO_2 polymorphs:

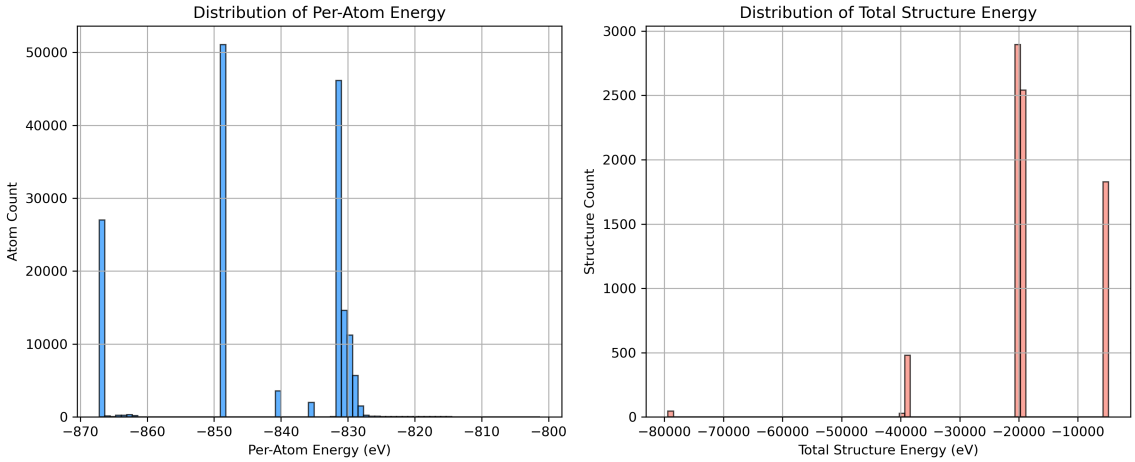

Figure 1: Distribution of per-atom (left) and total structure (right) energies across the dataset

2.2 Descriptor Calculations

Used ACSF descriptors, specifically G2 (radial) and G4 (angular) symmetry functions, computed using the DScribe library with a cutoff radius of 6.0 Å. Chose G2 and G4 parameters such that they span a range of sensitivities to ensure coverage. Species split into Titanium and Oxygen.

2.3 Machine Learning Models

Used a few models in order to illustrate their differences as well as see what would or could work best:

- **Ridge Regressor**: A simple linear model with L2 regularization, used more as a baseline to assess how much a non-deep model can capture from the ACSF descriptors. While fast and interpretable, it lacks the capacity to learn more complex interactions.
- **MLP Regressor (Scikit-learn)**: A feedforward neural network trained using backpropagation. The model served as a lightweight and relatively easy-to-tune benchmark for comparing against TensorFlow models. It includes fixed-depth, fully connected layers with configurable activation and learning rate.
- **TensorFlow ANN Models**: Several custom neural networks were built using TensorFlow and Keras. The models varied in depth, width, activation functions, dropout, and L2 regularization. A grid search or randomized search was performed over these hyperparameters to find configurations that performed most optimally. These models were the most flexible and had the highest potential, though they required more careful tuning and longer training time, getting half-decent results did not take as long with, for example, the MLP Regressor.

2.4 Training Procedure

The machine learning models were trained using Atom-Centered Symmetry Function (ACSF) descriptors, using radial (G_2) and angular (G_4) symmetry parameters. The dataset was split to 80% training, 20% testing, withing the training set 10% was allocated to validation. Prior to model training, input features and target variables (per-atom energies) (except for MLP Regressor, only input features were scaled) were standardized using scikit-learn's StandardScaler.
Three regression approaches were implemented and compared:

- **Ridge Regression**: Provided a baseline with L2 regularization, tuned via hyperparameter optimization for alpha.
- **Multi-Layer Perceptron (MLP) Regressors**: Implemented with scikit-learn's MLPRegressor, using GridSearchCV for hyperparameter tuning. This optimization included varying network architectures, activation functions (relu, tanh), learning rates, and regularization parameters.
- **Artificial Neural Networks (ANN)**: Developed using TensorFlow and Keras, incorporating Batch Normalization, Dropout, and L2 regularization to reduce overfitting. Hyperparameter optimization was done via a grid search, testing various configurations of hidden-layer sizes, learning rates, dropout rates, and batch sizes. Training employed the EarlyStopping and ReduceLROnPlateau callbacks for dynamic adjustment and hopefully improved training efficiency. Additionally used a randomized search on the per atom data (however I was unable to finish due to time constraints) and on per structure data.

For loss functions I used Mean Squared Error (MSE). Model performance was evaluated using Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and coefficient of determination (R^2). Individual models were trained separately for titanium (Ti) and oxygen (O) atoms, and their per-atom predictions were subsequently aggregated to predict total structure-level energies.

2.5 Implementation Tools

The project used a variety of Python libraries and tools to handle the atomistic datasets, feature extraction, machine learning modeling, and data visualization tasks:

- **Atomic Simulation Environment (ASE)**: For parsing .xsf files, storing atomic structures, lattice vectors, energies, and forces, facilitated through custom parsing functions for the dataset.
- **DScribe**: Used to calculate Atom-Centered Symmetry Functions (ACSF) which provides features needed for capturing local structural information.
- **Scikit-learn**: Used for preprocessing (feature scaling with StandardScaler), classical regression algorithms (Ridge regression), splitting datasets (train_test_split), and hyperparameter optimization (GridSearchCV).
- **XGBoost**: Used for feature importance analysis to identify the best predictive features for each atomic species.
- **TensorFlow and Keras**: Neural network architecture design, regularization (Batch Normalization, Dropout, L2 regularization), and training controls like EarlyStopping and ReduceLROnPlateau.
- **Matplotlib**: For creating plots/graphs and figures, including correlation plots of predicted versus true energies and error distribution histograms.
- **Pandas and NumPy**: Libraries good for data manipulation, transformation, and computation.

3 Results

3.1 Ridge Regression

As a sort of baseline method, I applied ridge regression to predict per-atom energies. A separate model was trained for each atomic species (Ti and O).
The optimal alpha values for both Ti and O were found to be 0.0, suggesting that no regularization was needed for best performance.

Table 1: Per-species ridge regression performance

Species	RMSE (eV/atom)	MAE (eV/atom)	R^2
Ti	5.64	4.07	0.8289
O	6.14	4.30	0.7912

To assess the overall performance, I combined predictions across all species. This yielded a per-atom RMSE of 5.97 eV, MAE of 4.22 eV, and $R^2 = 0.80$. Furthermore, when total structure energies were recovered by summing predicted atomic contributions per structure, we obtained a per-structure RMSE of 16.26 keV and MAE of 14.25 keV.

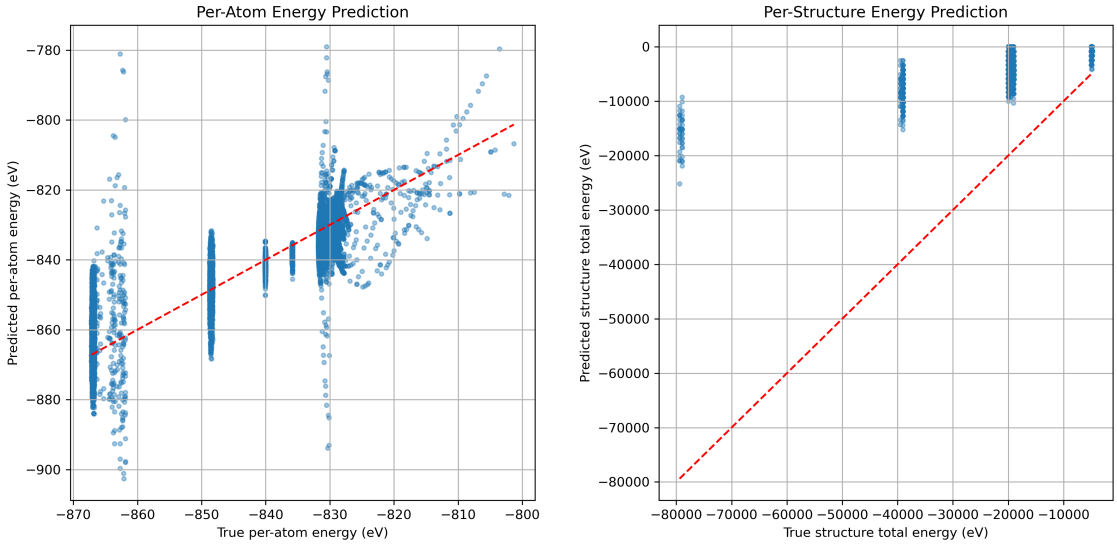

Figure 2: True vs predicted energy for Ridge Regression. Left: per-atom energies. Right: per-structure total energies. Red lines represent perfect prediction.

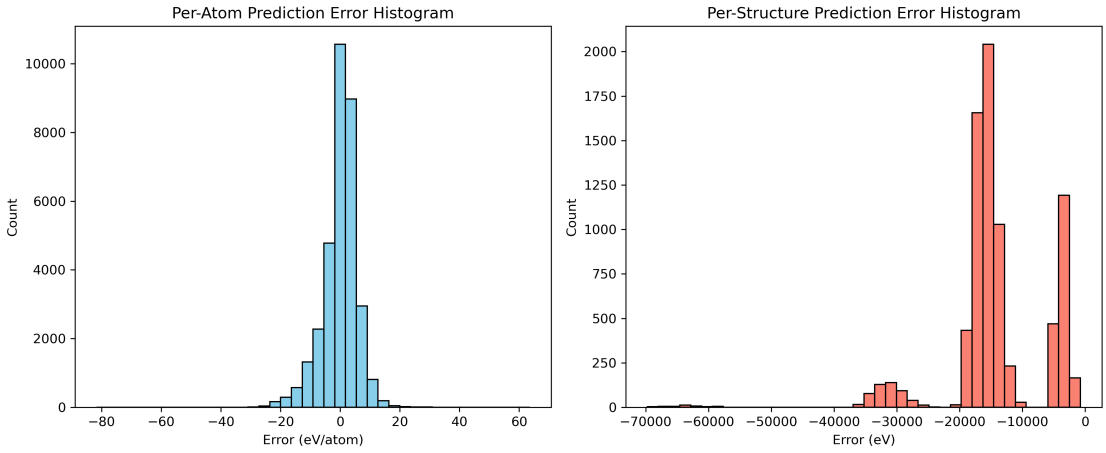

Figure 3: Histogram of prediction errors for Ridge Regression. Left: per-atom. Right: per-structure.

Although ridge regression provides a reasonably accurate and interpretable baseline, the relatively high structural error underscores the need for nonlinear models such as neural networks to better capture the complexity of the potential energy surface. There is little to make better here, it has no future.

3.2 MLP Regressor Results

I trained a multi-layer perceptron (MLP) regressor with three hidden layers (128, 64, 32 neurons) using the tanh activation function and L2 regularization ($\alpha = 0.01$) to predict per-atom energies of Ti and O species separately. Standardized feature inputs were used, and early stopping was enabled to prevent overfitting. The set of parameters was found through some hyper parameter optimization.

Per-Species Evaluation

The model achieved relatively alright accuracy in predicting per-atom energies for both titanium and oxygen atoms:

- **Titanium (Ti)**
  MAE: 1.7900 eV/atom
  RMSE: 3.2729 eV/atom
  R^2: 0.9431
- **Oxygen (O)**
  MAE: 2.5366 eV/atom
  RMSE: 4.2016 eV/atom
  R^2: 0.9030

This shows that the model captures both atom types reasonably well, with slightly better performance on Ti than O.

Combined Per-Atom Evaluation

When combining all atomic predictions across species:

- MAE: 2.2804 eV/atom
- RMSE: 3.9078 eV/atom
- R^2: 0.9171

This confirms consistent performance across for all atom data, with relatively low overall error and strong correlation with true values.

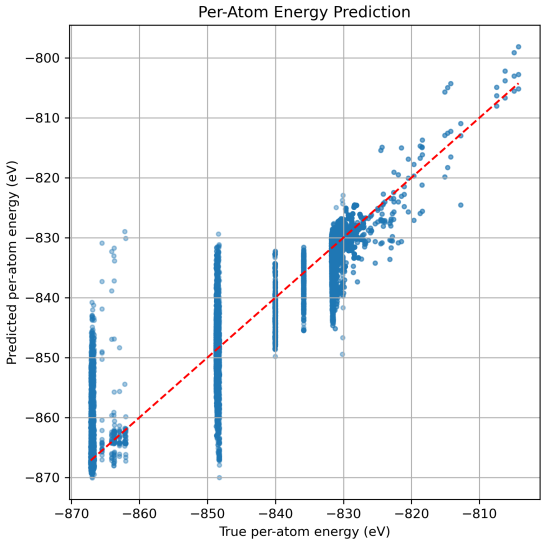

Figure 4: Per-atom true vs. predicted energy using the MLP regressor.

Per-Structure Evaluation

Getting total structure energies shows relatively decent error values and good R^2 as well:

- MAE: 332.7539 eV
- RMSE: 446.2731 eV
- R^2: 0.9978

The nearly perfect R^2 reflects that errors in per-atom predictions tend to average out when summed over entire structures, resulting in quite accurate total energy estimates.

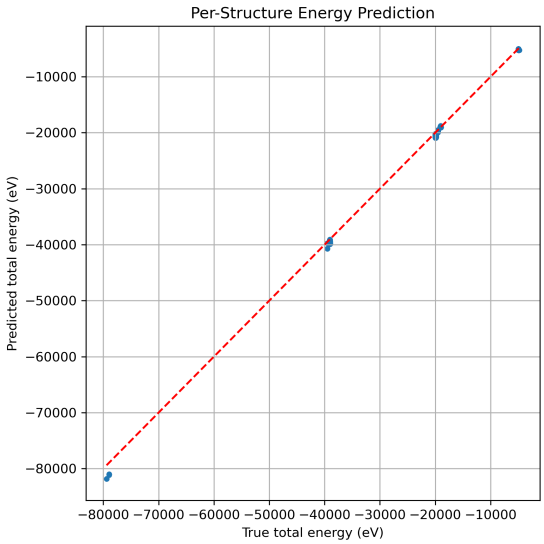

Figure 5: Total structure true vs. predicted energy using the MLP regressor.

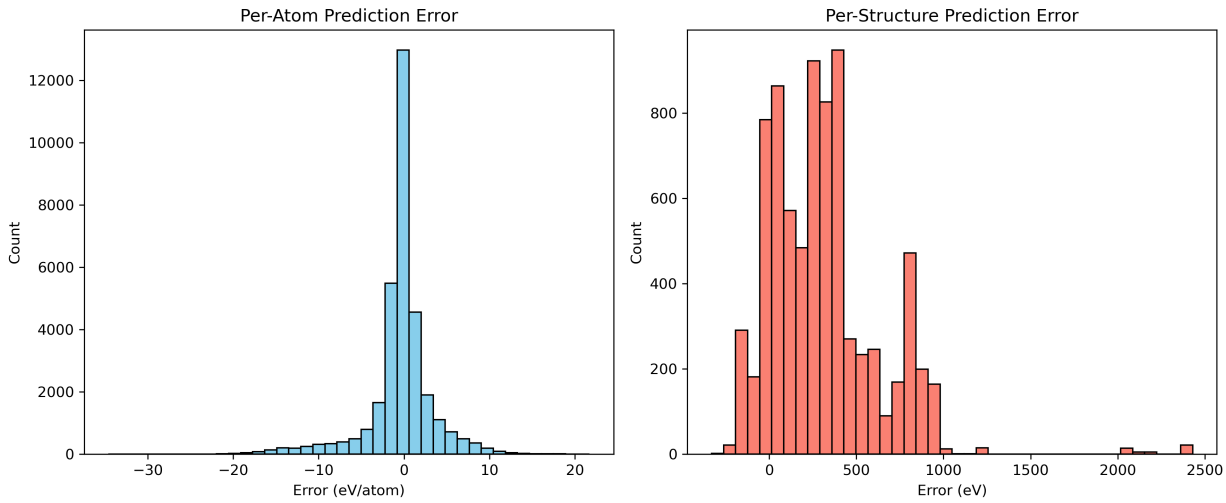

Figure 6: Histogram of prediction errors: (left) per-atom, (right) per-structure.

3.3 Artificial Neural Network (ANN) Results

I trained species-specific artificial neural networks (ANNs) to predict atomic energy contributions of Ti and O atoms in TiO_2 structures. Each model used a subset of features selected by prior XGBoost analysis. The networks consisted of three fully connected layers with tanh activations and were trained using the Adam optimizer with early stopping and learning rate reduction on plateau. Training was performed on standardized per-atom energies using mean squared error (MSE) loss and mean absolute error (MAE) as a monitoring metric. Used grid search to determine parameters.

Per-Atom Performance

The resulting performance for the Ti and O models was:

- **Ti**: MAE = 2.0083 eV/atom, RMSE = 3.4361 eV/atom, $R^2 = 0.9690$
- **O**: MAE = 3.3639 eV/atom, RMSE = 5.0232 eV/atom, $R^2 = 0.9525$
- **Combined**: MAE = 2.8986 eV/atom, RMSE = 4.5414 eV/atom, $R^2 = 0.8881$

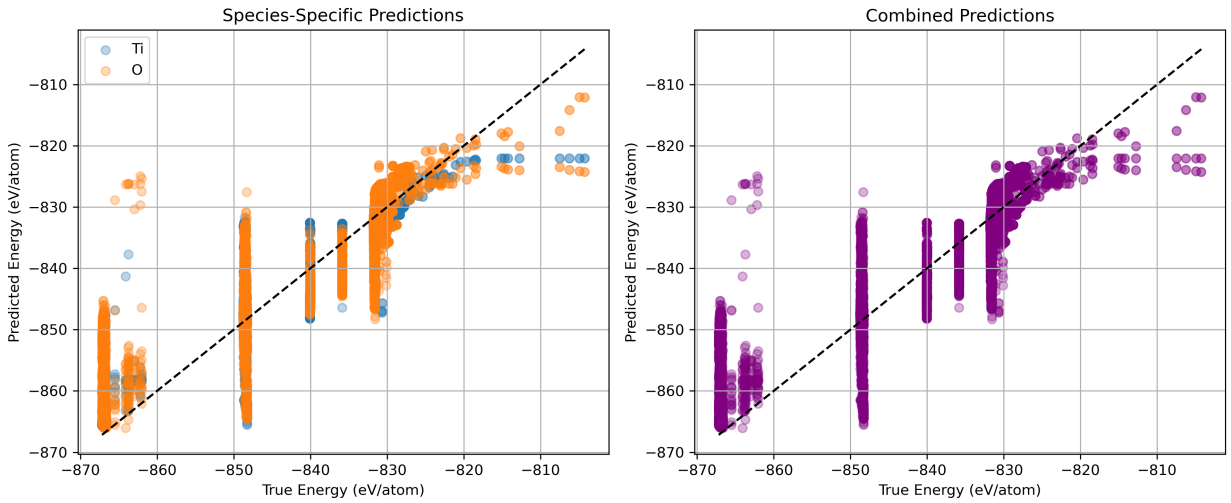

Figure 7: Per-atom energy predictions for Ti and O atoms compared with DFT references. The black dashed line represents the ideal prediction line.

Both models achieve strong enough scores, with the O model exhibiting higher per-atom error, most likely due to more complex or more variable atomic environments.

Per-Structure Total Energy Performance

Summing the predicted per-atom energies for each structure yields total energy estimates, which were compared directly with the original total energies. The resulting per-structure metrics were:

- MAE = 282.50 eV
- RMSE = 393.35 eV
- $R^2 = 0.9984$

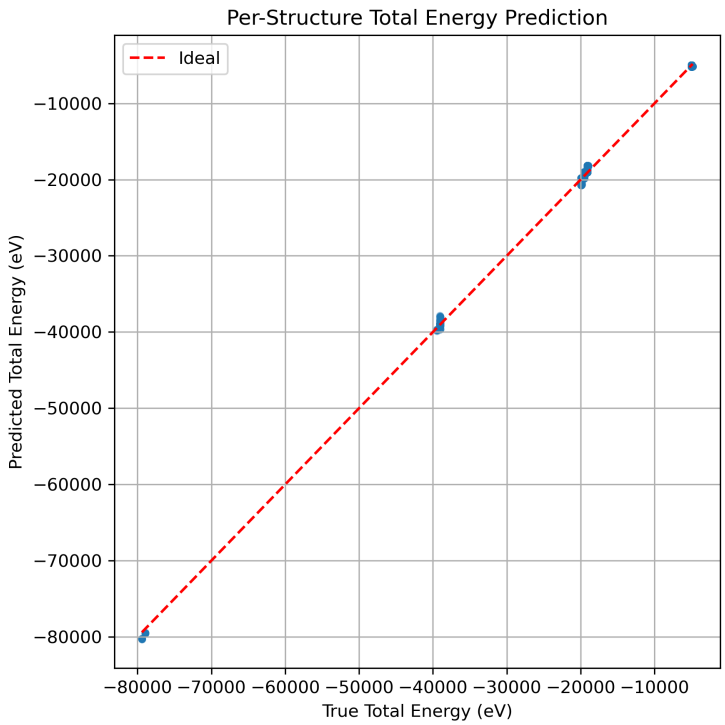

Figure 8: Per-structure total energy predictions vs. DFT reference energies. Predictions are obtained by summing per-atom ANN outputs.

While the absolute error appears large, this should correspond to per-atom errors consistent with those reported above.

Visualization and Training Behavior

Learning plots end early due to using early stopping.

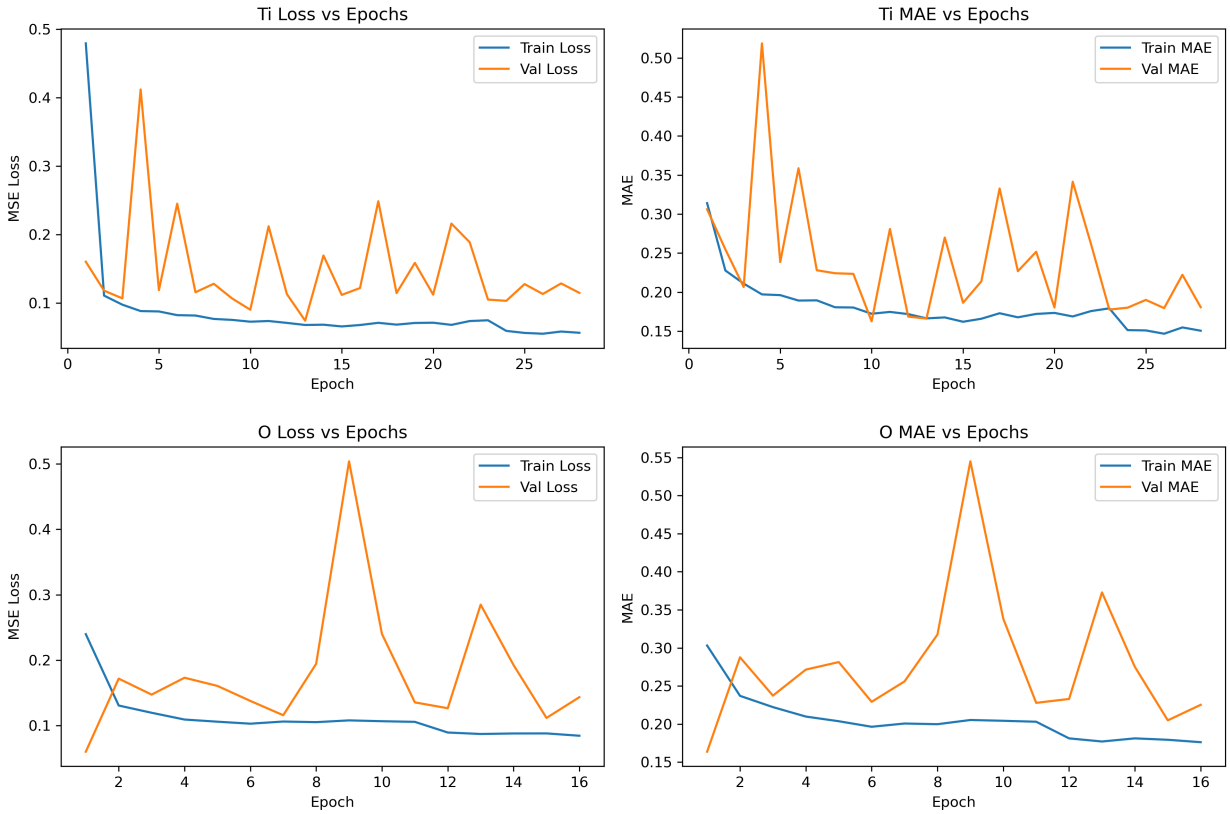

Figure 9: Training and validation loss/MAE curves for Ti (left) and O (right) species-specific models.

4 Discussion and Conclusion

The original goal of this project was to reproduce the results from Artrith and Urban (2016), which reported extremely low per-atom errors in predicting TiO_2 energies using neural networks. While I didn't manage to match their accuracy, the process gave me a solid understanding of what actually goes into building a working machine learning potential.

I tried a range of models, from simple ridge regression to custom neural networks built in with TensorFlow. Ridge regression worked okay as a baseline, but could not capture enough of the complexity of the atomic interactions. The neural networks did much better, with per-atom MAEs in the 2–3 eV range and solid enough per-structure performance, sometimes hitting R^2 scores above 0.99. That said, there's still a noticeable gap between these models and what's possible with highly optimized research-grade systems.

Some key takeaways were just how sensitive these models are to feature scaling, symmetry function tuning, and the choice of model architecture. I also found that training separate models for each element helped, and that getting even half-decent results didn't necessarily require super long training times if you had early stopping and good regularization. However, getting good results requires lots of time, I believe I could have achieved significantly better results had I had more time. This is because with a lack of time and a relatively weak

computer, running enough hyperparameter optimization was not feasible. In fact, I had to scrap my randomized search completely due to time constraints.
In the future, it'd be interesting to try other descriptors like SOAP, or even more advanced methods like graph neural networks. Enough time to further polish the ANN's could also be good, or optimizing the G2 and G4 parameters. But even with basic tools, I was able to build something that works pretty well, even if it did not come close to the performance of Artrith et al.

References

[1] Nongnuch Artrith and Alexander Urban. An implementation of artificial neural-network potentials for atomistic materials simulations: Performance for TiO_2. *Computational Materials Science*, 114:135–150, 2016.