Machine learning for quantum chemistry: learning chemical properties with minimal supervision

Beryl Aribowo
beryl.aribowo@univie.ac.at

Arnold Neumaier
arnold.neumaier@univie.ac.at

O. Anatole von Lilienfeld
anatole.vonlilienfeld@utoronto.ca

July 16, 2024

Abstract

Machine learning methods have often been shown to reach chemical accuracy given sufficient training data at a fraction of the computational cost associated with *ab initio* methods. However, reaching chemical accuracy with only a small amount of molecules for training remains a non–trivial task. Here, the main quest is based on the QM9 challenge [1]: predicting the energies of 130k molecules such that chemical accuracy is achieved using only 100 molecules for training. This challenge can be tackled from many directions, including devising machine learning model ansatz, designing chemical features, developing more robust optimization algorithms, and so on. In this work, mainly three procedures are explored: initial training set selection algorithm, pre–fitting delta machine learning, and post–fitting training set optimization. An initial training set selection algorithm based on the farthest point strategy is used to select the most relevant data points for training. Delta machine learning with energies derived from graph–based features serves to reduce the errors prior to fitting with higher–resolution chemical features. A combinatorial optimization algorithm based on Tabu search is employed with the purpose of finding the optimal training set given a set of pre-optimized model and feature hyperparameters. Here we highlight the aforementioned procedures that led to our current record of 3.25 kcal/mol MAE for the prediction of 130k out of sample molecules using only 100 molecules ($< 0.1\%$ of total amount of molecules) for training. Given the extracted features, the model with best found parameters combination needs only ~ 2.9 seconds to evaluate all 130k molecules.

1 Introduction

Chemical information contains useful properties (quantities) for important applications such as protein structure prediction and molecular docking (e.g., for drug discovery). One such property is the energy of the chemical system that can be determined given the structure of the system. Traditionally, these properties can be predicted by *ab initio* methods accurately. However, *ab initio* methods have a massive drawback in terms of computational cost, such that it is intractable for large chemical systems [2]. On the other side of the spectrum, force field methods exist – cheap but less accurate and often system specific [3].

With this regard, machine learning enters with the promise of the best of both worlds. In hindsight, the advancement of machine learning for chemical science can be categorized into two major approaches (in principle, this is applicable for any machine learning applications in general):

(i). Designing a set of atomic or molecular **features** (or often referred as atomic or molecular **representation**), i.e., chemical data feature engineering. Chemical data feature engineering mostly requires specific domain knowledge in physics and chemistry rather than generic modelling abstraction, although arguably some components can be promoted into generic abstractions to some degree. Several ready–to–use prominent atomic features as software libraries including but not limited to [4–7]. These features incorporate symmetry such that they are invariant when the molecular coordinates are translated, rotated, or when atoms with equal type are permuted [8].

(ii). Designing the fitting model; in contrast to the features, this is more of a generic approach (applicable to domains outside of chemical science too). Well known fitting models for chemical quantity regressions ranging from linear models to the neural networks [9–11]. Often, specific domain knowledge of the problem to be solved is not required to design such models. Usually the techniques are abstracted in more generic forms that are common in mathematics and computer science disciplines. There can also be a case where a fitting model were devised from the necessity of solving domain specific problem, one of the most recent well known model of this case is the message passing neural network (MPNN) [12].

Some approaches have more interconnected relationship between the feature and the fitting model, e.g., [13] devised the symmetric atomic feature alongside a neural network that is symmetric with regard to the atomic energy, and [14] incorporated symmetries in the structure of the network without the need of any prior feature extraction (analogous to deep neural networks for general problems, they do not need intensively engineered features as input). All of the aforementioned techniques provide a zoo of feature–model combinations that can be utilized and tailored according to problem's requirement. Something to note is that the term "model" here can also refer to the combined feature and fitting model themselves, depending on the context; as it is also a common practice in mathematical modelling to distinguish the ansatz and formulas which is aggregated as the "model," from the algorithms or ways to find the solutions which is often dubbed as the "solver".

The quality of machine learning techniques are often judged by the rate of decay of the prediction errors. In expectation, the prediction errors decay as the number of data used for training is larger. While machine learning techniques themselves are known to be fast, the accurate data from *ab initio* techniques that are used for the training process do not come for cheap. This suggests that machine learning methods with efficient data usage for training are desired. Furthermore, aside from the obvious speed up in computation due to lower amount of data to be processed for training process, accurate data efficient techniques would also imply better generalization, model transferability would not be out of the realm of possibility.

A machine learning for quantum chemistry challenge, dubbed the **QM9 challenge**, was proposed by [1] and then reiterated in [3,15,16]. The challenge uses the QM9 dataset [17] for the benchmark. The QM9 dataset contains the geometries of 134k equilibrium molecules containing up to 9 heavy atoms (C, N, O, F) excluding H. This challenge addresses the goal of reaching chemical accuracy: 1 kcal/mol, with only 100 molecules for training, i.e., less than 0.1% of the total number of molecules in the QM9 dataset (and the rest of the 130k molecules are for testing). This challenge sets a fitting stage and goal for this work.

In this work, we address the paths to find the optimal combination of techniques with the spirit of reaching chemical accuracy using only 100 molecules for training. Pipeline–wise, the initial procedure is composed of a way to select the most relevant 100 molecules from all of the 134k molecules; the discussion is mainly based on a sampling algorithm: farthest point sampling (FPS) [18]; in particular the stochastic adaptation: `usequence` [19]. The fitting itself is split into three main procedures: pre–fitting with **delta machine learning (ΔML)**, the main procedure of fitting with feature–model combination, and post–fitting optimization; lastly, hyperparameter optimization as an additional component that overarches most of the computational parts.

The ΔML procedure has the purpose of reducing the fitting load. Although here the ΔML for pre–fitting itself is technically a fitting mechanism, albeit simpler. An expansion of the many–body in terms of graph representation of molecules, dubbed as dressed [atom, bond, angle, torsion] (DA, DB, DN, DT), analogous to "bonds, angles, machine learning" (BAML) [20] and "universal force fields" (UFF) [21], are used as the main sets of features for the ΔML pre–fitting procedure. Typically, ΔML mechanism is used to learn the correction (i.e., difference) between two levels of accuracy of the same quantum property [22–24], this implies at least two sets of quantum property targets computed from *ab initio* methods are used. However in this work, we make use only one level of accuracy of a quantum property directly from the QM9 dataset, since loosely speaking, using multiple sets of targets from many sources contradicts the spirit of this work, and more importantly it violates one of the QM9 challenge constraint [1]. Therefore here we use the ΔML to learn the levels of quantum property recursively derived from only a level of quantum property, without additional *ab initio* calculation or resources.

Content: Section 2 discusses the theories, mainly the succinct definition of the ΔML, the molecule selection techniques, and the fitting procedure. Section 3 encapsulates the computational results and discussion, a few benchmarks for "sanity check" are done in the earlier subsections, then experiments that are more relevant to the QM9 challenge are done in the hyperparameter optimization subsection and the optimal training set selection subsection. Additionally a subsection containing analysis of the delta learning level and is provided. Section 4 analyzes the distribution of the molecules based on the best generated model by the means of visual representation.

2 Theory

2.1 Delta machine learning (ΔML)

The crux of ΔML is to take out as much low–level energy as possible before learning the rest of the high–level energy, i,e., to reduce the workload of the machine to learn the complicated relation between the molecular geometries and the high–level energies. As described by Zaspel et al [23], the general form of the ΔML can be formalised as

$$E^l = E^{l-1} + \mathtt{ML}^l(\theta^l, F^l), \quad E^0 = \mathtt{ML}^0(\theta^0, F^0) \tag{2.1.1}$$

where $\mathtt{ML}^l$ is a function to learn and predict $E_t^l := E_t - E^{l-1}$, which is the difference between two energy levels; where E_t is the target energy, θ^l is the set of parameters and F^l is the set of features. This recipe implies a multi–level recursive learning scheme for different levels of energies. Given a set of energy levels $l = 1, .., n$, we call all levels $l < n$ as **baselines**. In general it is expected that the lowest level of energy will have the largest magnitude, with decaying magnitude of energy as we traverse up the stairs of energy levels (for more details on the baseline feature extractions refer to supplementary material).

2.2 Fitting technique

The fitting process essentially tries to minimize the error between the actual energies and the predicted energies, the assessment of error used here is the MAE:

$$\text{MAE} = \frac{1}{|W|} \sum_{i \in W} |E_i - \hat{E}_i|, \tag{2.2.1}$$

where W is a set containing the indices of selected molecules in the dataset and $\hat{E}$ is the predicted energy; where the predicted energies $\hat{E}$ is the output of a fitted function

$$\hat{E} = f(\mathbf{x}, \theta), \tag{2.2.2}$$

where $\mathbf{x}$ is the collection of features of a dataset, and θ is the regression coefficients.

Here the fitting problems are formulated into linear systems of equations, which can be solved in a simpler way compared to nonlinear systems. In particular, kernel models are used.

By using the kernel models, nonlinear set of features can be represented in a linear way. Given a pair of features $u, v \in \mathbb{R}^m$, a kernel function $\kappa(u, v)$ maps the pair into a scalar. The kernel function is then used to form a kernel matrix $K = (K_{ij})$ which contains in each row molecule's condensed information. Let $X \in \mathbb{R}^{n \times m}$ be a matrix where each row is a molecular feature vector, then the entries of the kernel matrix are

$$K_{ij} := \kappa(X_{i:}, X_{j:}). \tag{2.2.3}$$

Given E and K formed from the training set, the regression coefficients θ in (2.2.2) can be found by a closed form solution if K is invertible

$$\theta = (K + \lambda I)^{-1} E, \tag{2.2.4}$$

where λ is a regularization constant. Typically in practice the solution is computed by row reduction techniques or by numerical linear solvers instead of explicitly inverting K for general K.

Given θ and K, we can rewrite the function in (2.2.2) to predict the energy of a molecule indexed by i into

$$f(\mathbf{x}_i, \theta) := \sum_j K_{ij}\theta_j, \tag{2.2.5}$$

where j runs over the indices of molecules in the training set.

Two types of kernel functions are used, the Gaussian kernel:

$$\kappa(u, v) = \exp\Big(-\frac{\|u-v\|_2^2}{2\sigma^2}\Big), \tag{2.2.6}$$

where the width of the Gaussian σ is a hyperparameter; and the linear kernel:

$$\kappa(u, v) = u^T v. \tag{2.2.7}$$

Typically, the set of features are available on the atomic level, in this case, the entries are

$$K_{ij} := \sum_l \kappa(x^i_{l:}, x^j_{l:}), \tag{2.2.8}$$

where $x^i_{l:}$ is a row vector containing atomic features of an atom indexed by l in a molecule indexed by i; this ansatz gives the "elemental screening" effect, i.e., it makes sure that only the same atomic environments are compared, which leads to faster computation speed and higher accuracy [25].

2.3 Data selection technique

2.3.1 Unsupervised selection technique

One of the biggest hurdle in the QM9 challenge is that the training set size is limited to 100, which means less than 0.1% of the total 134k is used. However it is allowed to hand–pick the data points of the training set such that it gives better model generalization for the rest of the test–set (often also called as out–of–sample set). Here we employ heuristics to select the relevant data points for the training set, which is an adaptation of the farthest point strategy (FPS) by Eldar et al [18].

In essence, FPS is a sampling technique that preserves the uniformity of the sparse samples of the global set. An algorithm dubbed `usequence` [19], is an adaptation of the FPS with additional stochastic feature.

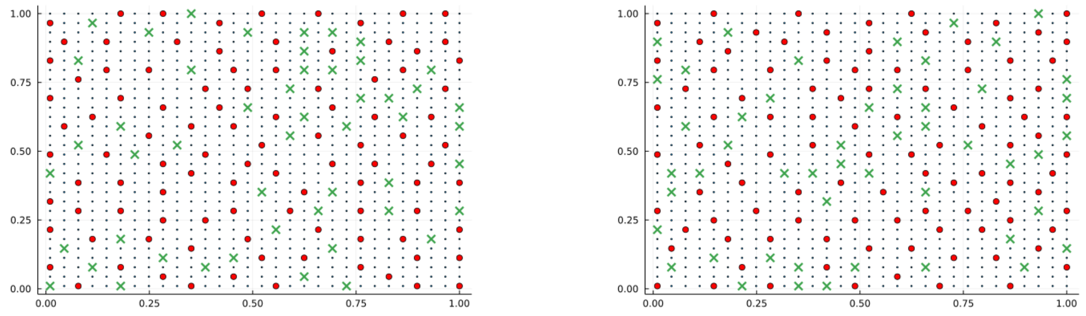

Figure 1: A grid of 900 points in 2D space, red dots are the first 100 points selected by usequence, green crosses are the next 50 points selected. Two sets (left and right image) of selection are generated from the same grid.

The `usequence` works on the molecular feature level. We can transform atomic feature vectors of a molecule into molecular feature vector in many ways. One way is to compute the sums of the atomic features sorted by the atomic types: given the ith molecule which contains number of atoms n_{atom} and a set of atom types $A = \{\text{H}, \text{C}, \text{O}, ...\}$ with cardinality n_{type}, a partition of the molecular feature is computed by

$$s^i_j = \sum_{l \in M_j} x^i_{l:} \tag{2.3.1}$$

where M_j is a set of indices which contains the row positions of atom type $j \in A$ in the x^l matrix. These partitions are then concatenated into the molecular feature vector of the ith molecule:

$$X_i := [s^i_1, ..., s^i_{n_{\text{type}}}]. \tag{2.3.2}$$

2.3.2 Supervised selection technique

Aside from unsupervised way of training–set selection using the `usequence`, here we also conduct the training–set selection experiments in a supervised manner, in which the energy is included as a component of the objective function to be minimized. The goal is to minimize an objective function where the training indices act as the decision variables. Generally, this can be categorized as a combinatorial (or discrete) optimization. Here we formulate the problem as

$$\begin{aligned} \text{min} \quad & f(x) \\ \text{subject to} \quad & c(x) = 0 \end{aligned} \tag{2.3.3}$$

where

$$f(x) = \frac{1}{N_{\text{test}}} \sum_k \left| \left[D(\neg x) K D(x) \Big(D(x) K D(x) \Big)^{-1} \Big(D(x) E \Big) \right]_k - \Big[D(\neg x) E \Big]_k \right| \tag{2.3.4}$$

is the nonlinear objective function to be minimized, we can also notice that in fact $f(x)$ is equivalent to the MAE.

$$c(x) = \sum_i x_i - 100, \quad x_i \in \{0, 1\}, \quad i \in [1, 2, ...N_{QM9}] \tag{2.3.5}$$

is the linear equality constraint to be fulfilled and x is a one–hot encoding vector of the training indices sorted by the label of the data where $x_i = 1$ if the data with label i is selected and $x_i = 0$ otherwise; and N_{QM9} is the size of the QM9 dataset.

$$D(x) \in \{0, 1\}^{N_{QM9} \times N_{QM9}}, \quad D(x)_{ii} := x_i \tag{2.3.6}$$

is a matrix that selects the rows and columns of K, where

$$K := K(H) \tag{2.3.7}$$

is the kernel matrix formed by the fixed hyperparameters H, where H should ideally be the best found hyperparameters.

We want to emphasize that (2.3.3 - 2.3.7) are only the formal mathematical representation, in the program implementation it is not desirable to compute $f(x)$ in that particular way due to inefficient (time and memory wise) K and $D(\cdot)$ computation. Slicing the rows and columns of K by x at the same time as it is being computed is the preferred way, this implies black–box optimization problem.

Selecting 100 training data from 130k data points that minimizes (2.3.3) is untractable, due to the very large number of possible combinations and due to the nature of discrete optimization where "descend directions" are not available (unlike continuous optimization). Therefore we rely on a heuristic, we devise an algorithm based on the Tabu search (algorithm 1). Essentially Tabu search eliminates penalized candidates, where the penalty scheme itself can be in many different forms depending on problem context [26].

Algorithm 1 Tabu search for training–set optimization

Input:

- a vector $p^{(0)}$ of initial penalty values of all data points.
- a set of training sets $P^{(0)}$ selected during the process of generating $p^{(0)}$.
- hyperparameters:
  - size n of best training sets to be selected from the global set $P^{(0)}$.
  - numbers m of training sets x to be replaced on each iteration.
- reset criteria, e.g,. number of iterations, rate of decay of objective value, etc.
- stopping criteria, e.g., number of iterations, objective value threshold, etc

Output: best obtained training set x^* and its corresponding objective value f^*

```
1: Pick $P \subset P^{(0)}$ where $|P| = n$ that have the lowest objective values.
2: while stopping criteria are not met do
3:     while reset criteria are not met do
4:         Pick $x \in P$ randomly.
5:         Replace $m$ numbers of data points $s \in x$ that have high penalty values with $s \notin x$ that have low penalty values.
6:         Evaluate $f(x)$ and update the penalty value $p(s) = \frac{g(s)+f^*}{h(s)+c}$, $\forall s \in x$ where $g(s)$ is the sum of all objective values of the simulations where $s$ is involved, $f^*$ is the currently best found objective value, $h(s)$ is the total number of simulations where $s$ is involved, and $c$ is a small positive constant, here we pick $c = 1$.
7:     end while
8:     Reinitialize $P$, and/or change the values of $m$ and/or $n$.
9: end while
```

3 Results & discussion

Experiment scenarios consist mostly of the enumeration of combination of features, models, energy levels, and training set selections in order to bring down the errors (or objective functions) to be as low as possible.

3.1 Computational details

3.1.1 Dataset

The QM9 dataset [17] is used for the benchmarks, in particular the pre–cleaned version which was released by RUPP [27]. QM9 dataset contains 134k organic molecules in ground states composed of `atom_types = [H,C,N,O,F]`, with several properties computed from quantum chemistry, one of which is the internal energy at 0 K U_0 property is the target of our interest.

The dataset is further cleaned, 3054 uncharacterized molecules and 11 molecules with geometries that are difficult to converge are removed [27], this procedure results in **130823** molecules. The pre–processed QM9 dataset is available in the supplementary material.

3.1.2 Setup

Throughout all of the scenarios of the numerical experiments, the hyperparameter space tested will be mostly the same, although it is possible that there will be some additional or removal of the hyperparameters' components depending on the scenario which will be described exclusively within the section that correspond to the said scenario.

The hyperparameters that are commonly used throughout the numerical experiments:

- `ntrain` $= 100$. The number of molecules included in the training set; in particular, the main interest is in `ntrain` $= 100$ (as described in the QM9 challenge). Unless specified within the relevant section, `ntrain` $= 100$ is fixed. All of the MAEs shown in the plots are the MAEs of the out–of–sample sets (test sets) rather than the training sets', unless specified otherwise.

- `feature` $\in$ {ACSF, SOAP, FCHL19, MBDF, CMBDF}. The type of atomic features used. ACSF refers to the atomic centered symmetry function [4]. SOAP refers to the smooth overlap of atomic positions [5]. FCHL19 refers to Faber–Christensen–Huang–Lilienfeld [25]. MBDF refers to many–body–density–functional [7], and CMBDF is its convolutional variant [28].

  - ACSF is extracted using `DescriptorZoo` Julia package using the default hyperparameters available in the package [29, 30], this results in atomic feature vectors where each vector has 51 entries.
  - SOAP is extracted using `DScribe` Python package [31], the hyperparameters used are (`rcut, nmax, lmax, sigma`) $= (6, 3, 3, 0.1)$ [7], this leads to each atomic vector having 480 entries.
  - FCHL19 is extracted using the development branch of `QML` Python package [32], the hyperparameters used are (`rcut, nRs2, nRs3`) $= (6., 12, 10)$ [7], this gives each atomic feature vector 360 entries.
  - MBDF and CMBDF are extracted using the default parameters embedded in the program package [28], results in atomic feature vector with 40 entries.

- `model` $\in$ {GK, DPK}. GK refers to Gaussian kernel function in (2.2.6), here $\sigma = 32$ is chosen as default, unless stated otherwise. DPK refers to the linear kernel in (2.2.7).

- `solver` $\in$ {CGLS, direct}. Direct refers to an ensemble of row reduction techniques embedded in the backslash operator "\" of Julia programming language [33]; while CGLS refers to conjugate gradient for linear least squares problem [34–36]. For both solver, regularization variable $\lambda = 1e-8$ is used as the default value, and CGLS is limited to 500 iterations.

- `elevel` $\in$ {`A, AB, ABN, ABNT`} or equivalently `elevel` $\in$ {DA, DB, DN, DT}. The energy levels after reduction by the baselines. DA refers to only dressed atoms as baseline, DB refers to both dressed atoms and dressed bonds as baselines, DN refers to baselines up to dressed angles, while DT refers to baselines up to dressed torsions.

3.2 Benchmark: baseline learning curves

As a sanity check (to see whether the error curves decay or not), a batch of numerical experiments are done by fixing `100k` data points as test set and the rest as training sets tested at different sizes `ntrain = [100, 200, 500, 1k, 2k, 4k, 8k, 16k, 30k]`. The delta–levels include dressed atom up to dressed torsion `[A, AB, ABN, ABNT]`. The variations in the data selection methods are `[rand, sid57]`, that correspond to random selection and the identifier of one of the selected training sets generated by `usequence` respectively; `sid57` set was generated by the molecular features computed from the atomic features of CMBDF. An additional hyperparameter is also introduced: the presence of **orbital hybridization**, `hybrid = [true, false]` (see supplementary material) for the description of the hybrid classes).

Generally, features with higher resolution have higher variance and lower bias, and the opposite is also true. Moreover, features with high variance tend to have high error with less data points included for training and becomes more accurate with higher number of data points included for training, while features with low variance does not become more accurate with addition of training data points; as shown in Figure 2, `ABN` and `ABNT` curves have lower MAEs (higher accuracy) as `Ntrain` increases, while `A` and `AB` are stagnant.

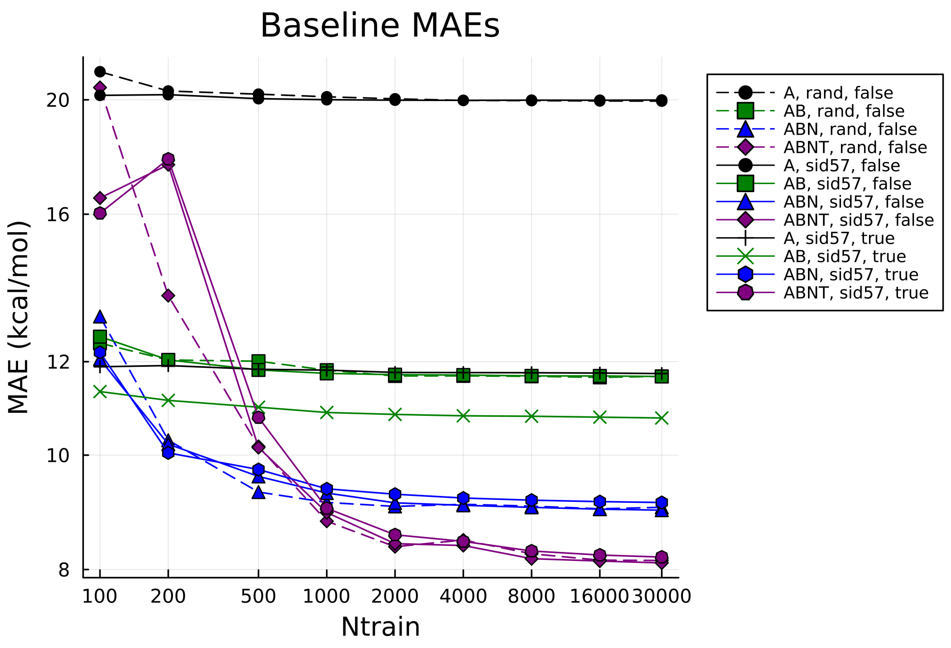

Figure 2: MAE of the dressed features in increasing delta–levels `[A, AB, ABN, ABNT]`, with variations of training set selection `[rand, sid57]`, and the presence of hybridization `[true, false]`.

3.3 Benchmark: training with higher resolution features

3.3.1 Learning curves with `ntrain` ≥ 100

In this section we want to see what the learning curves for the higher resolution feature look like when `ntrain` $\geq$ 100, another sanity check to see whether the prediction accuracy can actually reach 1 kcal/mol or lower. Similar to section 3.2 we fix `100k` data as test set and the rest as the training data where they are reorganized into many different sizes `ntrain`.

For the setup, we enumerate the hyperparameters:

- `elevel` $\in$ {`A, AB, ABN, ABNT`},
- `model` $\in$ {GK, DPK},
- `hybrid` $\in$ {`true, false`},
- $\sigma \in \{1, 2, .., 32\}$,

the hyperparameter `solver = direct` is kept fixed.

The benchmark between `rand` and `usequence` is shown in Figure 3.3.1, the curves are generated using the lowest MAE from all of the possible hyperparameters for each combination of atomic feature, data selection method, and `ntrain`. The hyperparameters are tuned in two steps: first, the `elevel`, `model`, and `hybrid` parameters are optimized; then on the second step, only σ is optimized, this still guarantees minima while reducing the parameter space that results in considerable total computation time speedup.

We can observe that for lower `ntrain`, proper training data selection is more relevant, while on larger `ntrain` typically there are almost no difference in accuracy between `rand` and `usequence`, this is expected since the distribution of the training data becomes more dense hence the accuracy is more robust with respect to the composition of the training data, in contrast to lower `ntrain` where the MAE is more sensitive to the composition of training data. On the point of interest, i.e., `ntrain` = 100 (even though this benchmark does not necessarily fulfill the QM9 challenge since the test set size is not `130k`), CMBDF has the lowest MAE at **3.71 kcal/mol**.

The total computation timing depends on the size atomic feature, which is also a relevant quantity of interest. Table 1 shows the mean of the sum of the kernel computation time, training time, and evaluation time for each atomic feature on all `ntrain` sizes.

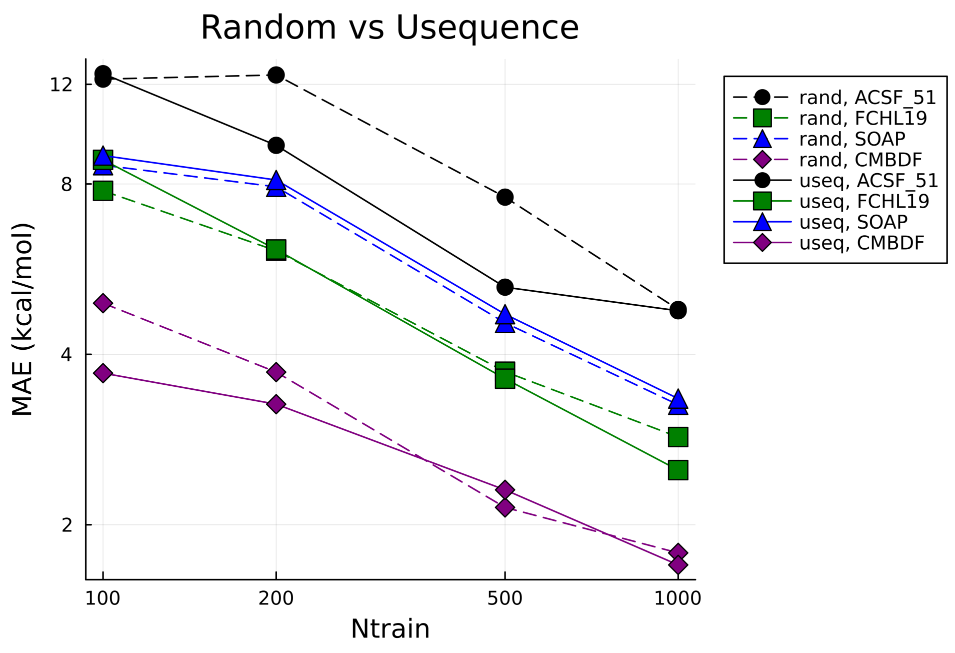

Figure 3: Benchmark of `rand` vs `usequence` (represented by sequence `sid57`) for`ntrain` up to 1k. MAEs are evaluated on 100k test molecules.

Table 1: Mean of the total time (in seconds) of the kernel model computation, training, and evaluation using 100k test molecules for each feature on each `ntrain`.

ntrain	**Feature**			
	ACSF	FCHL19	SOAP	CMBDF
100	2.542	16.766	21.027	1.837
200	4.123	32.680	41.401	3.321
500	10.719	79.640	101.847	8.362
1000	20.149	155.616	207.597	16.796

3.3.2 Hypertuning intensification for `ntrain` = 100

With respect to the QM9 challenge, the main goal is reaching chemical accuracy (or lower) with only 100 training data ($< 0.1\%$ of total data), we want to find the most optimal configuration for the fitting procedure. Here we attempt to optimize the hyperparameters that are listed in section 3.1.2 with the addition of a few extra parameters, by setting the MAE as the objective function in a minimization problem (the complete list of the hyperparameters is available in the supplementary material).

We utilize a black box mixed integer optimizer, **matrix adaptation trust-region strategy** (MATRS) [19, 37] for the hypertuning. The hyperparameter optimization gives varying results depending on how close the objective function is to the chemical accuracy target — the closer it is, the less gain the hyperparameter optimization gives. The summary of the hypertuning results is in Table 2.

Chronologically, the first run of the hyperparameter optimization did not include MBDF and CMBDF, as both of them were not available yet; accuracy gain of 0.5kcal/mol were obtained, and a total of 8689 iterations were done and the minima were found in the 4085th iteration. The next experiments were done when MBDF was available; a total of 3829 iterations were done, and the minima were found in the 500th iteration; accuracy gain of 0.5kcal/mol were also obtained. Then finally with CMBDF, the lowest point were found at the 115th iteration from a total of 9824 iterations; this gives only 0.05kcal/mol accuracy gain. For each set of experiments, the starting point used is the best training set generated from `usequence`, and we fix it for all of the iterations, i.e., no change or no new training set generation during the optimization process.

Additionally, the CPU timing for each run of prediction of 130k molecules takes around **2.9 seconds** on a 64 cores CPU using the current best found hyperparameters. More details on the hardware specification used is available in the supplementary material.

Table 2: The MAE of the initial point and the minima found after hypertuning. Additionally the total iterations and the iteration in which the minimizer is found for each set of computational experiments is also reported.

	Set 1 (ACSF)		**Set 2 (MBDF)**		**Set 3 (CMBDF)**	
	start	**minima**	**start**	**minima**	**start**	**minima**
MAE	7.59	7.09	5.78	5.29	3.72	3.67
total_iter	8689		3829		9824	
minimizer_at	4085		500		115	

3.4 Optimal training set selection

In the previous sections we have seen the performance of `usequence` when used as a one–shot unsupervised data selection technique (completely detached from the computation of the objective function). Here we combine both `usequence` and supervised data selection technique in Section 2.3.2.

We use the Tabu search described in Algorithm (1) for the supervised selection process. In particular for the algorithm's configuration, the input pairs $p^{(0)}$ and $P^{(0)}$ can be obtained by setting $p^{(0)} := f(x^{(0)})$ for all available $x^{(0)}$ and storing all $x^{(0)}$ into $P^{(0)}$; where $x^{(0)}$ and $f(x^{(0)})$ can be obtained from simulation data or by spawning many random training sets and evaluating (2.3.4) for each training set (it must also comply to (2.3.5) by only selecting 100 data points for each training set). The `usequence` is used to generate $P^{(0)}$ where $|P^{(0)}| \geq 1000$ instead of randomly selecting the data points, in other words the training sets from `usequence` are the initial starting points for the optimization process. The reset mechanism is useful to avoid being trapped in local minima, usually this is done after there are no gains within some numbers of iterations. In the numerical experiments, we sample 10k data points as the test set to compute the objective function (2.3.4) instead of the whole 130k, this significantly increases the computation speed by a factor of 50 while only sacrificing the accuracy in the 0.01 kcal/mol range.

Each iteration of the Tabu search takes only **0.1 second** on average in a serial process, and only `3` $\times$ `130k` entries of `Float64` need to be kept track of by the algorithm that is equivalent to `3.2MB` (independent of the memory requirement for the models). The initial (data) points include the current best point so far and points with high penalties. Points with high penalties are included in the hope that the algorithm would learn to exclude them earlier. After around 100k iterations, the current best found point is MAE $:= f(x) = 3.214$ kcal/mol, and it gives MAE = **3.25 kcal/mol** when evaluated on 130k data points as test set. Figure 5 shows the information of the 100 selected molecules as the final solution of the optimization process. Figure 4 shows the atomization energies' distributions of the 100 selected molecules and all of the QM9 molecules.

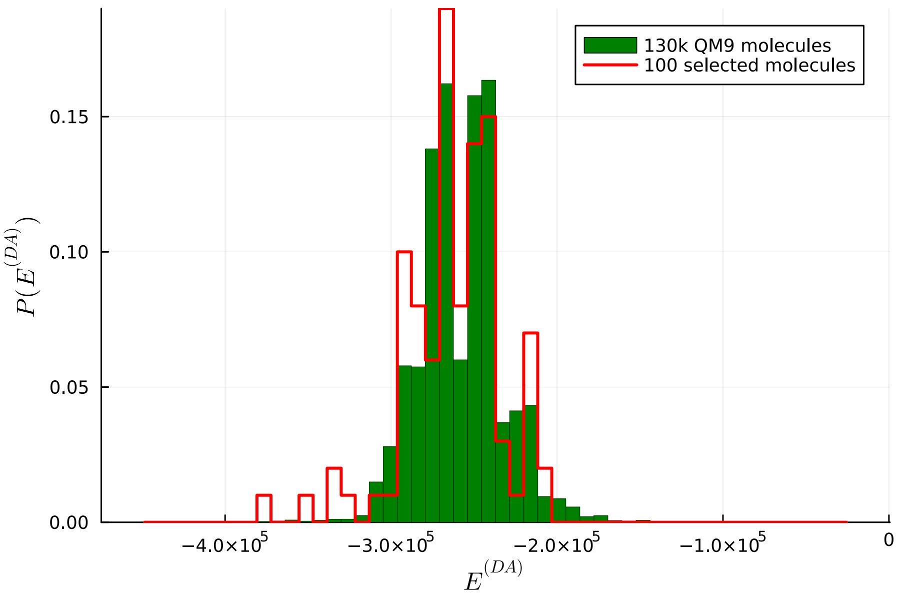

Figure 4: The atomization energies' $E^{(DA)}$ (in kcal/mol) distributions $P(E^{(DA)})$ of the 100 selected molecules and all of the 130k QM9 molecules. The distribution of each set is normalized such that the sum of the bins' height is 1.

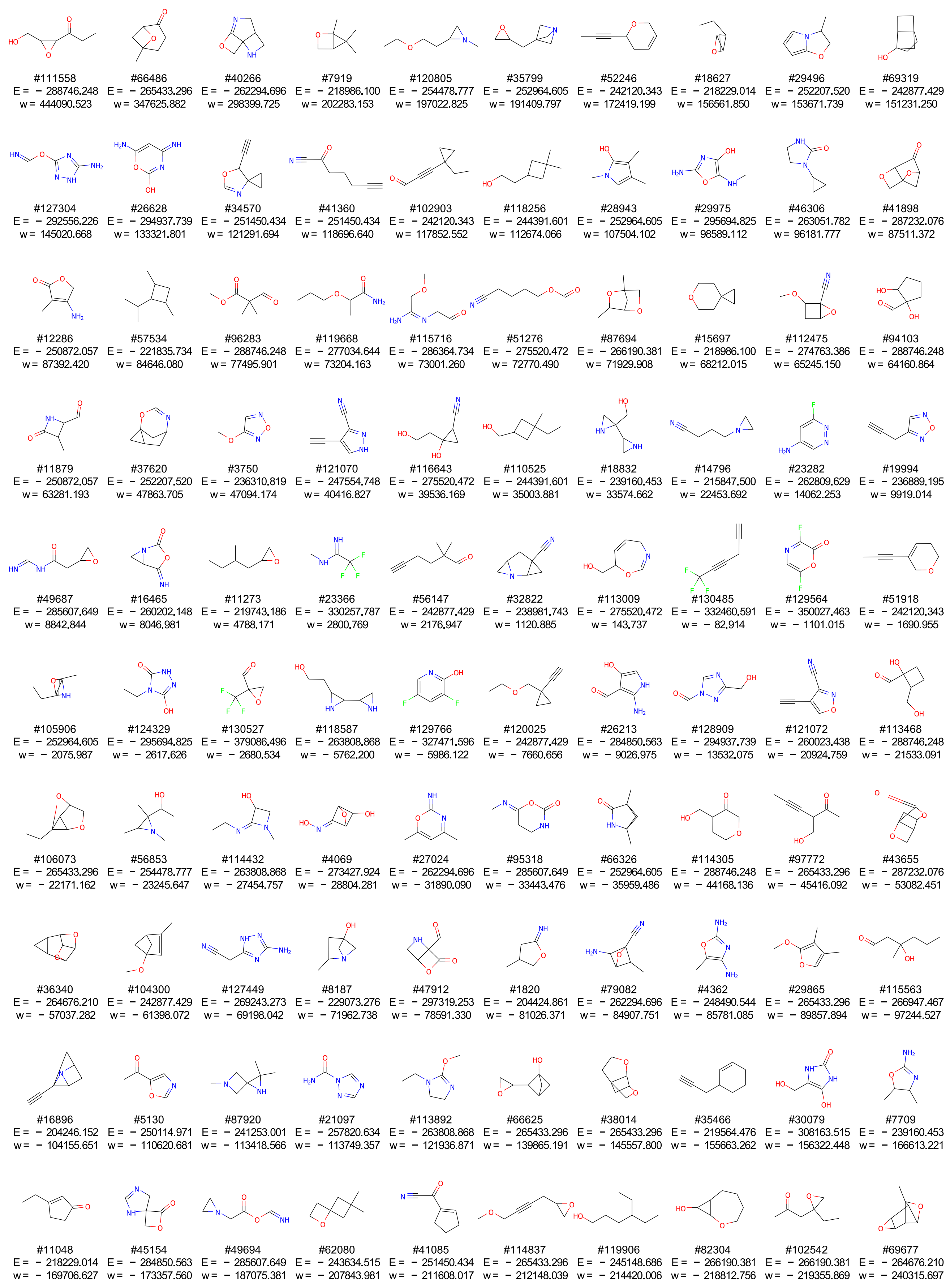
#111558 E = − 288746.248 w = 444090.523
#66486 E = − 265433.296 w = 347625.882
#40266 E = − 262294.696 w = 298399.725
#7919 E = − 218986.100 w = 202283.153
#120805 E = − 254478.777 w = 197022.825
#35799 E = − 252964.605 w = 191409.797
#52246 E = − 242120.343 w = 172419.199
#18627 E = − 218229.014 w = 156561.850
#29496 E = − 252207.520 w = 153671.739
#69319 E = − 242877.429 w = 151231.250
#127304 E = − 292556.226 w = 145020.668
#26628 E = − 294937.739 w = 133321.801
#34570 E = − 251450.434 w = 121291.694
#41360 E = − 251450.434 w = 118696.640
#102903 E = − 242120.343 w = 117852.552
#118256 E = − 244391.601 w = 112674.066
#28943 E = − 252964.605 w = 107504.102
#29975 E = − 295694.825 w = 98589.112
#46306 E = − 263051.782 w = 96181.777
#41898 E = − 287232.076 w = 87511.372
#12286 E = − 250872.057 w = 87392.420
#57534 E = − 221835.734 w = 84646.080
#96283 E = − 288746.248 w = 77495.901
#119668 E = − 277034.644 w = 73204.163
#115716 E = − 286364.734 w = 73001.260
#51276 E = − 275520.472 w = 72770.490
#87694 E = − 266190.381 w = 71929.908
#15697 E = − 218986.100 w = 68212.015
#112475 E = − 274763.386 w = 65245.150
#94103 E = − 288746.248 w = 64160.864
#11879 E = − 250872.057 w = 63281.193
#37620 E = − 252207.520 w = 47863.705
#3750 E = − 236310.819 w = 47094.174
#121070 E = − 247554.748 w = 40416.827
#116643 E = − 275520.472 w = 39536.169
#110525 E = − 244391.601 w = 35003.881
#18832 E = − 239160.453 w = 33574.662
#14796 E = − 215847.500 w = 22453.692
#23282 E = − 262809.629 w = 14062.253
#19994 E = − 236889.195 w = 9919.014
#49687 E = − 285607.649 w = 8842.844
#16465 E = − 260202.148 w = 8046.981
#11273 E = − 219743.186 w = 4788.171
#23366 E = − 330257.787 w = 2800.769
#56147 E = − 242877.429 w = 2176.947
#32822 E = − 238981.743 w = 1120.885
#113009 E = − 275520.472 w = 143.737
#130485 E = − 332460.591 w = − 82.914
#129564 E = − 350027.463 w = − 1101.015
#51918 E = − 242120.343 w = − 1690.955
#105906 E = − 252964.605 w = − 2075.987
#124329 E = − 295694.825 w = − 2617.626
#130527 E = − 379086.496 w = − 2680.534
#118587 E = − 263808.868 w = − 5762.200
#129766 E = − 327471.596 w = − 5986.122
#120025 E = − 242877.429 w = − 7660.656
#26213 E = − 284850.563 w = − 9026.975
#128909 E = − 294937.739 w = − 13532.075
#121072 E = − 260023.438 w = − 20924.759
#113468 E = − 288746.248 w = − 21533.091
#106073 E = − 265433.296 w = − 22171.162
#56853 E = − 254478.777 w = − 23245.647
#114432 E = − 263808.868 w = − 27454.757
#4069 E = − 273427.924 w = − 28804.281
#27024 E = − 262294.696 w = − 31890.090
#95318 E = − 285607.649 w = − 33443.476
#66326 E = − 252964.605 w = − 35959.486
#114305 E = − 288746.248 w = − 44168.136
#97772 E = − 265433.296 w = − 45416.092
#43655 E = − 287232.076 w = − 53082.451
#36340 E = − 264676.210 w = − 57037.282
#104300 E = − 242877.429 w = − 61398.072
#127449 E = − 269243.273 w = − 69198.042
#8187 E = − 229073.276 w = − 71962.738
#47912 E = − 297319.253 w = − 78591.330
#1820 E = − 204424.861 w = − 81026.371
#79082 E = − 262294.696 w = − 84907.751
#4362 E = − 248490.544 w = − 85781.085
#29865 E = − 265433.296 w = − 89857.894
#115563 E = − 266947.467 w = − 97244.527
#16896 E = − 204246.152 w = − 104155.651
#5130 E = − 250114.971 w = − 110620.681
#87920 E = − 241253.001 w = − 113418.566
#21097 E = − 257820.634 w = − 113749.357
#113892 E = − 263808.868 w = − 121936.871
#66625 E = − 265433.296 w = − 139865.191
#38014 E = − 265433.296 w = − 145557.800
#35466 E = − 219564.476 w = − 155663.262
#30079 E = − 308163.515 w = − 156322.448
#7709 E = − 239160.453 w = − 166613.221
#11048 E = − 218229.014 w = − 169706.627
#45154 E = − 284850.563 w = − 173357.560
#49694 E = − 285607.649 w = − 187075.381
#62080 E = − 243634.515 w = − 207843.981
#41085 E = − 251450.434 w = − 211608.017
#114837 E = − 265433.296 w = − 212148.039
#119906 E = − 245148.686 w = − 214420.006
#82304 E = − 266190.381 w = − 218812.756
#102542 E = − 266190.381 w = − 219355.869
#69677 E = − 264676.210 w = − 240315.697

Figure 5: The molecular graphs of the 100 selected molecules that correspond to the training set with the lowest test MAE after optimization processes. The molecules are placed in a row–major ordering based on the sorted (descending) weights θ of the model computed from training (see Section 2.2). The text under the molecules are: the label (index) of each molecule relative to the preprocessed QM9 dataset; the atomization energy $E := E^{(DA)}$ (in kcal/mol); and the corresponding weight $w \in \theta$ (in kcal/mol). More details of the selected molecules are available in the supplementary material.

3.5 Optimal level of delta learning

In this section, we want to see the extent of delta learning in helping the higher resolution of atomic features in reducing the prediction errors: we hypothesized that atomic features with higher resolution, e.g., the ones that contain higher n–body terms would depend less on the dressed features, and vice versa.

We fix 100 training molecules (the previously generated training set `sid57`) and the rest of 130k molecules for prediction. The atomic features for benchmark are ACSF, SOAP, FCHL19, MBDF, CMBDF, CM, and BOB. We have already seen ACSF up to CMBDF perform in the previous experiments, in fact, those are atomic features that contain some variations of 3–body terms, while CM and BOB contain variations of 2–body terms only. This suggests that CM and BOB may need higher level of dressed feature to enhance the accuracy. As in Table 3, indeed, lower resolution atomic features get better gain of accuracy from higher level of dressed feature. BOB and CM gain accuracy up to dressed angle (DN) level of dressed feature; while the rest of the atomic features that contain 3–body terms either can receive benefits up to dressed bond (DB), or dressed atom (DA) is already sufficient; while dressed torsion (DT) is useful only when used independently (see Figure 2).

Table 3: The minimum prediction MAE of 130k molecules (trained using molecules in Figure 5) for each level of dressed fragment [Dressed Atom (DA), Dressed Bond (DB), Dressed Angle (DN), Dressed Torsion (DT)] and atomic features [ACSF, SOAP, FCHL19, MBDF, CMBDF, CM, BOB]. The bold numbers represent the minimum MAE for each atomic features from all of the dressed features.

	ACSF	SOAP	FCHL19	MBDF	CMBDF	CM	BOB
DA	10.937	10.199	**8.620**	**5.664**	**3.250**	21.625	20.200
DB	**8.050**	**9.687**	9.013	9.142	5.261	13.423	12.721
DN	10.152	10.479	9.866	10.880	8.524	**12.452**	**11.917**
DT	38.099	38.500	38.481	38.391	37.544	38.696	38.762

4 Explainable artificial intelligence (X–AI): kernel feature principal component analysis

In this section we wish to see the distribution of the most optimally selected data in the feature space. In particular, the 2D visualization of the feature space serves as the basis of the analysis.

We fix the current best configuration from all of the previous experiments, in terms of the hyperparameters generated in section 3.3.2. Here we used the best training set obtained from the Tabu search that is listed in Figure 5. This configuration gives 3.25 kcal/mol as we have seen in section 3.5.

First, the Gaussian kernel is computed given the CMBDF atomic features (the hyperparameter configuration used is available in supplementary material), then we perform a PCA on the kernel where it only returns two of the most important feature vectors that corresponds to the two largest eigenvalues: 99.124 and 0.495. The first eigenvalue dominates the second component by a large margin, this implies that only the first principal component is significant, however for visualization purpose we include the second principal component. For the 2D visualization, we project the Gaussian kernel using the eigenvectors that correspond to the mentioned eigenvalues, this outputs a coordinate matrix with 2 columns, where each column corresponds to one of the axis (dimension), and each row corresponds to one data point, then we re-scale each column into $[0, 1]$; this results in the distribution of the data points as shown in Figure 6.

We observe diagonal linear pattern in Figure 6, this may imply a pattern in the chemical space, in particular, in terms of the atomic composition of the molecules. To bring the pattern to light, we sample 3 linear equations f_1, f_2, f_3, where $f_i : \mathbb{R} \to \mathbb{R}$, and then classify each molecule into a cluster $i \in \{1, 2, 3\}$, based on the proximity criteria

$$|f_i(x) - y| \leq \delta y,$$

where x is the first principal component value `PC1` of a molecule, y is the second principal component value `PC2` of a molecule, and $0 < \delta y < 1$ is the proximity threshold, here we use $\delta y = 0.07$. Once the clusters are formed, for each cluster we sort the molecules based on the magnitude of `PC2`.

By observing the clusters, we can generalize a pattern based on the number of atoms within each molecule. In Table 4, it is visible that the frequency distribution of the number of atoms `n_atom` shifts from one cluster to another. In particular, it appears that the distributional peak shifts by 2 values of `n_atom` when moving from a cluster to its next. If we assume the clusters as parallel lines where each cluster is $y_i = ax + b_i$ (similar to f_i) and the only difference between the clusters are the values of b_i, then the clusters with lower values of b_i have distributional peak that are located on lower values of `n_atom`, and vice versa.

A pattern that relates to the atomic composition also emerges, as seen in Figure 7. For example, at the second column at the bottom left, the two lowest molecules are almost isomorphic. Another one is at the rightmost columns, it is visible that the molecules contain more than one Fluorine atoms.

Table 4: The frequency of `n_atom` in clusters $i = 1, 2, 3$.

`n_atom`	$i = 1$	$i = 2$	$i = 3$
9	0	0	3
10	0	0	20
11	0	0	98
12	0	20	377
13	0	103	1243
14	3	505	3125
15	58	1576	6278
16	220	4118	9331
17	734	9178	7018
18	2291	15059	35
19	5801	12173	0
20	11817	8	0
21	11557	0	0
22	0	0	0
23	0	0	0

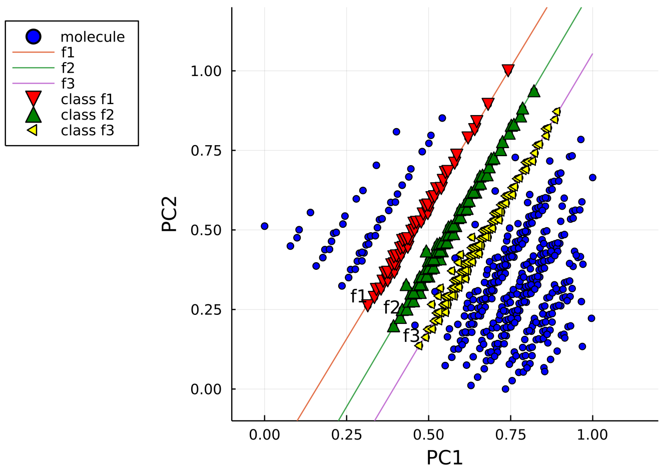

Figure 6: Superpositioned lines f_1, f_2, f_3 and their corresponding clusters on top of the principal components' coordinates of the kernel.

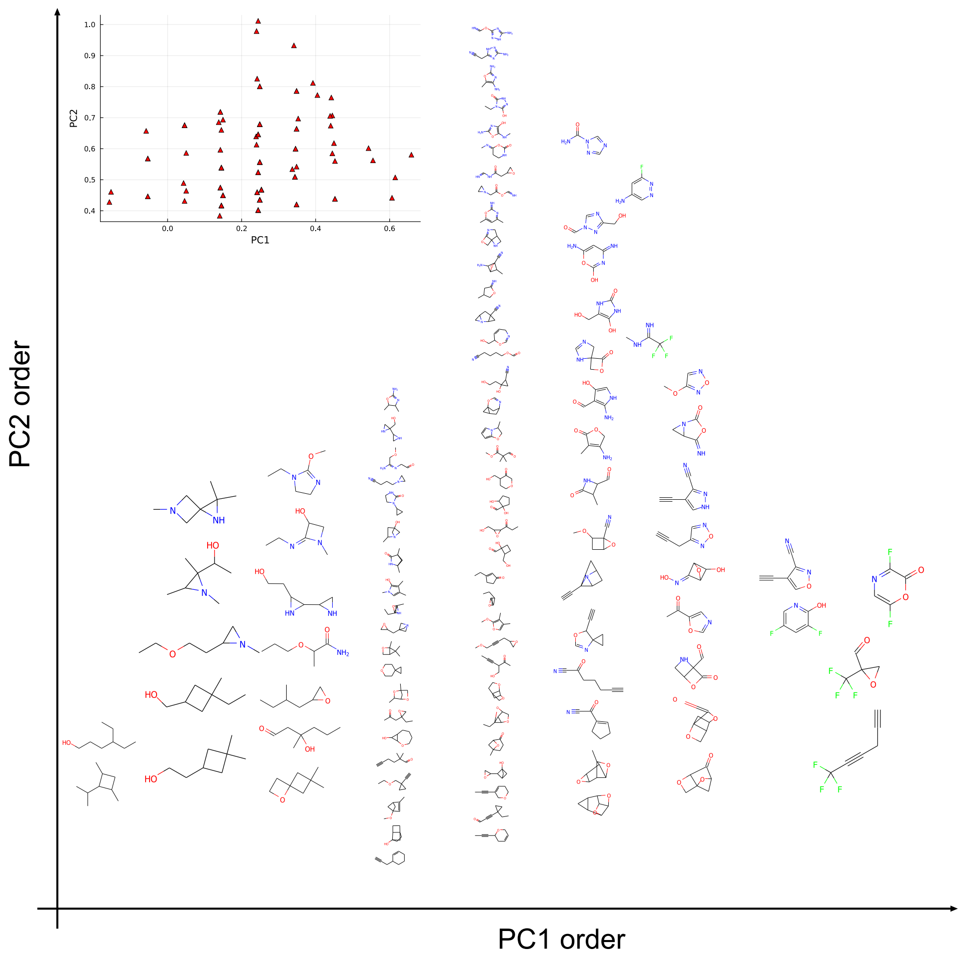

Figure 7: Molecular graphs of the 100 training molecules placed based on the ordering of transformed (`PC1, PC2`) values. A subplot on the top left shows the precise (`PC1, PC2`) coordinates of the training sets after (`PC1, PC2`) coordinates transformation.

Conclusion

We have conducted computational experiments that minimizes the prediction errors of 130k molecules, by mainly using only 100 molecules. The first batch of trials is to make use of unsupervised selection method with `usequence` algorithm. By using `usequence`, it has been seen that the fitting quality is better compared to the standard random sampling with no drawback in computational timing efficiency; as the `usequence` is detached from the objective function calculations.

Up to some level of ΔML, the fitting accuracy increases compared to the standard dressed atom level on some cases. Although, given a sufficiently accurate atomic features, the ΔML does not contribute any additional fitting accuracy, such as the case of CMBDF which is already highly accurate with dressed atom level of energy.

We have also conducted training selection in a supervised manner by the means of combinatorial optimization. Due to the large intractable variable space, we rely on metaheuristic algoirthm to obtain good enough minima. The current best combination of model and training set yields 3.25 kcal/mol accuracy coupled with fast model evaluation of the 130k molecules with only 2.9 seconds on average (excluding the feature extraction).

We analyzed the best found combination of model and training molecules in terms of explainable AI. We observe several patterns, such as the distribution of number of atoms within each cluster of the low dimensional feature space, and the structural similarity between molecules that are close together in the feature space. It is possible that if we can interpret more patterns via the means of explainable AI, then we may know the next direction that we need to take next to reach chemical accuracy with only 100 molecules for training, which would result in solving the QM9 challenge.

Supplementary material

The supplementary materials containing additional details of the theories and computational setup are collected and available in the supplementary material PDF file and github repository https://github.com/berylgithub/supp_mlqm/.

Acknowledgements

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), RGPIN-2023-04853. O.A.v.L. has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 772834). This research was undertaken thanks in part to funding provided to the University of Toronto's Acceleration Consortium from the Canada First Research Excellence Fund, grant number: CFREF-2022-00042. O.A.v.L. has received support as the Ed Clark Chair of Advanced Materials and as a Canada CIFAR AI Chair. B. A. acknowledges the support of Vienna Graduate School on Computational Optimization (VGSCO) under the funding of FWF-Project No. W1260-N35. The computational results presented have been achieved [in part] using the Vienna Scientific Cluster (VSC).

References

[1] O. A. von Lilienfeld, “The QM9 challenge: Who will be the first to reach 1kcal/mol @ N=100?.” Twitter, 2018. Available at https://twitter.com/ProfvLilienfeld/status/1073179005854121984.

[2] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, “Fast and accurate modeling of molecular atomization energies with machine learning,” *Physical Review Letter*, vol. 108, p. 058301, Jan 2012.

[3] B. Cheng, “The QM9 challenge: learning quantum chemistry from small dataset.” University website, 2020. Available at https://www.cst.cam.ac.uk/files/project_suggestions_from_bingqing.pdf.

[4] J. Behler, “Atom-centered symmetry functions for constructing high-dimensional neural network potentials,” *The Journal of Chemical Physics*, vol. 134, p. 074106, 02 2011.

[5] A. P. Bartók, R. Kondor, and G. Csányi, “On representing chemical environments,” *Physical Review B*, vol. 87, p. 184115, May 2013.

[6] F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld, “Alchemical and structural distribution based representation for universal quantum machine learning,” *The Journal of Chemical Physics*, vol. 148, p. 241717, 03 2018.

[7] D. Khan, S. Heinen, and O. A. von Lilienfeld, “Kernel based quantum machine learning at record rate: Many-body distribution functionals as compact representations,” *The Journal of Chemical Physics*, vol. 159, p. 034106, 07 2023.

[8] K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, *Machine Learning Meets Quantum Physics*. Springer, 2020. Part of the book series: Lecture Notes in Physics (LNP, volume 968).

[9] K.-R. Müller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, “An introduction to kernel-based learning algorithms,” *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181–201, 2001.

[10] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, Oct 2001.

[11] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, “Molecular graph convolutions: moving beyond fingerprints,” *Journal of Computer-Aided Molecular Design*, vol. 30, pp. 595–608, Aug 2016.

[12] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML’17, p. 1263–1272, JMLR.org, 2017.

[13] J. Behler and M. Parrinello, “Generalized neural-network representation of high-dimensional potential-energy surfaces,” *Physical Review Letters*, vol. 98, p. 146401, Apr 2007.

[14] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, “Quantum-chemical insights from deep tensor neural networks,” *Nature Communications*, vol. 8, p. 13890, Jan 2017.

[15] O. A. von Lilienfeld, K.-R. Müller, and A. Tkatchenko, "Exploring chemical compound space with quantum-based machine learning," *Nature Reviews Chemistry*, vol. 4, pp. 347–358, Jul 2020.

[16] O. A. von Lilienfeld and K. Burke, "Retrospective on a decade of machine learning for chemical discovery," *Nature Communications*, vol. 11, p. 4895, Sep 2020.

[17] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific Data*, vol. 1, p. 140022, Aug 2014.

[18] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Zeevi, "The farthest point strategy for progressive image sampling," *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1305–1315, 1997.

[19] M. Kimiaei and A. Neumaier, "Heuristic methods for noisy derivative-free bound-constrained mixed-integer optimization," *Optimization Online*, 2023.

[20] B. Huang and O. A. von Lilienfeld, "Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity," *The Journal of Chemical Physics*, vol. 145, p. 161102, 10 2016.

[21] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. I. Goddard, and W. M. Skiff, "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations," *Journal of the American Chemical Society*, vol. 114, no. 25, pp. 10024–10035, 1992.

[22] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Big Data Meets Quantum Chemistry Approximations: The Δ-Machine Learning Approach," *Journal of Chemical Theory and Computation*, vol. 11, no. 5, pp. 2087–2096, 2015. PMID: 26574412.

[23] P. Zaspel, B. Huang, H. Harbrecht, and O. A. von Lilienfeld, "Boosting quantum machine learning models with a multilevel combination technique: Pople diagrams revisited," *Journal of Chemical Theory and Computation*, vol. 15, no. 3, pp. 1546–1559, 2019. PMID: 30516999.

[24] S. Wengert, G. Csányi, K. Reuter, and J. T. Margraf, "Data-efficient machine learning for molecular crystal structure prediction," *Chemical Science*, vol. 12, pp. 4536–4546, 2021.

[25] A. S. Christensen, L. A. Bratholm, F. A. Faber, and O. Anatole von Lilienfeld, "FCHL revisited: Faster and more accurate quantum machine learning," *The Journal of Chemical Physics*, vol. 152, p. 044107, 01 2020.

[26] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers Operations Research*, vol. 13, no. 5, pp. 533–549, 1986. Applications of Integer Programming.

[27] M. Rupp, "Quantum Mechanics / Machine Learning. A community resource." Website. "Accessed: 08-04-2024".

[28] D. Khan, "Convolutional MBDF via fast fourier transforms." Github, 2023. Accessed: 08-04-2024. Available at `https://github.com/dkhan42/cMBDF`.

[29] B. Onat, C. Ortner, and J. R. Kermode, "Sensitivity and dimensionality of atomic environment representations used for machine learning interatomic potentials," *The Journal of Chemical Physics*, vol. 153, p. 144106, 10 2020.

[30] B. Onat, "DescriptorZoo Julia Package to provide single interface for supported descriptors and registry.." Github, 2020. Accessed: 08-04-2024. Available at https://github.com/DescriptorZoo/DescriptorZoo.jl.

[31] L. Himanen, M. O. J. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke, and A. S. Foster, "DScribe: Library of descriptors for machine learning in materials science," *Computer Physics Communications*, vol. 247, p. 106949, 2020.

[32] A. S. Christensen, F. A. Faber, B. Huang, L. A. Bratholm, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "qmlcode/qml: Release v0.3.1," June 2017. Available at https://doi.org/10.5281/zenodo.817332.

[33] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[34] A. Montoison and D. Orban, "Krylov.jl: A julia basket of hand-picked krylov methods," *Journal of Open Source Software*, vol. 8, no. 89, p. 5187, 2023.

[35] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–435, 1952.

[36] Å. Björck, T. Elfving, and Z. Strakos, "Stability of conjugate gradient and lanczos methods for linear least squares problems," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 3, pp. 720–736, 1998.

[37] M. Kimiaei, "MATRSv2.0," February 2024. Available at https://doi.org/10.5281/zenodo.10692874.