Approaches to DFT Parameter Learning

Auguste Poiroux, Nataliya Paulish, Philipp Weder

I. Introduction

Simulations of materials are crucial for the discovery of novel functional materials that are needed for energy storage and conversion, high-tech industries, environmental protection and many other fields of cutting-edge science and technology. Density functional theory (DFT) is one of the most powerful quantum mechanical modeling methods to investigate the electronic structure of atoms, molecules and condensed phases. Its significance is impressively demonstrated by the fact that DFT is presented by 12 papers in top-100 most cited papers in the entire scientific literature ever [1]. DFT helps experimentalists to confirm their findings and also greatly facilitates identifying promising materials for further research by performing large scale computations for many materials.

The computations are based on a complex iterative algorithm and the accuracy of the results depends strongly on the parameters of the computation, where by accuracy one typically means the difference in total energy with respect to some high-accuracy reference calculation. The computation time greatly increases when the parameters are chosen to increase the accuracy. Thus, before any computation, one needs to perform convergence studies with respect to the parameters in order to minimize computational time, while still getting reasonable results. This is an intricate and time-consuming procedure that slows down large-scale screening of the periodic table.

One of the ways to address this problem is to use published data on convergence studies for individual chemical elements, for instance [2]. However, for many-element systems, the convergence curves may significantly differ from the ones computed for individual elements.

There have been attempts to automate the search for optimal parameters using machine learning techniques, e.g. [3] or very recently in [4], but generally results in this area remain sparse.

The goal of this project was to use machine learning techniques to predict both the energy accuracy and the duration of a simulation run for a given set of parameters and chemical structure. Moreover, we also developed an approach to solve the inverse problem, i.e. predicting a set of parameters that yields a given energy accuracy for a prescribed chemical structure since this would be extremely useful in practice.

For simplicity, we restricted ourselves to the most relevant computation parameters: The energy cut-off for the wave functions and the charge density as well as the density of the so-called k-point grid (for a brief explanation see section A in the appendix). Furthermore, in this first attempt, we only considered materials consisting of two atomic species.

II. The data set

There is a lot of data from DFT simulations available. However, they typically do not evenly cover the parameter space and may be heavily biased by heuristics. Therefore, we decided to generate our own dataset on the basis of the Quantum ESPRESSO software package [5].

A. Simulations and their outputs

To perform simulation, we first queried the API from www.materialscloud.org [6] to collect all the configuration files for materials with two distinct elements and two atoms per unit cell. Then, using the script provided by the THEOS lab, we generated input files with the desired parameter grid and submitted the jobs to the EPFL cluster Fidis. Next, the output files were parsed to generate a JSON file with selected input parameters and output results. In total, we have run a few hundred thousand computations for about 900 systems, covering most part of the periodic table, see figure 1. This resulted in a new dataset of about 500'000 data points.

B. Encoding of the chemical structures

To allow a model to discriminate between different chemical structures, they have to be encoded into numeric values. However, there is no distinctive way to encode a chemical structure. Clearly, one desires such an encoding to capture as much of the underlying chemistry as possible, such that elements with similar properties have similar features. In doing so, one hopes the resulting models will be able to generalize to some extent to unseen structures. Yet, if the encoding is too general, the model will not be able to distinguish between structures in the train set and hence the prediction accuracy suffers. Indeed, then two structures with different outputs might have identical inputs, but different outputs.

The following encodings are motivated by the properties of the periodic table, sorted from element-specific to general:

a) Atomic encoding (A): This encoding consists of having one column per element storing the relative contribution of each element to the total number of atoms in the structure. For example, the structure $\mathrm{Ge_1Te_1}$ would translate to an entry of 0.5 in the columns corresponding to Ge and Te, respectively, and zeros otherwise.

b) Valence configuration encoding (VC): Each electron has a unique electron configuration, which is determined by its quantum numbers. The most important electrons are the valence electrons, which are the ones occupying the outermost orbitals. This part of the electron configuration is called the *valence configuration*, cf. [7] for a precise definition. For each element in the structure, we then added its number of valence electrons in an orbital s, p, d or f to the corresponding column

in the dataset, weighted by the index of the outermost shell to distinguish between small and large cores.

c) Column-mass encoding (CM): In this encoding, we have one column for each column in the periodic table, where we treat the lanthanides separately. In each column, we then store the relative contribution of elements in this column to the total mass of atoms in the structure.

d) Column encoding (C): This encoding is similar to the one above with the only difference, that here the relative atomic mass is ignored, i.e. in the column corresponding to column I in the periodic table, we only store the relative contribution to that column by atoms in the structure.

In what follows, we will train our models on each of these encodings and compare performance.

III. Toolbox

A. Prediction quality assessment

Being able to assess the quality of predictions made by models is an important task in machine learning. It allows us to discriminate models in order to take the best ones. But it is important to keep in mind, that the loss function should be meaningful and relevant. So, next to the mean squared error (MSE) and the mean absolute error (MAE), we also consider the mean absolute percentage error (MAPE) to have a relative measure of deviation from the target.

Our models should generalize in two ways: On one hand to unseen parameters for known structures and on the other hand to unseen structures for known parameters. The second one is clearly more challenging since the model has to infer the structure's properties from the encoding. We evaluate our modes on test sets targeting both generalizations.

The target ΔE spans ten orders of magnitude, that is from 10^0 to 10^{-9}. Hence, we had to be careful when choosing a loss function to evaluate our models and assessing the results we got for this target (see section IV).

B. Models

Since we are not aware of any prediction methods on the tasks we are tackling in this project, we had to set up simple baselines on which we can improve. Thus, as baselines, we used `DummyRegression` and `LinearRegression` from `sklearn` [8]. We then used more complex models in an attempt to get more accurate results. Thus, we tried linear regression models with feature augmentation, random forest (`RandomForest` from `sklearn`), gradient boosting methods (`GradientBoosting` from `sklearn`, `XGBoost` [9] and `LightGBM` [10]), and finally neural networks.

While we found some relevant feature augmentations at the beginning, they became computationally intractable as our dataset became larger and larger. We also found on smaller datasets, that these feature augmentations only improved significantly the prediction accuracy of the linear regression models. And even then, linear model predictions were already worse than our best models. Therefore, we ended up not using them in the end.

At first glance, neural networks seem to be a promising path due to the flexibility of these models. Current neural network libraries allow us to customize the whole model, including the loss we want to optimize. Despite the different ideas we tried, we found that the results we obtained with neural networks were disappointing. Prediction performances were not better than the ones we got with other models, and the training time was much longer. Although this was expected [11], we still tried to get decent results. Eventually, we stopped working on these methods and stuck to the other models mentioned above.

C. Hyperparameter tuning and model selection

In order to improve the performances of our models, we tried to tune their hyperparameters. However, for instance, one single fit of a random forest model takes a considerable amount of time due to the size of the dataset. Thus, with the computational resources at our disposal and the different objectives we had to train on, making a full cross-validated grid search was infeasible. Therefore, we first searched manually for the more crucial parameters of our models. For example, in the random forest case, in our setting, the most significant parameters seem to be the number of estimators `n_estimators`, the number of features considered at every split `max_features` and the maximal depth of the trees `max_depth`. After this first step, we then applied `RandomizedSearchCV` from `sklearn` to find good sets of hyperparameters. We finally decided to use the parameters reported in table 3.

IV. Forward prediction of the energy accuracy

Running a DFT simulation corresponds to a forward process, which maps the input simulation parameters P and the chemical structure S to the output, in this case the energy accuracy $\Delta E := |E_{\text{sim}} - E_{\text{ref}}|$. We defined the reference energy E_{ref} to be the total energy of the simulation with the largest cut-off radii and the most k-points for a given structure. In more mathematical terms, we are considering the map

$$f_{\Delta E} : (P, S) \in \mathcal{P} \times \mathcal{S} \mapsto \Delta E \in \mathbb{R}_+,$$

where $\mathcal{P}$ and $\mathcal{S}$ denote the space of parameter sets and structures, respectively.

A. Feature engineering

Since the variable to predict is continuous, we decided to first apply standard regression techniques. A graphical analysis of the target variable ΔE shows that there is a strong exponential relationship with the input parameters, especially with `ecutwfc`, cf. figure 2. This motivates a $\log$-transformation of the target variable before training the regression models. Indeed, after the transformation, the target variable is much more evenly distributed. We therefore trained regression models on both ΔE and $\log(\Delta E)$.

B. Regression results

The scores of the different models we trained on predicting ΔE are in the table 4. Given the results in this table, the atomic encoding performed best in almost all settings for the

best models (Random forest and XGBoost). On the parameter generalization test set, random forest outperforms XGBoost for all the losses. However, on the structure generalization test set, XGBoost performs better on two of the three losses. In each case, the losses are almost the same for these two methods. In order to have a better intuition of the actual accuracy of these models, we sampled some predictions shown in the figure 7. We see in this figure that sometimes XGBoost predicts negative values and that the random forest predictions seem better than the ones from XGBoost. Therefore, in this case, MAPE seems to measure more accurately the generalization performance of the models.

The scores for random forest and XGBoost model trained on $\log(\Delta E)$ are reported in table 1, where the output has been transformed back before evaluating the scores to ensure comparability with the direct models. Here, random forest outperforms XGBoost for all losses and on all sets.

While the MAPE seems to be an intuitive quality measure of the predictions, there are some caveats. The MAPE is not reliable as a standalone measure in our case for two reasons: first, a constant prediction of zero would yield a MAPE of 100%, which might be a local minimum (neural networks keep converging there). Second, a MAPE of less than 1000% might still be acceptable for an upper prediction (prediction larger than the actual value) of an error since then the error stays in the same order of magnitude.

C. A classification approach

On one hand, it proved to be difficult to enhance and correctly assess the performance of the regression models above. On the other hand, we observed, that some of our regression models correctly predict the order of magnitude of the error ΔE very consistently. Furthermore, in practice, a scientist typically specifies the order of magnitude that the error should not exceed, and then chooses the parameters accordingly. This motivated the training of classifiers, which predict the order of magnitude of ΔE by using the transformation $y \in \mathbb{R}_+ \mapsto \lfloor \log_{10}(y) \rfloor \in \mathbb{N}_0$.

D. Classification results

To evaluate the generalization ability of our classifier, we ran a five-fold cross-validation for both models. The results are reported in table 2. The cross-validation was performed both with and without shuffling of the data before performing the splits. Since our data was parsed from ordered directories, the structures appear in blocks. Hence, cross-validation with shuffling corresponds to parameter generalization, i.e. the model sees (almost) all structures upon training but not all possible parameters. Conversely, cross-validation without shuffling corresponds to leaving out structures from the training set. Therefore, it measures the generalization error of the model. From the results, we may conclude that the model is rather robust with respect to both types of changes in the dataset, which is a good sign. Nonetheless, there is a large discrepancy between the error with respect to unseen parameters and the one with respect to unseen structures.

The results from the classification approach look rather promising, especially with respect to the prediction on unseen parameters. This means, that the model interpolates very well on the domain of the cut-off radii `ecutwfc` and `ecutrho` and the number of k-points `k_density`. The model performs less well on unseen structures, only giving the correct result two thirds of the time.

By looking at the confusion matrix in figure 4, we see that the model confuses primarily adjacent orders of magnitude. Furthermore, we point out that from the user's perspective, only the case where the model underestimates the error is really harmful. Since the confusions are distributed quite symmetrically around the main diagonal, we may conclude that in practice choosing parameters, which are predicted to yield a certain energy accuracy, yields the desired result even more than two thirds of the time for unseen structures. Hence, our model provides a good starting point to choose parameters for a prescribed energy accuracy and chemical structure, at least for two atomic species. To facilitate this task further, one can plot the decision function of our model as a function of `ecutwfc` and `k_density`, as we have done in figure 5 for the structure NaCl. Recall that according to a common heuristic, the parameter `ecutrho` is a fixed multiple of `ecutwfc`, with some exceptions, cf. [2]. Therefore, the set of parameters is in principle completely defined by choosing `ecutwfc` and `k_density` and thus a two-dimensional plot is sufficient. This is certainly not a perfect way of choosing optimal parameters, especially since there is still too much freedom to automate it. Nevertheless, it would be interesting to see how they apply in practice, especially compared to the rather sparse one-dimensional convergence plots currently available on Materials Cloud [2].

V. Forward prediction of the simulation time

In addition to the forward prediction of the energy difference ΔE, we also want to predict the simulation duration T_{sim}, that is; we approximate the map

$$f_T : (P, S) \in \mathcal{P} \times \mathcal{S} \mapsto T_{\mathrm{sim}} \in \mathbb{R}_+.$$

At first glance, a standalone application of such a model might not seem very interesting. However, as we will see in section VI below, we can combine this forward model with the forward model for ΔE to approximately solve the inverse problem.

The training approach for the simulation time stays the same as the one for the energy difference, with the sole exception of the MAPE becoming more meaningful as an error measure.

A. Results

Results for the different models trained on this task are in the table 5. Here, the predictions results seem to depend less on the structure encoding used. This is on par with the intuition that the simulation time depends more on the simulation parameters than on the chemical structure.

For this task, XGBoost performs slightly better than random forest for almost all losses on the test sets.

VI. The inverse problem

So far, we have only treated the forward problem of predicting the energy difference ΔE to the reference value. However, in an ideal world, a scientist would like to choose a chemical structure and prescribe an energy accuracy and then get a suitable set of parameters. But the forward process $f_{\Delta E}$ is clearly not invertible since there might be various sets of parameters yielding the same energy accuracy for a given structure. Nevertheless, from the perspective of the scientist, there is still one preferred set of parameters among those, namely the one for which the simulation converges fastest. Furthermore, in practice, a prescribed energy accuracy is to be understood as an upper bound rather than a value to be matched exactly.

In summary, we cannot invert the map $f_{\Delta E} : (P, S) \mapsto \Delta E$, but we can solve the following constrained optimization problem: for a given structure $S \in \mathcal{S}$ and a prescribed energy accuracy $\Delta E \in \mathbb{R}_+$, find

$$\arg \min_{P \in \Omega(S)} f_T(P, S),$$

where the feasible set $\Omega(S)$ is given by

$$\Omega(S) := \{P \in \mathcal{P} : f_{\Delta E}(P, S) \leq \Delta E\}.$$

A. Methodology

We do not know the maps $f_{\Delta E}$ and f_T exactly, therefore we substitute them by the regression models described above. Some models we use are not differentiable, therefore we cannot use gradient-based optimization methods. We currently use the method `differential_evolution` from `sklearn`, which is gradient-free. Since our constraint is also not differentiable, we choose to optimize a penalized version of our optimization problem:

$$\arg \min_{P \in \mathcal{P}} f_T(P, S) + \mu \max(f_{\Delta E}(P, S) - \Delta E, 0),$$

for some penalization parameter $\mu > 0$. Since `differential_evolution` is gradient-free we can use an arbitrary large μ without hurting the optimization process. Therefore, we choose $\mu = 10^{100}$, which is far larger than any feasible simulation time. By doing this, we force the optimization process to prefer solutions that respect the constraint to any solution that does not.

B. Results

We applied our algorithm to 100 structures and six different bounds for ΔE, yielding 600 predictions for optimal parameters. The computation time for each of the optimization was at the order of seconds, which is acceptable for our use case. Note that these combinations of chemical structures and parameters are not necessarily present in our dataset. Therefore, we had to run the actual simulations to evaluate the predictions.

The results show that our predicted parameters respect the prescribed energy bound 50% of the time and that the simulations stay within one order of magnitude in 82% of the time. In fact, the expression $\log_{10}(\Delta E_{\text{sim.}}/\Delta E_{\text{pred.}})$ appears to be approximately normally distributed cf. figure 6.

While complementary analyses of the results are needed in order to better assess the quality of our predictions, we already have some ideas to improve them further. One known issue is the training objective. Since it is symmetrical, we equally penalize the model for both under- and overestimating the error. Yet, the constraint in the optimization problem is an upper bound and thus underestimation should be penalized more during the training process of the model. This could be addressed by an asymmetrical training loss.

VII. Conclusion

In this project, we managed to take first steps towards optimal parameter learning for DFT simulations: We gathered a large dataset of simulation in- and outputs. We explored several ways of encoding chemical structures into numerical input features. We trained regression models to predict the energy accuracy and the simulation time, respectively, for a given set of parameters and chemical structure. The evaluation of the model predicting the energy accuracy was not straight-forward, which motivated us to train a classifier predicting the order of magnitude of the energy discrepancy. The classifier proved to perform very well on unseen parameters and satisfactorily on unseen structures. In particular, it allows a scientist to graphically choose parameters for a given chemical structure, with which a desired energy accuracy should be achieved. For both the regression and the classification models, the results show a clear picture: the random forest models trained on the atomic encoding generally work best. Lastly, we developed an approach to solve the inverse problem by numerically solving an optimization problem based on the forward regressors.

Future steps may include the following: first, it would be beneficial to recompute the values for ΔE on the basis of more accurate reference calculations. Indeed, it seems like much of the uncertainty in the model comes from the data points with very small values of ΔE, i.e. it is possible that there is not much information left if the reference energy stems from the same kind of calculation. Second, how to measure the prediction quality for this problem is still unclear to us. Initially, the MAPE seemed to be an intuitive way to measure the deviation of the predictions compared to the actual values. However, as pointed out in III, it turns out that this measure is not well suited to handle data with several orders of magnitude difference. Thus, the design of a specialized scoring method for this type of problem would be interesting. Another observation is that measuring errors on $\log(\Delta E)$ predictions seems to be more straightforward and consistent than on ΔE. Further work may analyze more in depth the effect of predicting $\log(\Delta E)$ instead of ΔE. Lastly, one could further develop the inversion approach by, for example, using an analytic approximation of the computational complexity, implementing an invertible neural network architecture such as in [12], or implementing some active learning method using the simulation parameters predicted by the optimization process.

References

[1] R. V. Noorden, B. Maher, and R. Nuzzo, "The top 100 papers," *Nature*, vol. 514, no. 7524, pp. 550–553, Oct. 2014. DOI: 10.1038/514550a.

[2] G. Prandini, A. Marrazzo, I. E. Castelli, N. Mounet, E. Passaro, and N. Marzari, *A standard solid state pseudopotentials (sssp) library optimized for precision and efficiency*, en, 2021. DOI: 10.24435/MATERIALSCLOUD:RZ-77.

[3] K. Choudhary and F. Tavazza, "Convergence and machine learning predictions of monkhorst-pack k-points and plane-wave cut-off in high-throughput dft calculations," *Computational Materials Science*, vol. 161, pp. 300–308, 2019, ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2019.02.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927025619300813.

[4] J. Kirkpatrick, B. McMorrow, D. H. P. Turban, *et al.*, "Pushing the frontiers of density functionals by solving the fractional electron problem," *Science*, vol. 374, no. 6573, pp. 1385–1389, 2021. DOI: 10.1126/science.abj6511. eprint: https://www.science.org/doi/pdf/10.1126/science.abj6511. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.abj6511.

[5] P. Giannozzi, S. Baroni, N. Bonini, *et al.*, "QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials," *Journal of Physics: Condensed Matter*, vol. 21, no. 39, p. 395 502, Sep. 2009. DOI: 10.1088/0953-8984/21/39/395502. [Online]. Available: https://doi.org/10.1088/0953-8984/21/39/395502.

[6] A. Jain, G. Hautier, C. J. Moore, *et al.*, "A high-throughput infrastructure for density functional theory calculations," *Computational Materials Science*, vol. 50, no. 8, pp. 2295–2310, 2011, ISSN: 0927-0256. DOI: https://doi.org/10.1016/j.commatsci.2011.02.023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0927025611001133.

[7] "Configuration (electronic)," in *The IUPAC Compendium of Chemical Terminology*, International Union of Pure and Applied Chemistry (IUPAC), Feb. 2014. DOI: 10.1351/goldbook.c01248.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016. arXiv: 1603.02754. [Online]. Available: http://arxiv.org/abs/1603.02754.

[10] G. Ke, Q. Meng, T. Finley, *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[11] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *CoRR*, vol. abs/2110.01889, 2021. arXiv: 2110.01889. [Online]. Available: https://arxiv.org/abs/2110.01889.

[12] L. Ardizzone, J. Kruse, S. Wirkert, *et al.*, "Analyzing inverse problems with invertible neural networks," Aug. 14, 2018. arXiv: http://arxiv.org/abs/1808.04730v3 [cs.LG].

Appendix

A. Description of the features

In DFT, the quantum problem is defined by the number of electrons and the external potential, which uniquely correspond to the charge density. All system properties are defined by the wave functions determined by the Schrödinger Equation. Thus, the energy and all other properties computed using it, are functionals of the charge density. To perform computations, wave functions are represented as modulated linear combinations of plane waves according to the Bloch theorem. The decomposition into a set of plane waves is done by a Fourier transformation and corresponds to the reciprocal lattice of the crystal. The size of the set of plane waves influences the accuracy of the calculations and is determined by the radius of the cut-off sphere in reciprocal space. It enters the computation in terms of two parameters: the energy cut-off for the wave functions (`ecutwfc`) and the energy cut-off for the charge density (`ecutrho`). The higher these parameters are, the better is the accuracy and the higher is the computational complexity. To compute properties such as electron density, we have to integrate over the Brillouin zone. In order to perform this numerical integration, we need to use a mesh to sample the Brillouin zone. The number of the mesh points along each direction of the Brillouin zone (also called k-point grid) is passed as a parameter in the computation input file (nk_1, nk_2, nk_3). To ensure comparability between different systems, the density of the mesh was used as a feature to train our models (`k_density`).

B. Figures

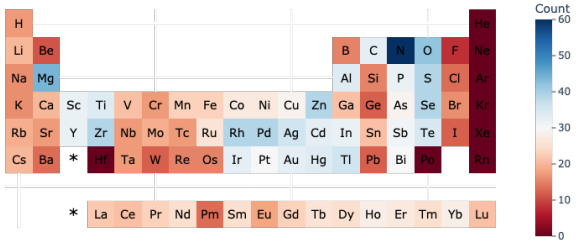

Fig. 1: Count of how many times an element appears in the dataset as part of a chemical structure.

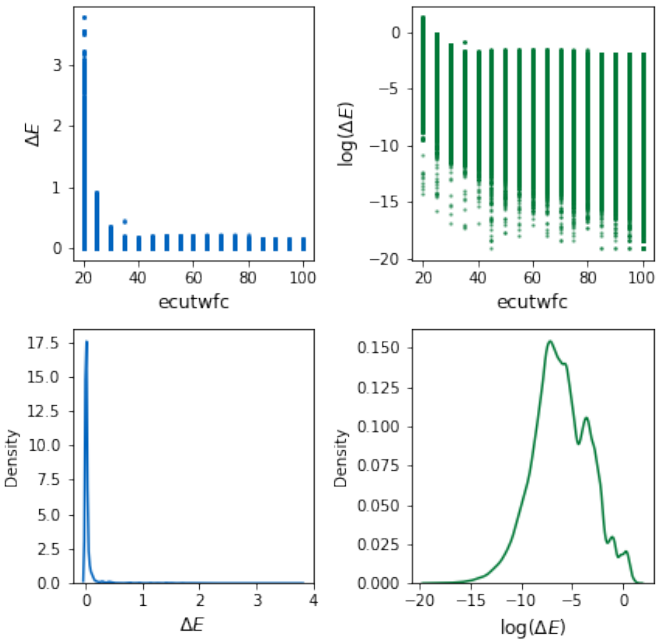

Fig. 2: Above: The distribution of the error ΔE and $\log(\Delta E)$ as a function of the parameter `ecutwfc`, respectively. Below: The corresponding density functions.

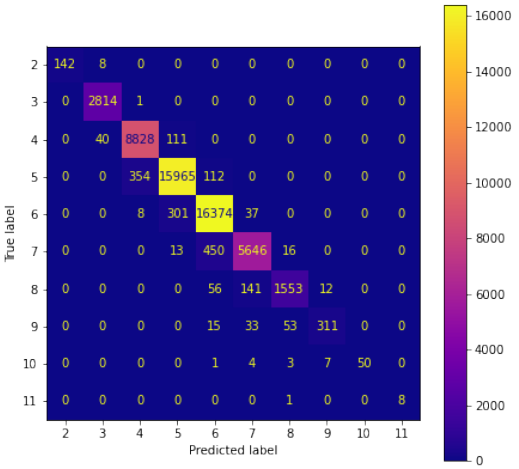

Fig. 4: Confusion matrix for the model predicting the negative order of magnitude of the energy accuracy ΔE.

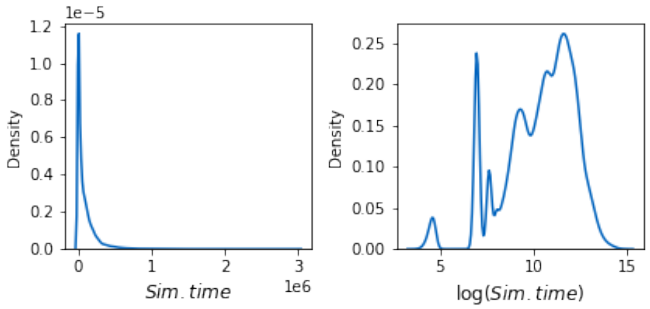

Fig. 3: The distribution of the simulation time and its log value.

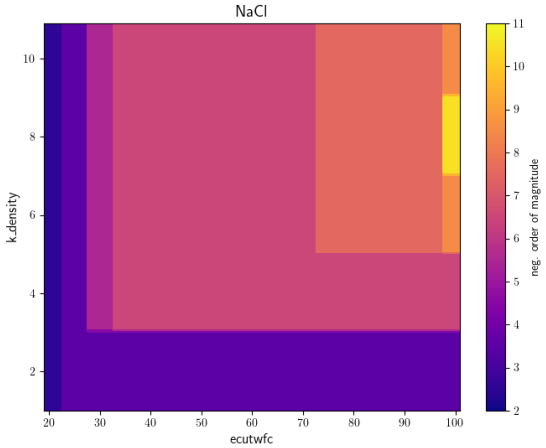

Fig. 5: The decision boundaries for the structure NaCl with dual factor 4, i.e. `ecutrho` = 4× `ecutwfc`.

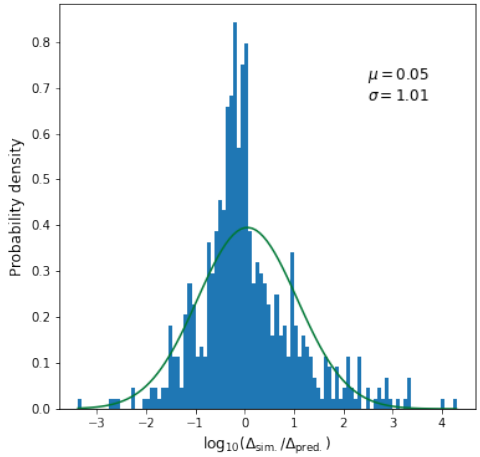

Fig. 6: Distribution of $\log_{10}(\Delta E_{\text{sim.}}/\Delta E_{\text{pred.}})$ for the set of predicted parameters together with Gaussian fit.

Test samples - Parameter gen.

Real	Dummy	Linear	Random Forest	XGBoost
7.372E-05	2.895E-04	5.832E-04	7.367E-05	3.152E-04
1.439E-05	2.895E-04	1.856E-04	1.436E-05	7.877E-05
2.549E-05	2.895E-04	5.398E-04	2.547E-05	7.164E-05
5.326E-07	2.895E-04	1.027E-04	4.753E-07	-4.113E-06
3.609E-05	2.895E-04	-1.808E-04	3.640E-05	3.654E-05
2.392E-05	2.895E-04	-1.231E-04	2.393E-05	6.109E-06
1.072E-08	2.895E-04	-4.034E-06	2.841E-09	-1.061E-05
2.554E-06	2.895E-04	1.436E-04	2.560E-06	-9.656E-06
5.155E-07	2.895E-04	-1.150E-04	5.115E-07	5.692E-06
6.055E-06	2.895E-04	3.093E-04	6.105E-06	2.551E-05

Test samples - Structure gen.

Real	Dummy	Linear	Random Forest	XGBoost
8.239E-06	2.895E-04	1.584E-04	7.949E-06	3.138E-05
8.276E-05	2.895E-04	5.767E-04	4.867E-05	2.311E-04
5.335E-06	2.895E-04	4.959E-04	6.055E-06	1.001E-05
4.270E-04	2.895E-04	8.433E-04	4.862E-05	4.461E-04
9.273E-04	2.895E-04	9.194E-04	9.163E-05	1.884E-04
4.050E-07	2.895E-04	3.863E-05	4.410E-07	1.290E-05
1.226E-07	2.895E-04	-1.498E-04	8.881E-07	-3.576E-07
8.365E-06	2.895E-04	-1.099E-05	8.043E-06	-1.251E-04
7.935E-04	2.895E-04	3.099E-04	7.061E-05	3.195E-04
3.328E-04	2.895E-04	3.212E-04	1.646E-04	1.565E-04

Fig. 7: Prediction samples from the models trained on predicting ΔE.

C. Tables

	Dummy		
	Train	*Test - Param. gen.*	*Test - Struct. gen.*
MSE - log	7.90E+00	7.84E+00	7.74E+00
MAE - log	2.23E+00	2.22E+00	2.20E+00
MAPE - log	3.59E-01	3.58E-01	3.71E-01
MSE	5.12E-02	5.21E-02	6.59E-02
MAE	5.38E-02	5.34E-02	6.32E-02
MAPE	1.39E+02	1.31E+02	1.10E+02
	Linear Regression		
	Train	*Test - Param. gen.*	*Test - Struct. gen.*
MSE - log	3.34E+00	3.28E+00	3.31E+00
MAE - log	1.40E+00	1.39E+00	1.38E+00
MAPE - log	2.15E-01	2.14E-01	2.24E-01
MSE	4.40E-02	4.46E-02	5.73E-02
MAE	4.88E-02	4.84E-02	5.81E-02
MAPE	1.38E+01	9.93E+00	2.10E+01
	Random Forest		
	Train	*Test - Param. gen.*	*Test - Struct. gen.*
MSE - log	**1.45E-02**	**9.14E-02**	1.40E+00
MAE - log	**2.74E-02**	**7.11E-02**	**7.56E-01**
MAPE - log	**2.41E-03**	**6.32E-03**	**8.65E-02**
MSE	**7.55E-07**	**2.58E-06**	**2.58E-03**
MAE	**6.13E-05**	**1.45E-04**	**1.04E-02**
MAPE	**3.02E-02**	**1.75E-01**	**3.08E+01**
	XGBoost		
	Train	*Test - Param. gen.*	*Test - Struct. gen.*
MSE - log	1.08E-01	1.67E-01	**1.33E+00**
MAE - log	1.91E-01	2.22E-01	7.95E-01
MAPE - log	2.29E-02	2.62E-02	1.03E-01
MSE	1.27E-03	1.72E-03	9.93E-02
MAE	6.37E-03	6.92E-03	4.41E-02
MAPE	2.72E-01	3.98E-01	8.91E+01

Tab. 1: Results for the regression models trained on the *log*-transformed target and atomic encoding. log annotated losses are computed on the log transformation of the target. For the others, the inverse transformation of the log is applied on the output of the models.

	No shuffle (Struct. gen.)		Shuffle (Param. gen.)	
	Train	*Test*	*Train*	*Test*
RF	**100.00**%	66.57%	**100.00**%	**96.91**%
XGB	84.79%	**67.45**%	83.52%	82.86%

Tab. 2: The results for the mean categorical accuracy from five-fold cross-validation, with our without shuffling of the data before performing the splits.

	Random Forest		
	`n_estimators`	`max_features`	`max_depth`
Reg. ΔE	100	sqrt	`None`
Reg. $\log(\Delta E)$	100	sqrt	`None`
Reg. sim. time	230	`auto`	40
Class. ΔE	218	`sqrt`	205
	XGBoost		
	`n_estimators`	`learning_rate`	`max_depth`
Reg. ΔE	400	1.0	7
Reg. $\log(\Delta E)$	400	1.0	7
Reg. sim. time	500	0.1	12
Class. ΔE	`default`	0.3	6

Tab. 3: The parameters used for Random Forest and XGBoost, respectively.

Dummy Regression												
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	4.88E-02	4.94E-02	5.00E-02	5.00E-02	5.11E-02	5.09E-02	4.80E-02	4.80E-02	5.70E-02	5.27E-02	5.08E-02	5.08E-02
MAE	8.32E-02	8.36E-02	8.47E-02	8.47E-02	8.46E-02	8.45E-02	8.34E-02	8.34E-02	8.96E-02	8.86E-02	8.53E-02	8.53E-02
MAPE	3.27E+03	4.50E+11	3.23E+03	3.23E+03	3.04E+03	4.46E+11	3.44E+03	3.44E+03	4.06E+03	4.39E+11	3.77E+03	3.77E+03
Linear Regression												
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	**4.16E-02**	4.33E-02	4.26E-02	1.74E+02	**4.01E-02**	4.46E-02	4.11E-02	4.08E-02	**4.21E-02**	4.58E-02	4.33E-02	4.27E-02
MAE	9.72E-02	**9.33E-02**	9.61E-02	9.64E-02	9.65E-02	**9.41E-02**	9.54E-02	9.58E-02	**9.82E-02**	9.90E-02	9.91E-02	9.72E-02
MAPE	5.18E+03	5.87E+11	**4.74E+03**	4.78E+03	5.08E+03	5.30E+11	**4.57E+03**	4.66E+03	5.58E+03	5.16E+11	**5.13E+03**	5.21E+03
Random Forest												
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	**3.04E-07**	1.07E-03	1.45E-04	6.06E-03	**2.14E-06**	2.82E-03	3.05E-04	8.34E-03	**1.97E-03**	1.47E-02	9.58E-03	8.46E-03
MAE	**4.13E-05**	3.36E-03	7.72E-04	2.03E-02	**1.14E-04**	5.49E-03	1.15E-03	2.40E-02	**1.01E-02**	2.83E-02	2.52E-02	2.56E-02
MAPE	**1.22E-01**	1.05E+09	2.09E+00	4.04E+01	**2.08E-01**	1.17E+09	6.51E-01	4.96E+01	**1.23E+01**	1.29E+10	4.38E+01	3.67E+01
XGBoost												
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	**1.14E-06**	1.13E-03	1.45E-04	6.04E-03	**2.34E-06**	2.01E-03	3.35E-04	8.28E-03	**1.37E-03**	1.64E-02	1.25E-02	8.51E-03
MAE	**5.23E-04**	5.22E-03	1.24E-03	2.04E-02	**5.87E-04**	6.31E-03	1.64E-03	2.33E-02	**7.60E-03**	3.46E-02	2.89E-02	2.56E-02
MAPE	**1.03E+01**	7.00E+09	1.16E+01	4.98E+01	1.19E+01	5.64E+09	**1.02E+01**	5.16E+01	**3.26E+01**	4.93E+10	1.74E+02	4.89E+01

Tab. 4: Results for the regression models predicting ΔE for all four encodings.

	Dummy Regression											
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	2.21E+10	2.37E+10	2.21E+10	2.21E+10	2.25E+10	2.36E+10	2.25E+10	2.25E+10	2.71E+10	1.51E+10	2.71E+10	2.71E+10
MAE	9.31E+04	9.60E+04	9.31E+04	9.31E+04	9.38E+04	9.58E+04	9.38E+04	9.38E+04	9.81E+04	8.50E+04	9.81E+04	9.81E+04
MAPE	3.29E+01	3.26E+01	3.29E+01	3.29E+01	3.29E+01	3.21E+01	3.29E+01	3.29E+04	2.73E+01	4.01E+01	2.73E+01	2.73E+01
	Linear Regression											
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	**9.36E+09**	1.39E+10	1.08E+10	1.14E+10	**9.56E+09**	1.39E+10	1.10E+10	1.17E+10	1.14E+10	**8.20E+09**	1.36E+10	1.44E+10
MAE	**5.69E+04**	6.55E+04	5.99E+04	6.03E+04	**5.73E+04**	6.53E+04	6.04E+04	6.07E+04	5.91E+04	**5.90E+04**	6.13E+04	6.24E+04
MAPE	3.02E+01	**1.93E+01**	2.40E+01	2.20E+01	3.05E+01	**1.92E+01**	2.41E+01	2.20E+01	2.55E+01	2.20E+01	1.99E+01	**1.75E+01**
	Random Forest											
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	**2.20E+08**	2.44E+08	5.31E+09	2.82E+08	**6.75E+08**	1.09E+09	7.01E+09	7.49E+08	5.82E+09	**3.88E+09**	9.98E+09	6.82E+09
MAE	6.08E+03	4.08E+03	2.77E+04	**3.23E+03**	9.19E+03	8.93E+03	3.25E+04	**6.87E+03**	2.74E+04	**2.68E+04**	3.80E+04	3.23E+04
MAPE	1.74E-01	7.74E-02	4.47E-01	**5.06E-02**	1.94E-01	1.40E-01	5.38E-01	**9.10E-02**	**3.58E-01**	5.43E-01	5.96E-01	5.55E-01
	XGBoost											
	Train				*Test 1 - Parameter generalization*				*Test 2 - Structure generalization*			
	A	VC	CM	C	A	VC	CM	C	A	VC	CM	C
MSE	**1.15E+08**	2.44E+08	5.31E+09	3.84E+08	**5.14E+08**	1.01E+09	6.90E+09	7.26E+08	4.37E+09	**3.69E+09**	9.98E+09	7.17E+09
MAE	**4.01E+03**	5.02E+03	2.77E+04	5.62E+03	**6.42E+03**	9.08E+03	3.17E+04	7.63E+03	**2.23E+04**	2.48E+04	3.80E+04	3.31E+04
MAPE	1.59E-01	**1.47E-01**	4.56E-01	1.53E-01	1.77E-01	1.85E-01	5.07E-01	**1.70E-01**	**3.11E-01**	5.62E-01	6.00E-01	6.83E-01

Tab. 5: Results for the regression models predicting the simulation time for all four encodings.