UAB
Universitat Autònoma
de Barcelona

Treball Final de Grau
Grau en Física

Study of the physical adequacy of Machine Learning Potentials

Sergi Ortiz Ropero

Supervisor
Dr. Jordi Faraudo Gener
Departament de Física, UAB

Academic Year
2024/25

Call
July 2025

Acronyms

ACE Atomic Cluster Expansion viii, 12, 45, 46, 53, 56

ACSF Atom Centered Symmetry Function viii, 12, 45, 46, 55

CASSCF Complete Active Space Self Consistent Field 30

CC Coupled Cluster 30

CCSD(T) Coupled Cluster with single, double and perturbational triples 30, 55

CI Configuration Interaction 30

CNN Convolutional Neural Network 32, 35, 36

DFT Density Functional Theory 24, 25, 30, 55, 56

DL Deep Learning vii, 5

ES Electronic Structure 3, 4, 40, 42

FF Forcefield 3, 4, 5, 10, 12, 46

GAT Graph Attention Network viii, 36, 37, 38, 39

GCN Graph Convolution Network viii, 36, 37, 38, 39

GGA Generalized Gradient Approximation 30

GNN Graph Neural Network vii, viii, 7, 8, 12, 13, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 48

HF Hartree-Fock 29

LDA Local Density Approximation 30

ML Machine Learning vii, viii, 1, 2, 4, 5, 7, 8, 9, 10, 11, 15, 27, 31, 32, 41, 42, 43

MLIP Machine Learning Interatomic Potential vii, 1, 2, 3, 4, 5, 10, 11, 14, 42

MLP Machine Learning Potential / Multi-layer Perceptron vii, 5, 10, 32, 43, 44, 45

MM Molecular Mechanics viii, 4, 41

MP Møller-Plesset (perturbation theory method) 30

MP2 Møller-Plesset second order perturbation theory 30

MPNN Message Passing Neural Network viii, 8, 12, 32, 37, 38, 39, 48, 53, 56

MSE Mean Squared Error 5, 6, 9

NN Neural Network vii, viii, 5, 6, 7, 9, 10, 11, 12, 40, 44, 45, 47

NNP Neural Network Potential viii, 4, 11, 15, 16, 17, 22, 24, 39, 40, 43, 44, 45, 46, 49, 55

PDE Partial Differential Equation 9

PES Potential Energy Surface 29

PFF Polarizable Force Field 4

PINN Physics Informed Neural Network vii, 8, 9, 42

QM Quantum Mechanics 2, 3, 4, 10, 24, 41, 42, 52

SCF Self-consistent Field 29, 30

SQM Semiempirical Quantum Mechanics 4

TISE Time Independent Schrödinger Equation 3, 28, 29, 42

Contents

Abstract

Important areas such as material design, drug discovery and catalysis rely on modeling the system as set of interacting atoms. Knowledge of these interactions is crucial to correctly depicting the system, but often requires time-consuming calculations. This work focuses on *machine learning interatomic potentials* (MLIPs), a novel approach to atomistic modeling in which, rather than performing time-consuming quantum mechanical calculations to obtain the energy and forces of a given set of atomic positions, a ML model is trained on quantum mechanical results to predict these properties solely based on the atomic arrangement. In order to prove useful as physical models, these should capture the basic physics of any system, as well as the 'rules' of quantum mechanics to successfully describe atomic interactions. However, as these are not fundamentally physical models, it is not clear what basic aspects regarding symmetry, invariance and atomic interactions the models have *learnt* or *understood*. To assess the *physical adequacy* of the models, a series of tests have been conducted to evaluate their translational and rotational invariance, energy and linear/angular momentum conservation, as well as the description of bonds, angles and dihedrals. While the majority of the models have been found to be physically adequate, some of them presented significant flaws regarding rotational invariance and bond descriptions, which will affect their applicability in real-world scenarios.

Keywords: Machine Learning Potentials · Machine Learning · Atomistic Modeling · Machine Learning Force Field · Computational Chemistry · Computational Physics

1 Introduction

Several fields of science such as novel material design, protein structure[1], drug discovery[2,3], and catalysis[4], have seen important development in the past few decades. The study of these atomic systems at a given condition (temperature and pressure) of interest relies on an accurate description of atomic interactions, namely energies and forces, to sample the system's phase space and ultimately, predict new materials, drugs and catalysts using statistical mechanics and *in silico* simulations.

In order to perform atomistic simulations of materials, biomolecules and condensed matter, we need the system's energy and forces. These can be obtained through an *ab initio* (first principles) approach to solving the Schrödinger equation[5] for a given atomic arrangement. Even though well-validated quantum mechanical (QM) techniques exist, it is still a complex task that can become prohibitively expensive with system size. This leads to necessary approximations of the description of the interactions and trading accuracy for computational efficiency[6].

ML has seen widespread adoption and increased applicability in many fields of science, including Physics[7,8], and is an active area of research to this day[9]. In this work, we will study ML interatomic potentials (MLIPs), models trained on quantum mechanical results to predict the atomic interactions solely based on the atomic arrangement, promising to bring near-QM accuracy at much lower computational cost[6] than *ab initio* approaches.

However, while QM methods capture all physics through a set of equations, and thus are intrinsically physical methods, ML models are not designed with any physical considerations in mind. Thus, using ML in physics differs from using ML in other fields in a very subtle way, it not only means training a model, but a *physically adequate* one: at its core, it must capture the underlying 'effective' physics (e.g. the 'rules' of quantum mechanics), but most importantly, it should satisfy and exploit conservation laws, symmetries and invariances of the system to prove useful as a physical alternative to *ab initio* techniques.

Even though MLIPs are benchmarked on large datasets to examine their performance, the study of their physical adequacy is often assumed. Thus, the aim of this project is to validate the *physical adequacy* of three families of MLIPs, `ANI`, `MACE` and `Orbital`, through a series of tests to assess their physical understanding of symmetry, invariances and physical interactions.

This project is structured in two distinct parts. First, a state-of-the-art section provides the necessary concepts regarding atomistic modeling and ML, and introduces the core aspects of MLIPs and the families of models, as in current literature. Then, the series of tests are motivated, and their results are presented and discussed.

2 State of the art

2.1 Atomistic modeling and MLIPs

Atomistic modeling aims to provide the tools to describe the interactions (forces and energies) of an atomic configuration (e.g. molecules, condensed matter, etc.), by first obtaining the potential energy of the system, and from it, the forces. This task essentially involves solving the time-independent Schrödinger equation[5,10] (TISE) of a system of N electrons in the field of M puntual charges (the nuclei), to obtain the energy and electron density (responsible for bonds and interatomic forces) from the system's Hamiltonian $\hat{H}$ (in atomic units),

$$\hat{H} = \underbrace{-\sum_{i}^{N} \hat{\nabla}_i^2(\mathbf{r}_i)}_{T_{\text{electrons}}} \underbrace{-\sum_{\alpha}^{M} \frac{m_e}{m_\alpha} \hat{\nabla}_\alpha^2(\mathbf{R}_\alpha)}_{T_{\text{nuclei}}} \underbrace{-\sum_{i}^{N}\sum_{\alpha}^{M} \frac{Z_\alpha}{r_{i\alpha}}}_{V_{\text{nucleus-electron}}} + \underbrace{\sum_{i}^{N}\sum_{j>i}^{N} \frac{1}{r_{ij}}}_{V_{\text{electron-electron}}} + \underbrace{\sum_{\alpha}^{M}\sum_{\beta>\alpha}^{M} \frac{Z_\alpha Z_\beta}{r_{\alpha\beta}}}_{V_{\text{nucleus-nucleus}}} . \tag{2.1}$$

The set of methods and approximations used to solve this problem are known as *ab initio* Electronic Structure (ES) methods[11] and are discussed in greater detail in Appendix A.

Most importantly, the computational scaling of these methods is $\mathcal{O}(n^4)$, with n the number of electrons (see Hartree-Fock in Fig. 1), or larger for high-level methods[12,13], making the evaluation of atomic interactions with ES methods unfeasible for large systems, such as proteins in solution. In these cases, a trade-off between accuracy in exchange for better computational efficiency is needed, often requiring a linear $\mathcal{O}(N)$ scaling with the number of atoms, N. This alternative description of the interactions are based on the use of *interatomic potentials* (also known as molecular mechanics *force fields* (FFs)), which simplify the system's Hamiltonian to a physics-inspired parametrized functional form[13,14] of the atomic coordinates, $\mathbf{r}^N$, such as

$$\begin{aligned} U(\mathbf{r}^N) \equiv & \overbrace{\sum_{b}^{\text{bonds}} k_b(r-r_b)^2 + \sum_{a}^{\text{angles}} k_a(\theta-\theta_a)^2 + \sum_{n}^{\substack{\text{proper}\\ \text{dihedrals}}} k_n(1+\cos(n\phi-\gamma)) + \sum_{\Phi}^{\substack{\text{improper}\\ \text{dihedrals}}} k_\Phi(r-r_\Phi)^2}^{\text{bonded interactions}} \\ & + \underbrace{\sum_{i}^{N}\sum_{j>i}^{N} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r}\right)^{12} - \left(\frac{\sigma_{ij}}{r}\right)^{6}\right] + \sum_{i}^{N}\sum_{j>i}^{N} \frac{1}{4\pi\varepsilon_0}\frac{q_i q_j}{r_{ij}}}_{\text{non-bonded interactions}} , \end{aligned} \tag{2.2}$$

fitted from empirical or QM results, from which the energy U is computed (with forces as the gradient of U). Note that FFs consist of two clearly distinguished contributions to U.

Bonded terms describe short-range interactions between bonded atoms. Bonds and angles are modeled with a harmonic potential, while proper (usual) 4-atom dihedrals (angle between two planes) are modeled as a Fourier series of cosines. Improper dihedrals represent 4-atom bonded contributions and are used to impose planarity between a central atom and its three bonded neighbors modeled with a harmonic potential of the solid angle.

Non-bonded terms consider pairwise combination of atoms (not directly linked) modeling electrostatics with Coulomb's law (where a point charge is assigned at each atom's position) and dispersion with a Lennard-Jones[15] potential.

Additional multipole and many-body expansion terms can be included to better model long-range and dispersion interactions, charge transfer and bond breaking. In modeling, the description accuracy of U plays a crucial role in the quality of *in silico* simulations, and thus, of all the physical properties that can be extracted from them[16], *inter alia*, thermodynamic, structural and dynamic properties.

Note that, while *ab initio* methods work without assuming fixed functional dependencies (only requiring the geometry and spin of the system), conventional MM-FFs require a preconceived notion of the bonding patterns and system-dependent non-trivial parametrization of all bonded and non-bonded contributions of Eq. (2.2). Thus, these lack an 'out-of-the-box' applicability between systems, property known as *transferability*, where the same functional form is used for different systems.

Ideally, we would be interested in obtaining a transferable potential energy functional form able to describe the interactions at QM accuracy with FF computational cost. It is precisely in this task where ML can prove useful. MLIPs are ML transferable FFs trained on QM reference data that *learn* the mapping between an atomic arrangement and the potential energy, providing near-QM quality at near-FF cost, all without imposing a fixed functional form nor relying on a preconceived notion of fixed chemical bonds or knowledge about the relevant interactions.

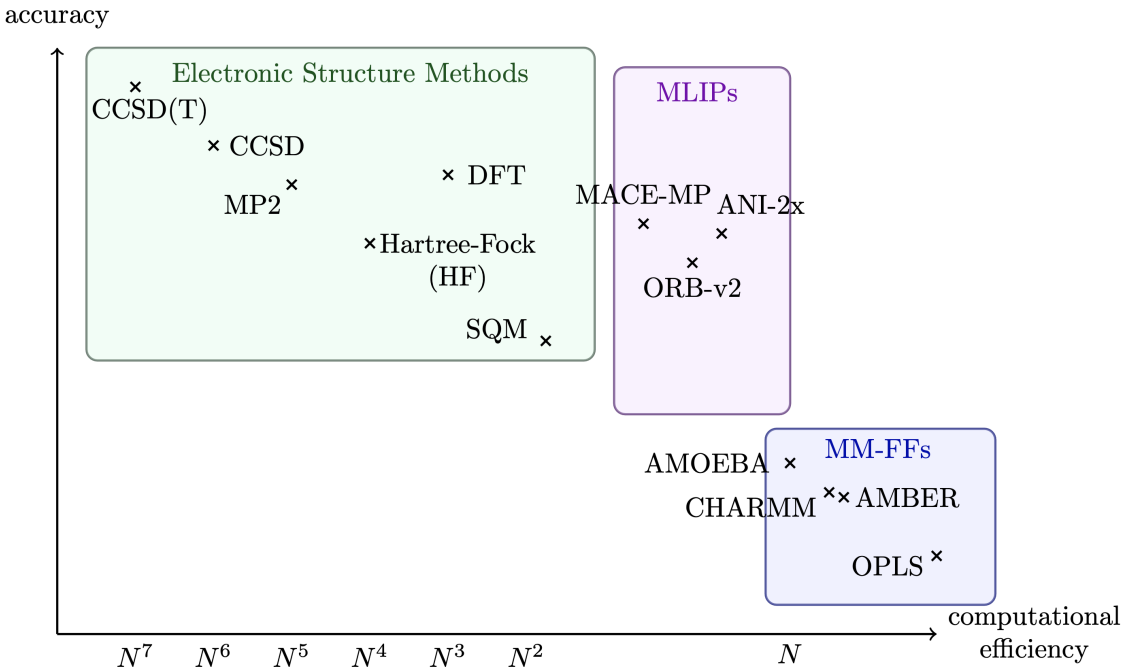


Figure 1: Accuracy of $U(\mathbf{r}^N)$ and approximate scalability of each method with respect to the number of atoms N. Most common ES methods (AM1[17], MNDO[18], PM6[19], DBT-B[20]), FFs (AMOEBA[21], AMBER[22], CHARMM[23], OPLS[24]), and three MLIP models[25–27] are represented. Note that all the NNPs are substantially more accurate compared to MM-FFs while having similar computational cost. NNPs also surpass semiempirical QM (SQM) methods while at a fraction of the cost. PFFs such as AMOEBA[21,28] offer better long-range descriptions, but come at a higher cost.

Fig. 1 shows how transferable MLIPs bridge the gap between ES methods and conventional approximate MM-FFs, with near linear scaling with system size, while being much more accurate than traditional FFs, due to them being trained on high-level QM data. However,

as MLIPs are not constrained on the functional form they learn (as opposed to FFs, which use a physically motivated one), it might not capture the relevant 'basic physics' needed for an adequate, physically correct, prediction of energies, such as symmetries and invariances.

2.2 ML and DL algorithms

A ML algorithm is one able to *learn* from data. The core concepts of a ML algorithm are the *task* to be performed, the *features* used and extracted from data, and the *model*, which is *trained* on the features to accomplish the task with a measured performance. In this sense, the mapping between input and output of the model is not hard-coded as in a traditional algorithm, but rather, it is *learnt from data*. The core concepts are defined as follows[29–31]

- **The task.** ML algorithms allow us to tackle tasks that are too difficult to solve with fixed hand-designed programs (e.g. image classification). The task is what the model is expected to be able to perform after learning, rather than the learning itself. The models studied in this project will focus on regression.
- **The data.** To train a ML model, data is needed. The dataset is a collection of *examples* (or *data points*), consisting of a set of *features* that have been quantitatively measured from some object or event that we want the ML model to process.
- **The performance.** To assess the abilities of a ML algorithm, a task-specific quantitative measure of the performance must be designed, called a *loss or objective function* $\mathcal{L}$. For a classification task, that could be the model accuracy, or the Mean Squared Error (MSE) for regression.
- **The learning.** ML algorithms can be broadly categorized as *unsupervised* or *supervised* (although other categories exist). Supervised learning algorithms learn from example features from the dataset, but, additionally, each is also associated with a *label* (or target) provided by the 'teacher', which the model has to correctly predict. In a regression task to predict molecular energies, an example consists of features (atomic positions, atomic number and other descriptors) and a label (the energy).
- **The model**. Given the training data, a goal, and a training procedure; an architecture is chosen and trained to fulfill the task, producing a model. The choice of architecture is highly dependent on the *task*, and contains a series of trainable parameters, θ, often classified as weights w, and biases, b, which are optimized during training by maximizing the performance (minimizing the loss function).
- **Generalization**. The ultimate goal of the model is to correctly generalize to previously unseen data: if a model has been trained on predicting the energies for an atomic arrangement, whether it also correctly predicts the energies for new (not in the training set) ones. For a regression task, the *generalization error* is defined to be the error between the predicted and true value of previously unseen examples.

DL is a particular kind of machine learning in which the model learns a complex representation of the data based on a hierarchy of concepts, each defined as a function of simpler ones, which allows it to build an abstract representation computed in terms of less abstract ones[32].

2.2.1 Neural Networks (NNs) and the Multilayer Perceptron (MLP)

The quinetessential example of a ML/DL model is the multilayer perceptron[33–35] (modern version of the artificial NN). A NN is a simple non-linear model formed by a set of *nodes*

(or neurons), each holding a value. The NN of Fig. 2a consists of two visible input and output layers, with two hidden layers of $\{\ell_1, \ell_2\}$ nodes. Node i values, h_i are computed in a feed-forward way from the previous layer values. For nodes in the first hidden layer,

$$h_i^{(1)} = \sigma\left(b_i^{(1)} + \sum_j^{\text{features}} w_{ij}^{(1)} x_j\right) , \qquad \text{or equivalently,} \qquad \mathbf{h}^{(1)} = \sigma\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) , \quad (2.3)$$

as seen in Fig. 2b. The NN *depth* is the number of hidden layers it consists of. Deeper networks allow each layer to extract more abstract information out of the data from previous ones, allowing it to learn highly complex feature spaces efficiently.

To assess the model's performance, a loss function is defined (e.g. MSE), and is minimized during training by optimizing the trainable parameters θ,

$$\mathcal{L}_{\text{MSE}}(\theta) = \min_\theta \sum_i^{\text{examples}} \left[\mathbf{y}_{\text{pred}}(\mathbf{x}_i, \theta) - \mathbf{y}_i\right]^2 , \qquad \nabla_\theta \mathcal{L} \to 0 . \quad (2.4)$$

Additionally, every model contains a set of **hyperparameters**, which are not modified in training, e.g. the number of nodes in a hidden layer or the number of hidden layers. These can affect the generalization error and must be optimized separately from training.

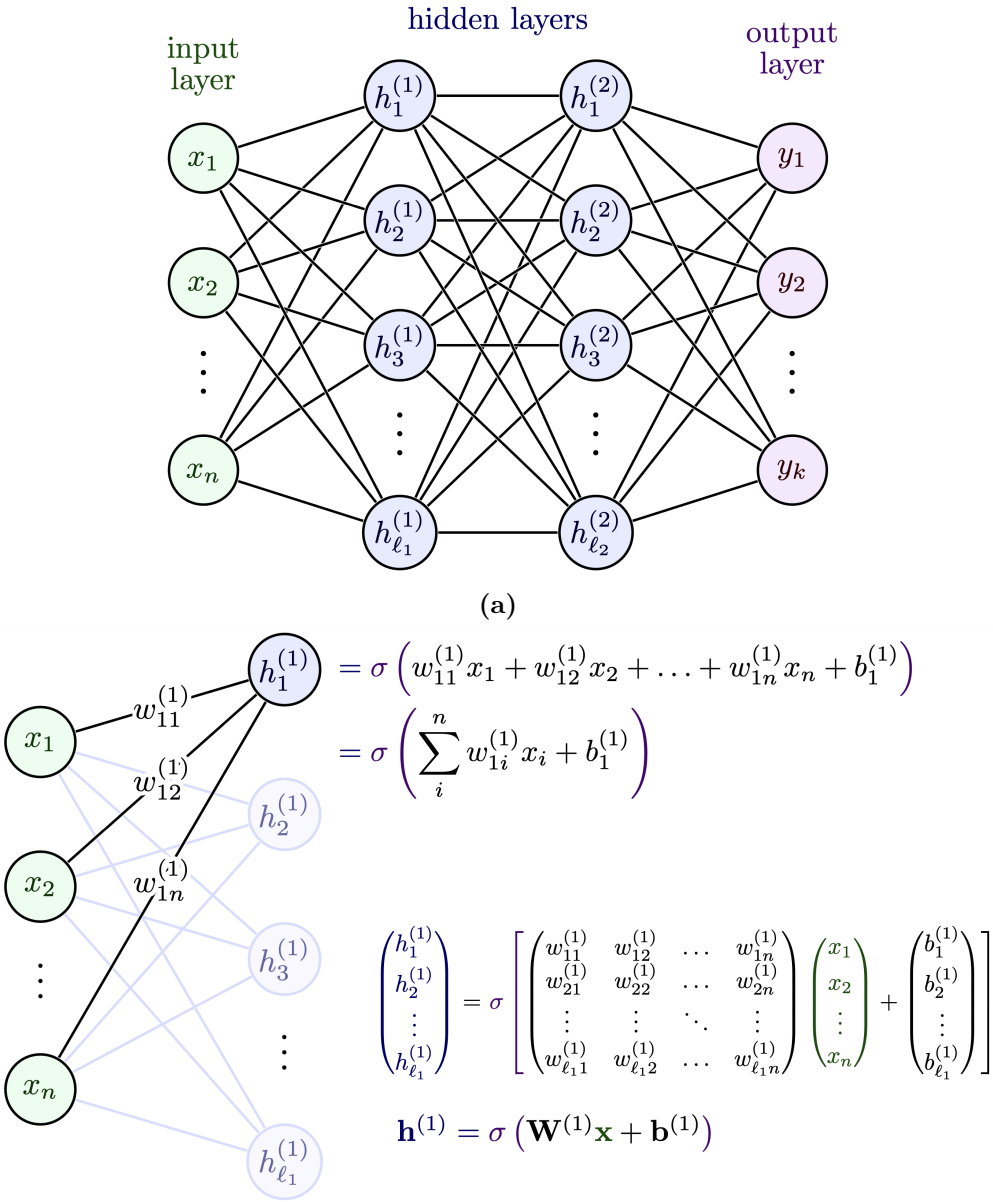


Figure 2: **(a)** Feed-forward NN with two hidden layers of $\{\ell_1, \ell_2\}$ nodes each. The input layer takes an example, consisting of n $\{x_i\}$ features. The result is passed to the output layer, consisting of k y_i labels. In a feed-forward way, the input is passed to the next layer until it reaches the output layer. The trainable parameters θ are trained until the loss function $\mathcal{L}$ maximizes the performance of the network. **(b)** Information passing mechanism in an artificial NN. Each node in the next layer takes the inputs of all the nodes of the previous layer and performs a non-linear transformation σ. In the image, $\mathbf{x}$ is the input feature vector, $\mathbf{W}^{(1)}$ and $\mathbf{b}^{(1)}$ the weight matrix and bias vector of the first hidden layer, respectively (trainable parameters of the model), and σ is a non-linear activation function, such as ReLU(x).

2.2.2 Graph Neural Networks (GNNs)

Often, the architecture used in a ML model is linked to the data format or structure of a specific problem, with the objective of obtaining data-efficient models, ones that can take advantage and exploit the structure and symmetries in the data. GNNs take advantage of data that can be structured as *graphs*. A first intuitive example is molecules, which can be naturally modeled as graphs (Fig. 3), where we can think of each atom as a node and each bond as an edge connecting the nodes.

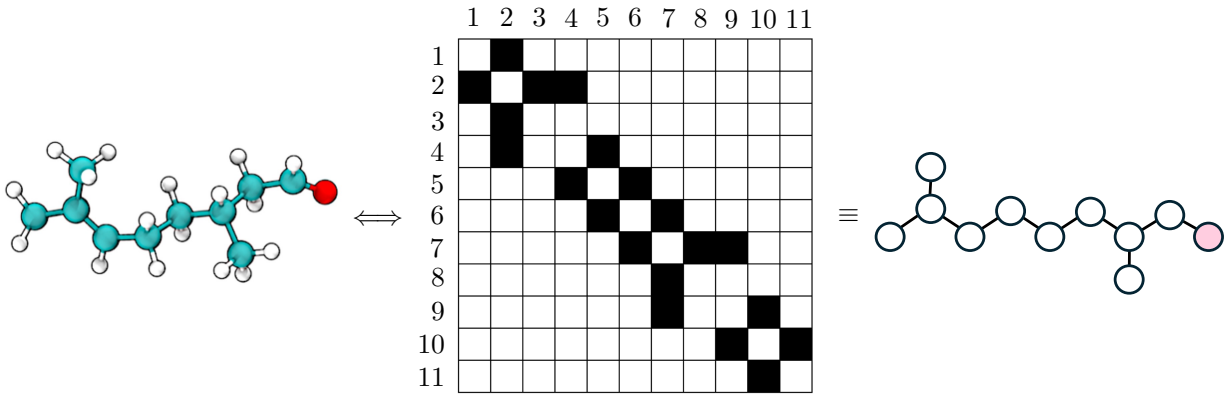


Figure 3: An *a priori* perspective on how molecules can be naturally represented as graphs. Each node represents a non-H atom and contains a node feature vector $\mathbf{x}_i$ with atomic attributes (e.g. atomic number, atomic mass, partial charge). Edges are identified with bonds, containing edge feature vectors $\mathbf{e}_{ij}$ with attributes about bond type (single, double, triple bond, etc.). The connectivity of the graph is captured from the atomic connectivity, and stored in the adjacency matrix $\mathbf{A}$. Adapted from Ref. [36].

We encourage the reader to refer to Appendix C to understand Graph Neural Networks (GNNs), and more specifically Message Passing Neural Networks (MPNNs), a state-of-the-art architecture, that is being implemented in a wide variety of fields in science with great success[1,37].

2.2.3 General ML process and data representation.

The first step in any ML application, irrespective of the problem, task, dataset, architecture, and training procedure, is to define the problem and task. Rather than hard-coding the algorithm, the human intervention is in the *design of the model*. The nature of the problem will determine the data and features that can be extracted and ultimately influence the design process. In it, we must choose how the dataset is preprocessed, what data is fed to the model and how, the architecture, the loss function and devise the training procedure to ultimately minimize generalization error. Once the trainable parameters have been optimized, the model can be benchmarked on new unseen data to evaluate the final performance on the task.

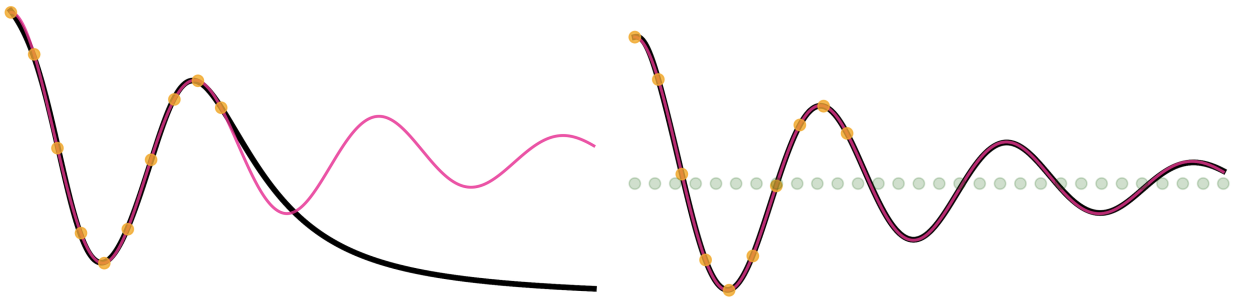
The collection and curation of the dataset used for training is arguably as important as model design itself. It must contain both the right information and representation to facilitate the training and improve the generalization of the model. Particularly, it is not only the collection of data that is important, but rather *what features are selected and how they are represented such that the model trains as effectively as possible*. Practical considerations regarding model complexity are given in Appendix B.

2.2.4 Physical priors in ML. PINNs

Even though ML models have been successfully applied to a variety of scientific applications, most of these approaches are 'data-driven': a model is trained solely based on data. While this is perfectly fine from the data science perspective, physical applications may require the model to understand specific physical aspects to produce meaningful results. However, as no physical priors are used in this approach, basic constraints such as energy conservation and other symmetry-derived properties (e.g. linear/angular momentum conservation) must

be *learnt through data*, which often results in insufficient physical understanding (physics are not *enforced*). To circumnavigate this issue, these symmetry constraints can be introduced as *inductive biases* in the architecture design[38–45].

These physical priors can be interpreted as additional system data, used in the *design* of the model rather than in *training*, to exploit the physical knowledge about the system and produce symmetry-aware and physics-informed models that are *physically adequate*.



(a) Data-driven naïve approach. **(b)** Physics-informed approach

Figure 4: Trajectory prediction for an underdamped oscillator for a **(a)** NN and **(b)** PINN models. Training points are represented in orange and the real solution in magenta. The inclusion of physical priors with the PINN model greatly improves its prediction compared to the purely data-driven approach, which predicts an un-physical trajectory. Figures extracted from Ref. [46].

Fig. 4a exemplifies the idea of including available physical information as priors to improve scientific-ML models. In this example, a simple data-driven NN model fails to predict the system's trajectory, providing a physically inadequate prediction. However, as we know this object behaves as an underdamped oscillator, its corresponding PDE can be used as an inductive training bias. This is the recipe for PINNs[47–49], an architecture that uses physical priors (knowledge of the system) to improve the model. This is accomplished in training via altering the loss function $\mathcal{L}$ to include both a 'data loss' (MSE) and 'physics loss' (deviation from the reference PDE) as seen in

$$\begin{aligned}\mathcal{L}(\theta) &= \mathcal{L}_{\text{data}}(\theta) + \lambda \mathcal{L}_{\text{physics}}(\theta) \\ &\equiv \underbrace{\frac{1}{N}\sum_{i}^{N}\left[y(x_i) - u_{\text{pred}}(x_i;\theta)\right]^2}_{\text{data loss, MSE}} + \lambda \underbrace{\frac{1}{M}\sum_{j}^{M}\left(\left[m\frac{d^2}{dx^2} + \mu\frac{d}{dx} + k\right]u_{\text{pred}}(x_j;\theta)\right)^2}_{\text{physics loss (includes physics knowledge)}}, \end{aligned} \tag{2.5}$$

with λ a regulation parameter (see Ref. [46] for further information). This 'suggestion' on what the solution should physically resemble translates into more physically adequate, generalizable and interpretable models (Fig. 4b).

2.3 Machine Learning Potentials

Once the general ML concepts have been covered, we move on to briefly motivate the design choices of MLIPs, their physical understanding, and present `ANI`, `MACE` and `Orbital` families of MLIPs. As briefly discussed in Sec. 2.1, MLIPs aim to predict the energy of an atomic arrangement, defined by coordinates and identities (atomic numbers), trained on highly accurate QM data[6,38,50,51]. The incentive of this approach is to 'skip' all the expensive QM calculations and provide comparable accuracy at much less computational cost.

MLIPs deal with a fundamentally physical task, and as such, it is crucial that models understand and exploit the physical constraints and symmetries of the system through *inductive biases*. Correctly capturing the symmetry is especially important with molecular systems. Scalar properties such as energies must remain *invariant* with respect to translations, rotations and atom-wise permutations, while other vectorial properties, such as the dipole moment, must be *equivariant* to these (transforming according to e.g. a rotation). Although for energy prediction invariant models suffice, equivariant architectures correctly capture all Euclidean symmetries and perform substantially better[52–54] (see Appendix. E on equivariant models).

Additionally, conservation laws such as energy conservation and linear/angular momentum conservation as well as atomic permutational invariance provide strong constraints that can be used as guiding principles in search of physically adequate models[6] (see Sec. D.1 for additional information).

Inadequate description of symmetry can lead to severe consequences, such as breaking fundamental conservation laws, thus resulting in fundamentally flawed non-physical models.

2.3.1 Model respresentation

There are several practical considerations that must be accounted for when designing a MLIP. Consider a model consisting of a MLP such as that of Fig. 2a. This basic model accepts a *fixed length*, *ordered* input, the coordinates of the atoms $\mathbf{r}$ and outputs the potential energy prediction, E. We define the *representation* of the model as the description of the atomic arrangement that is parsed as input to the model[6,38,39,51], efectively 'translating' the atomic arrangement into a suitable input to the NN. In order for this model to be *physically adequate* (satisfy the basic physical constraints and system symmetries), the predicted energy must be rotation-, translation- and permutation-invariant[38,39]. To ensure this, we must design a representation that is translation, rotation and permutation-invariant as well as *complete*[38] (describes a molecule's conformation in a unique way). This would guarantee that, given any symmetry-modified coordinates, the same representation is provided as input to the model and thus the same output (energy) is returned, satisfying invariance.

As an example, a cartesian representation of the system, whilst simple and complete, is not suitable[39,51,55]: the list of coordinates may be arbitrarily ordered, and they are not rotation- and translation-invariant, providing a different input to the model and thus the energy would not be invariant[6]. Note, that conventional FFs (Eq. 2.2) use a sum of up to 4-body internal coordinates, making them rotation-, translation- and permutation-invariant and complete.

Most importantly, however, is the consequence of using a *fixed length* input (Fig. 2a). As it is inherent to the model (cannot be changed), its applicability is limited to a fixed number of atoms[6,38,50]. To make the model independent of system size, the system energy, E, is decomposed into local (site) contributions[38]

$$E(\mathbf{r}_1, \ldots, \mathbf{r}_N) = \sum_{i=1}^{N} E_i(\mathbf{r}_1, \ldots, \mathbf{r}_N)\,, \tag{2.6}$$

which makes E permutation-invariant. The problem transforms into a different one: rather than predicting the energy of the whole molecule (considering all possible interactions), we predict the *energy of an atom in its chemical environment*, E_i.

Note that closer atoms in the environment will have larger contributions to the local energy. This naturally leads to a *locality approximation* defined by a cutoff function, $f_c(r)$, determining 'how many interactions should be accounted for' at the expense of computational effort[6], with interactions between atoms separated by more than the cutoff radius neglected (Fig. D.2). However, this locality approximation fails to describe systems where long-range electrostatic interactions are important[55] (see Sec. C.5.1 for further detail).

Fig. 5 represents the general workflow of a MLIP. An atomic structure is 'translated' into a physically adequate representation, that serves as input to the ML model to predict the energy E.

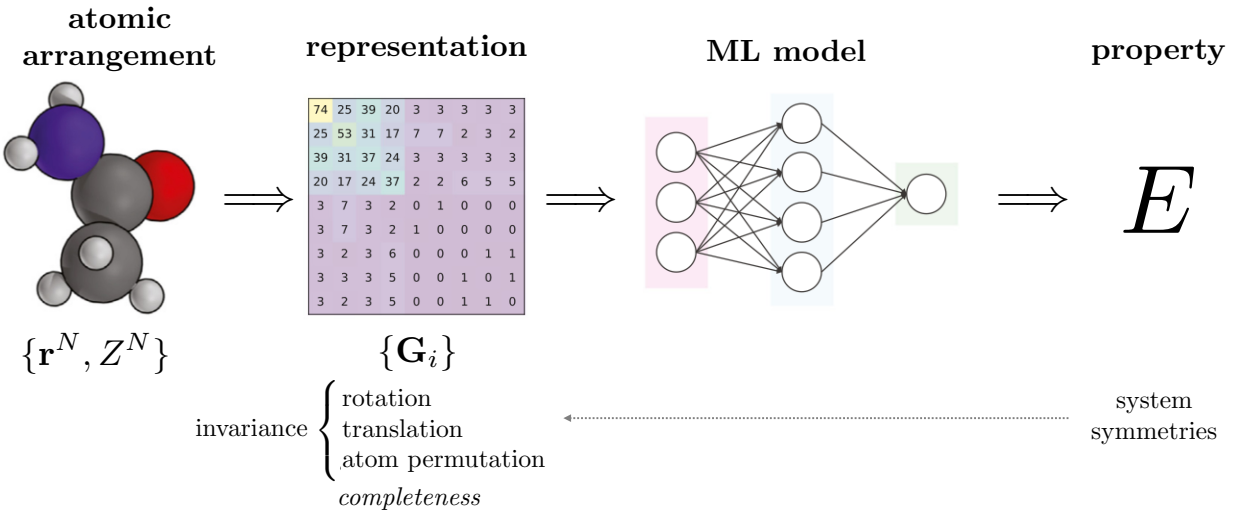


Figure 5: Overall NNP workflow. Given an atomic arrangement characterized by the atomic positions $\mathbf{r}^N = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ and identities $Z^N = (Z_1, \ldots, Z_N)$, a representation is built from a set of descriptors $\{\mathbf{G}_i\}$ for each atom. These encode the chemical environment and serve as input to the ML model, which predicts the desired property, in the case of NNPs, the molecular energy. Physical constraints and system symmetries, however, impose a certain *structure* on the atomic representation, mainly invariances, as well as completeness: the representation must be univocal for each atomic arrangement, such that no two different structures have the same representation. Adapted from Ref. [56]

2.3.2 Descriptor-based NN models. `ANI` models

Once the locality approximation has been introduced, is most important step is to design a suitable *representation* able to *describe* the chemical environment of each atom, and from it, derive the local contributions E_i to the total energy.

A first approach is to convert atomic coordinates $\mathbf{r}^N$ into an appropriate representation through the use of *descriptors*. These are many-body invariant symmetry functions G_k that describe the chemical environment uniquely[6,38,39,51,55], from which a model is able to predict the local energy contributions to the total energy. A general depiction of a descriptor-based model is shown in Fig. D.3. The environment of each atom i is encoded in a fixed set of M symmetry functions $\mathbf{G}_i = (G_{i1}, \ldots, G_{iM})$, which serve as input to the NN.

A relevant descriptor-based NN family of invariant and transferable models is `ANI`, consisting of `ANI-1x`[57], `ANI-1ccx`[58] and `ANI-2x`[26]. These use (a modified version of) atomic centered symmetry functions[38] (ACSFs), a set of M transferable symmetry functions G_k of two types: *radial* and *angular* symmetry functions, used to describe the radial and angular distribution of neighbors up to the cutoff radius, respectively. These functions depend on two-body and three-body internal coordinates, i.e. distances and angles, respectively, which ensures translation- and rotation-invariance. The aggregation of all internal features in the environment guarantees permutation invariance. Additional information of these descriptors is provided at Sec. D.3 and Refs. [38, 39, 51, 55]

`ANI` pretrained models are available in `Python` through `TorchANI`[59] (`GitHub`, `Documentation`) and are trained on small organic molecules supporting H, C, N, O (and F, S, Cl for `ANI-2x`)[26,58,60–62]. To ensure energy conservation, forces are predicted as the analytical gradient of the potential energy with respect to nuclear displacements. Additional information about pretrained models is available in Appendix F.

2.3.3 GNN models. `MACE` and `Orbital` models

ACSFs however, are just one type of descriptors, and others have been developed[39–41,63]. Among these, atomic cluster expansion (ACE) has proven to be a broad generalization of all before-mentioned sets of descriptors[40,64]. The idea behind ACE (see Ref. [40]) is to estimate the atomic energy E_i as a many-body expansion

$$E_i = V^{(1)}(\mathbf{r}_i) + \frac{1}{2}\sum_j V^{(2)}(\mathbf{r}_i, \mathbf{r}_j) + \frac{1}{3!}\sum_{j,k} V^{(3)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \frac{1}{4!}\sum_{j,k,l} V^{(4)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l) + \ldots \quad (2.7)$$

This is, at its core, what is used to express classical FFs in Eq. (2.2). ACE is a generalization of ACSFs that generates a symmetry-aware ν-body expansion at low computational cost. When ACE is expanded up to three-body contributions, it can be shown that it is equivalent to ACSFs[64].

The importance of many-body descriptors is fundamental and has proven to be a limiting factor in certain sets of descriptors, as there might exist different chemical environments which can only be distinguished by using higher-order descriptors[65,66], as depicted in Fig. D.4, hindering *completeness*.

A particularly interesting descriptor-based GNN (MPNN) family of *equivariant* transferable models is `MACE`[25,45,67,68], consisting of `MACE-MP`[25] and `MACE-OFF`[67] (`GitHub`, `Documentation`). These structure the molecule as a GNN would (Fig. 3), where nodes contain its atomic coordinates $\mathbf{r}_i$, fixed features $\boldsymbol{\theta}_i$ and the atomic environment description through generalized-ACE, $\mathbf{h}_i$. See Secs. D.3.5 and E.4 for further discussion.

`MACE-OFF` is trained on small organic molecules[69], supporting {H, C, N, O, F, P, S, Cl, Br, I} (thus comparable to `ANI-2x`), while `MACE-MP` is trained on materials[70], covering almost all the periodic table. Forces are computed as in `ANI`.

Another recent family of transferable end-to-end GNNs is `Orbital`, comprised of `ORB-v2` and `ORB-D3-v2` models[27] (`GitHub`). Rather than using a descriptor-based approach using a set of handcrafted symmetry functions as chemical environment descriptors (as does `MACE`), these models use an *end-to-end* approach: the model *learns* a suitable invariant representation through data, which is then used as input to predict site energies (see Fig. D.5). As such, `Orbital` models are not invariant/equivariant (as are `ANI` and `MACE`, respectively); instead, they opt to learn rotational invariance through augmenting data during training rather than strictly enforcing it as an inductive bias in the model. This increases predictive speed at the expense of physical adequacy.

`Orbital` models are trained similarly to `MACE-MP`, with a focus on materials and almost all periodic table coverage. `ORB-D3-v2` is trained on a dispersion-corrected[71] training set of `ORB-v2`. Additionally, forces are predicted in a non-conservative way: rather than using the gradient of the potential energy (`ANI`, `MACE`), forces are predicted by the model itself, resulting in a near but not strict conservation of energy.

3 Objectives and Methodology

Up until this point, we have presented the core aspects of this field, although with no new contributions other than providing hindsight and literature about the topic. This section aims to briefly list this project's new contributions and their computational means.

MLIP models, when presented, are often treated as data science models, and thus are benchmarked on large datasets to show their performance. This assessment, however, does not show their physical understanding nor adequacy, which is often assumed.

The general objective of this work is to study the *physical adequacy* of the before-mentioned `ANI`, `MACE` and `Orbital` pretrained models, checking whether these satisfy important physical aspects such as symmetries and invariances, and contain a qualitatively correct understanding of physical interactions between atoms.

To accomplish this main task, three subtasks are considered, taking on the study of *symmetries*, *invariances* and *interactions*:

- **Symmetries and invariances.** Assess the symmetry-understanding and behavior of the models under translations and rotations. A set of molecules are translated and rotated in space to check *energy invariance* as well as *conservation of linear and angular momentum* through the conservation of energy, total force and torque.
- **Interactions.** Evaluate the physical description of interactions in atomistic systems (small molecules). A set of molecules are deformed to study how bonds, angles and dihedrals are described *via* examining the potential energy shape along the deformation.
- **Complex systems.** Showcase the importance of symmetry-aware models in the description of atomic interactions of two highly symmetric systems: COSAN and cyclobutadiene.

Regarding computational methodology, a set of `Python` scripts have been developed which use the `ASE`[72] implementation of the pretrained models to perform all the before-mentioned objectives. These scripts are publicly available at this project's `GitHub` repository.

4 Results

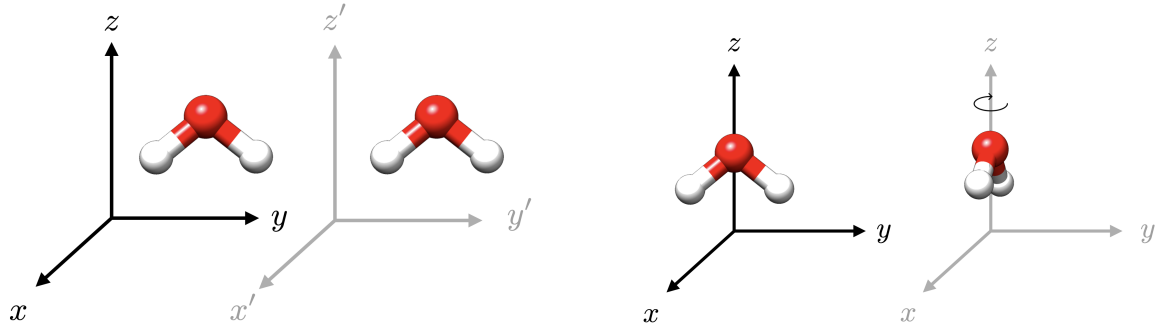
The most important aspect of a ML model when applied to science is its physical adequacy and transferability, determined by how well it is able to 'capture' the fundamental constraints and symmetries of any given system. In this section, we aim to evaluate what 'basic physics' several models understand by performing simple yet relevant tests to assess three main topics: *symmetry understanding*, *molecular deformations*, and *complex molecular systems*.

4.1 Physical adequacy. Invariance and Equivariance

As discussed in previous sections, NNPs must be physically correct in order to be used as a substitute to proper physical models. The following tests aim to check that physical constraints and system symmetries are respected, ensuring that the models do not break basic physics.

4.1.1 Rotation and translation invariance

It is important that energy is invariant to translations and rotations, as otherwise, space isotropy would be broken, thus resulting in a bias for certain configurations during simulations (affecting all the information extracted from them). To test whether symmetry invariances are respected, rotations and translations of a set of small molecules have been performed (as seen in Fig. 6) and the energy along the transformation recorded. If the models are invariant, these energy fluctuations will be negligible, as the initial and transformed molecule are identical.



(a) water translation (y-direction) **(b)** water rotation (z-axis)

Figure 6: Graphical view of the transformations applied to test symmetry and physical constraints.

Tables 1 and 2 display the largest energy fluctuation (model error) along a translation and rotation, respectively, for two molecules (more in Appendix G). When it comes to translations, all the models are proven to be invariant, as these maximum fluctuations are attributed to numerical errors. Translationally invariant models are trivial to implement (Sec. E.1). When it comes to rotations, `ANI` and `MACE` show rotational invariance, with fluctuations well in the range of numerical error. However, both `ORB` models are not rotationally invariant, as seen by the large errors (highlighted in bold in both tables) of $\gtrsim 10$ meV, sometimes exceeding 'chemical accuracy' (1 kcal mol^{-1}) for systems such as acetone and ethanol (Fig. G.8b).

Table 1: Maximum error committed (eV) when predicting energies for xy-axis translations of three simple molecules. Errors beyond numerical fluctuations are highlighted in bold. Figs. G.7 and G.8 display the graphical representations for all studied systems.

System	ANI-1x	ANI-1ccx	ANI-2x	MACE-MP	MACE-OFF	ORB-v2	ORB-D3-v2
water	5×10^{-7}	2×10^{-6}	2×10^{-6}	2×10^{-6}	1×10^{-8}	8×10^{-6}	1.2×10^{-5}
acetone	2×10^{-6}	4×10^{-6}	7×10^{-6}	1×10^{-7}	1×10^{-7}	2×10^{-5}	2×10^{-5}

Table 2: Maximum error committed (eV) when predicting energies for z-axis rotations of three simple molecules. Errors beyond numerical fluctuations are highlighted in bold. Figs. G.7 and G.8 show the graphical representations for all studied systems.

System	ANI-1x	ANI-1ccx	ANI-2x	MACE-MP	MACE-OFF	ORB-v2	ORB-D3-v2
water	3×10^{-7}	2×10^{-7}	7×10^{-7}	1×10^{-6}	1×10^{-8}	**0.009**	**0.010**
acetone	2×10^{-5}	1×10^{-5}	1×10^{-5}	4×10^{-6}	1×10^{-7}	**0.025**	**0.070**

These results are in line with how the models were designed: while `ANI` and `MACE` are designed to be rotationally invariant and equivariant, respectively, while `Orbital` models opt to *learn* rotation invariance through data augmentation rather than having it 'built-in', leading to an approximately invariant representation of the chemical environment. Even though this last approach makes the model faster when predicting energies, the lack of rotational invariance in `Orbital` models can have a biasing effect in *in silico* simulations, favoring a specific set of atomic configurations in space. These effects should be studied further to assess whether rotationally augmented models can be used over invariant or equivariant ones.

4.1.2 Conservation of linear and angular momentum

Besides energy invariance, NNPs must conserve total linear and angular momentum, which translates to zero net force and torque in the absence of external interactions (see Sec. D.1). These constraints are particularly important in simulations where the dynamics of the system are relevant: if a molecule is displaced and total force becomes different from zero along that transformation, the system might start randomly accelerating to a particular direction. Similarly, if total torque is not zero along that transformation, the molecule might start rotating, messing up the dynamics, and, consequently, the information retrieved from them. These checks are, thus, of particular importance to ensure meaningful predictions from NNPs.

Table 3: Maximum force deviation (eV/Å) when predicting the total force for xy-axis translations of three simple molecules. Deviations beyond numerical fluctuations are highlighted in bold. Results for additional systems are showcased in Figs. G.9 and G.10. Analogous results for z-axis rotations are displayed in Table 7.

System	ANI-1x	ANI-1ccx	ANI-2x	MACE-MP	MACE-OFF	ORB-v2	ORB-D3-v2
water	1×10^{-8}	6×10^{-8}	2×10^{-8}	3×10^{-8}	6×10^{-8}	9×10^{-7}	7×10^{-7}
acetone	7×10^{-7}	6×10^{-7}	7×10^{-7}	4×10^{-7}	4×10^{-7}	7×10^{-7}	7×10^{-7}

Table 4: Maximum torque deviation (eV) when performing z-axis rotations for three molecular systems. Deviations beyond numerical fluctuations are highlighted in bold. Results for additional systems are showcased in Fig. G.11.

System	ANI-1x	ANI-1ccx	ANI-2x	MACE-MP	MACE-OFF	ORB-v2	ORB-D3-v2
water	1×10^{-7}	5×10^{-7}	7×10^{-7}	1×10^{-7}	3×10^{-6}	7×10^{-5}	7×10^{-5}
acetone	5×10^{-6}	1×10^{-5}	2×10^{-5}	2×10^{-5}	3×10^{-5}	5×10^{-5}	7×10^{-5}

If the models preserve linear and angular momentum, no net force or torque should appear when each molecule is transformed. Similarly to energies, Tables 3 and 4 show the maximum net force and torque (error) along a translation for two molecules. All models preserve linear and angular momentum, with fluctuations in the range of numerical errors. Note, however, that not all models predict forces in the same way. While ANI and MACE obtain forces as the analytical gradient of the potential energy, ensuring energy conservation, ORB models predict forces directly in a non-conservative way. To satisfy physical constraints, these models correct the final prediction by subtracting the total force and torque.

4.2 Deformation Potentials

An important step to evaluate the adequacy and practical usability of NNPs as physical models is to qualitatively examine them to check for the presence of any fundamental flaws in the basic description of atomic interactions. The following tests aim to find qualitative defects in the potential energy description for bond distances, angles and torsions, for systems ranging from small diatomic molecules to larger molecules, and to determine the range of applicability of the models and where their description might fail and why.

4.2.1 2-body distortions. Bonds.

Bond distortions can be described using its distance (internal coordinate), which only depends on two atoms. In Chemistry, bonds (understood as a rearrangement of the electronic density) are formed due to a stabilization of the molecular system. The qualitative potential energy behavior for any bond is as described in Fig. 7 (black line) by a Morse potential. At $r \to \infty$ there is no net stabilization, as no formal interaction between the two atoms exists. As they come closer, stabilization increases due to delocalization of the electrons in a wider space, up to a minimum. Past the minimum, electron-electron and nuclear repulsion destabilizes the system, with the energy increasing asymptotically as $r \to 0$. This behavior is qualitatively analogous to a dispersion-based interaction described by a Lennard-Jones potential[15].

Small perturbations in internal coordinates with respect to the minimum energy structure are often described by a harmonic potential. Note, however, that a harmonic approximation cannot describe bond dissociation for large enough r, and is why a Morse potential is needed to correctly describe bond distortions beyond small $r - r_{\text{eq}}$.

To assess the qualitative behavior of the NNPs on bond deformations, a set of diatomic homonuclear molecules $\{H_2, N_2, C_2, O_2, Si_2, Cl_2\}$, as well as more complex molecules (ethane and ethanol) have been deformed to obtain their qualitative potential shapes, shown in Figs. 7, G.1, G.2 and 8, G.3, respectively.

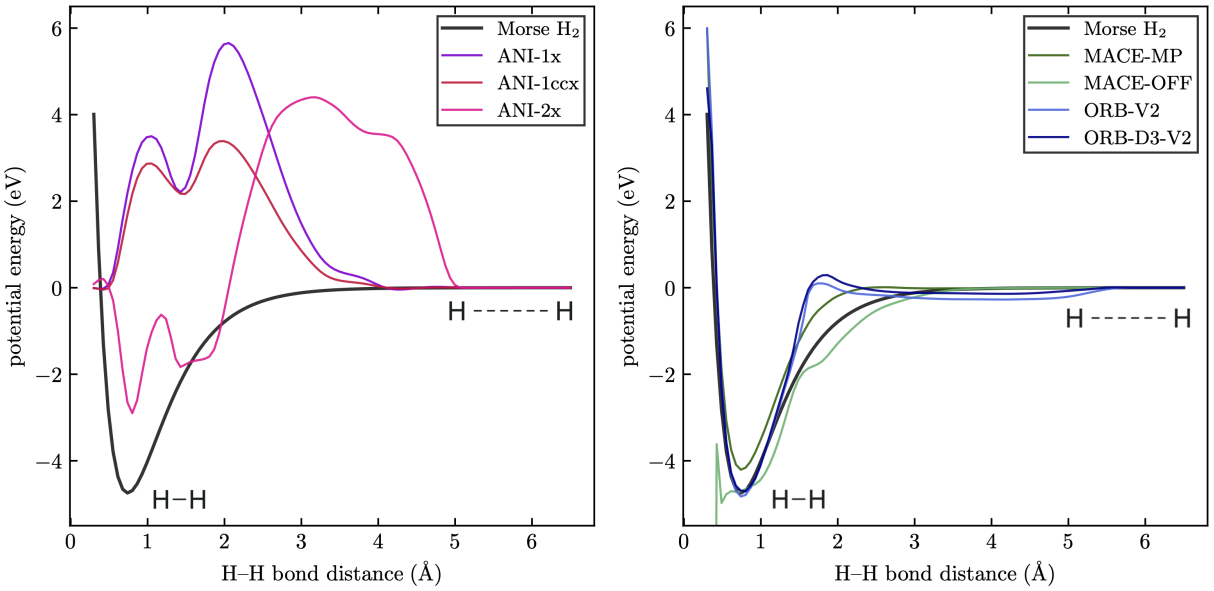
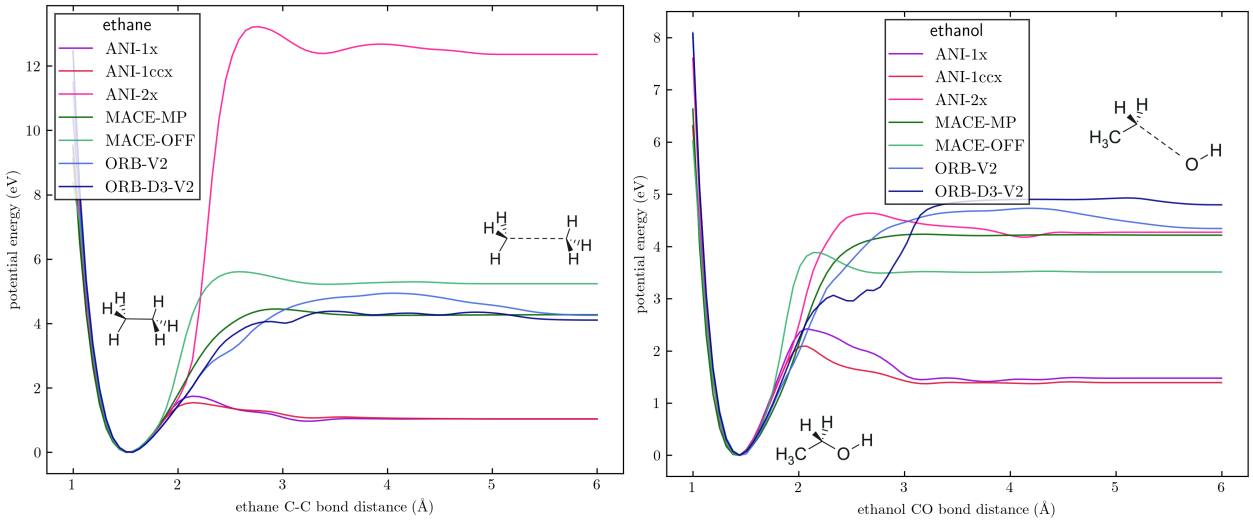


Figure 7: Dissociation potential of H_2. Energy reference is the dissociation limit ($r \to \infty$). In black, a qualitatively correct Morse potential for H_2. The models have been separated in two sets: (left) `ANI` models, which completely misdescribe the formation and dissociation of the molecule, and (right) `MACE` and `Orbital` models, which provide an overall qualitatively correct description of both the equilibrium distance and dissociation limit. Other diatomics are showcased in Figs. G.1 and G.2.



(a) ethane C–C bond deformation **(b)** ethanol C–O bond deformation

Figure 8: C–C and C–O bond deformation and dissociation of ethane and ethanol, respectively. Energies are represented with respect to the energy minimum. All models provide a qualitatively correct bond dissociation description, except for `ANI` potentials, which over- or underestimate it. Ethanol O–H bond dissociation is analogously showcased in Fig. G.3.

Taking a glimpse at the qualitative potential shape of Fig. 8, all models correctly predict a minimum for ethane and ethanol, which is to be expected, as that is the region the models have been trained on. When deforming the bond, however, different behaviors appear. `MACE` and `Orbital` correctly predict the dissociation shape, which increases monotonously as the distance of the bond increases, providing a qualitatively correct dissociation limit. `ANI` models, however, either overestimate or underestimate the dissociation potential, missing the

essential interactions. This qualitatively incorrect dissociation limit description is overly apparent in homonuclears (Fig. 7): while `MACE` and `Orbital` capture the essential interactions, resembling the qualitatively correct Morse potential, `ANI` completely misses the minimum and dissociation description, producing several fictitious minima with a large dissociation barrier preventing the bond formation and dissociation.

The correct description of the equilibrium distance and its dissociation is essential for applications such as modeling of chemical reactions, where bonds are formed and broken. The inconsistent dissociation barriers `ANI` describes in both figures would make it unsuitable for applications where deformed structures are relevant.

This qualitative misrepresentation of dissociations, however, is to be expected as none of the models have been trained on these high-energy deformed structures, only on the low-energy structures around the energy minimum (equilibrium distance).

Fig. 7 clearly shows the cutoff distance to be around 5 Å for all models, as seen by the abrupt convergence of all models. Past this point, according to the locality approximation, the two atoms 'do not see each other' and energy becomes analytically zero.

4.2.2 3-body distortions. Angles

A similar analysis can be carried out for three-body distortions, described by an A-B-C angle θ. As seen with bond deformations, angular minima can be modeled as $U \sim (\theta - \theta_{\text{ref}})^2$, with larger distortions breaking harmonicity.

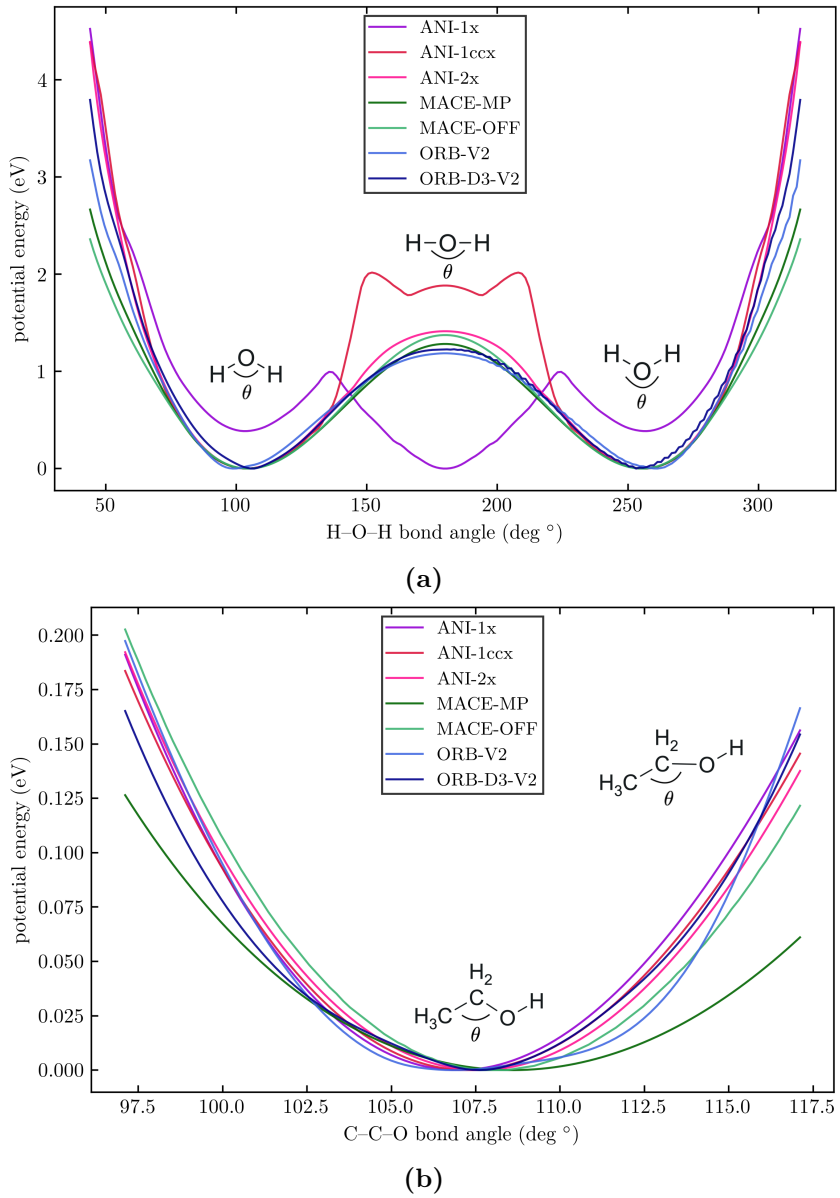


Figure 9: **(a)** Angular deformations for HOH angle in water and **(b)** CCO angle in ethanol.

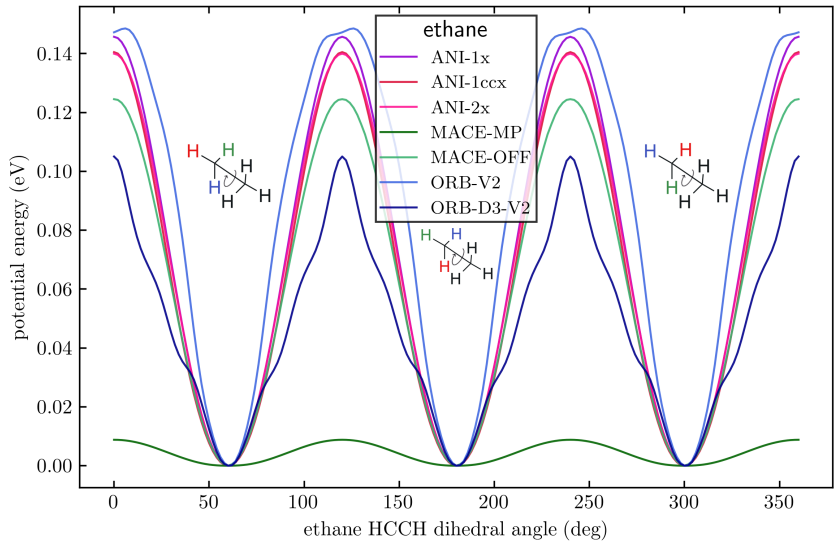
Fig. 9 shows the qualitative behavior of the models when these angular distortions are applied to water and the CCO angle in ethanol. All models correctly describe small angular perturbations harmonically, as seen in Fig. 9b. When investigating the complete HOH angular deformation of water (Fig. 9a), two symmetric minima appear, corresponding to the same-but-flipped molecule, with small angular changes around them described harmonically as with ethanol. Larger angular deformations break harmonicity, as seen by the appearance of a quadratic potential. All models correctly describe two minima (angular waters) connected by an energy maximum (linear water), however, `ANI-1x` and `ANI-1ccx` completely misunderstand the linear HOH transition, predicting a more stable linear water geometry or exaggerating the linear-geometry energy barrier, respectively.

This shortcoming is, once again, justified by the lack of training for large deformations, as linear water is far from equilibrium and not contained in the training set. It is interesting to note that `ANI-2x` correctly describes the linear transition, as this model improves torsional and angular deformations with respect to previous models[26].

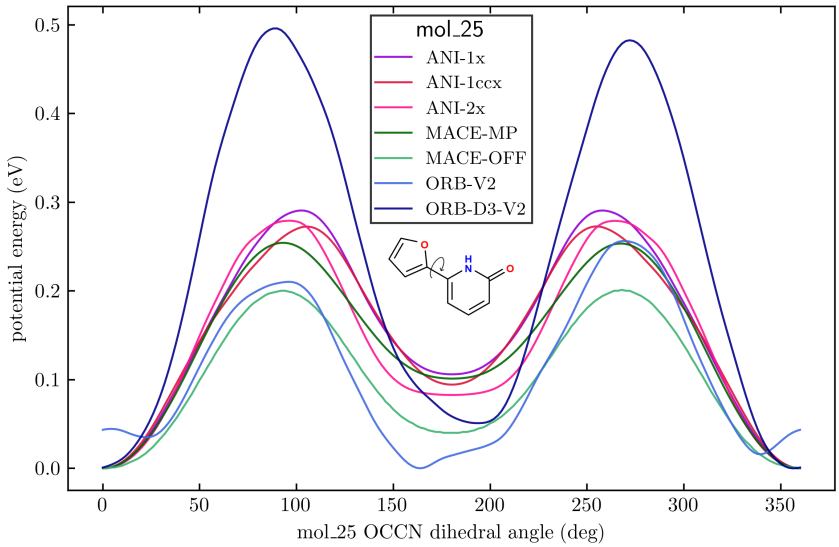
It is important to note that the potential energy shape must reflect the presence of certain symmetries in the molecule[6]: $\alpha°$ and $(360 - \alpha)°$ are equivalent water structures in Fig. 9a.

4.2.3 4-body distortions. Dihedrals

Lastly, usual deformation of molecules are rotations along bond axes. These rotations are described using dihedral angles formed by four atoms ABCD around the bond. The importance of correctly modeling these torsions stands from the fact that torsional minima are separated by small energy barriers, making molecules likely to 'jump' from one to another in simulations. Accurately describing these deformations is imperative, as they define the *flexibility* of a molecule, an especially important attribute for small organic molecules.



(a)



(b)

Figure 10: Potential energy scans of C–C bond rotations for ethane and a biphenyl-like molecule. Additional dihedral scans for other biphenyl-like organic molecules can be found in Fig. G.4. Biphenyl-like torsional barrier study is motivated by Ref. [73].

Fig. 10 showcases the torsional barrier descriptions of the HCCH and CCCC dihedrals (C–C bonds) of ethane and a byphenyl-like molecule. As seen with angles, these potentials must reflect the symmetries of the molecules, a correct description of which is extremely important for *in silico* simulations.

In the case of ethane (Fig. 10a), the models describe three equivalent minima and barriers, reflecting the molecule's symmetry. `MACE-MP`, however, completely neglects the torsional barrier, instead predicting free-rotation along the C–C bond. The dihedral in Fig. G.4b is much more difficult[73] due to all the non-C atoms of the molecule. It is important to note that, essentially, the clockwise or counterclockwise rotation of the bond produces equivalent results, which translates to a 180°-periodic torsional scan. While most models correctly identify this torsional symmetry, `Orbital` models do not, as reflected in the asymmetries of the potential curves, especially at $\sim 200°$. This is a consequence of the before-mentioned lack of rotational invariance in `ORB` models, and its biasing effects on *in silico* simulations should be carefully considered.

Overall, structural deformations are mostly qualitatively well described by these NNPs, with the notable exception of dissociation limits in `ANI` models and torsional and angular symmetries in `ORB` models.

4.3 Beyond symmetry: complex systems

The final step after evaluating the physical adequacy through assessing the symmetry and atomic interactions descriptions is to see how these two aspects work together to predict the energy of complex systems that heavily rely on symmetry. The following are two practical examples of molecular systems with degenerate and non-degenerate symmetry states, where we aim to evaluate how the models rank the stability and predict the energy barriers connecting them.

4.3.1 Non-degenerate symmetry system. COSAN.

COSAN (Fig. 11) is a cobalt complex with 3 validated symmetrically different states[74] (metastable states **A**, **B**; global minimum **C**), obtained by rotating one of its carbon-boron ligands with respect to the other, as depicted in Fig. 11. Each state is doubly degenerate, as clockwise and counterclockwise rotations go through the same three conformations. A qualitatively correct model would predict these states, their symmetries, and smoothly connect them with high-energy 'transition states'. This is exactly what `MACE-MP` achieves in Fig. 12a: a smooth symmetric potential showing that conformation **C** is the most stable in vacuum, in good agreement with Ref. [74] (see Table 5). As more C atoms overlap (from a top view perspective), the energy of the state increases, connected by high-energy transition states.

On the other hand, the description from `Orbital` models (Fig. 12b) is fundamentally flawed: it neither captures the rotational symmetry, instead predicting different energies for identical atomic arrangements, nor provides a smooth potential, as seen by the small fluctuations of high energy configurations. Furthermore, the energy scale is larger than with `MACE-MP`, potentially overestimating the transition energy.

This is a case where rotational invariance is essential to describe the system, and, as discussed previously, `Orbital` models are not formally invariant. This example showcases the need for

both a qualitatively correct description of atomic interactions, but most importantly, symmetry understanding to correctly describe nuanced systems and provide physically adequate results. While `Orbital` correctly captures interactions, its lack of symmetry understanding hinders its applicability substantially in real-world scenarios.

Finally, it must be noted that, while `MACE-MP` underestimates the energy differences between configurations (Table 5), it has been designed as a foundational model, that is, to provide a solid foundation with the ability to be fine-tuned for specific applications.

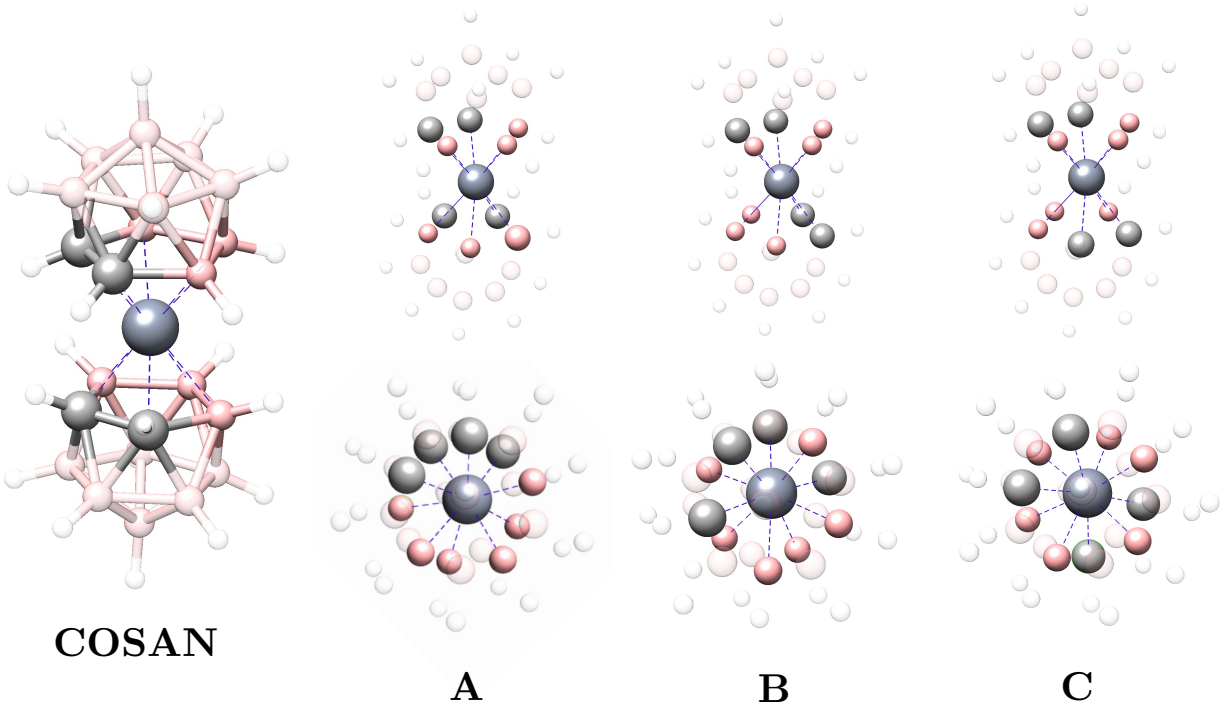
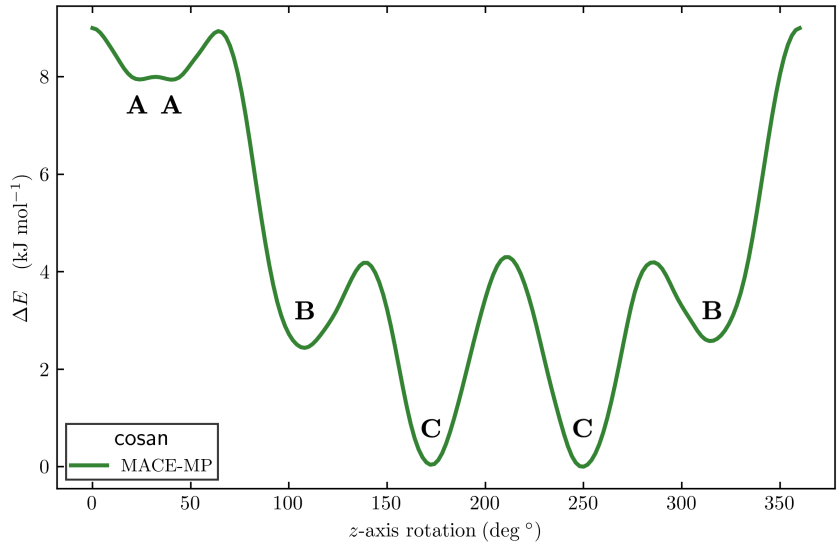
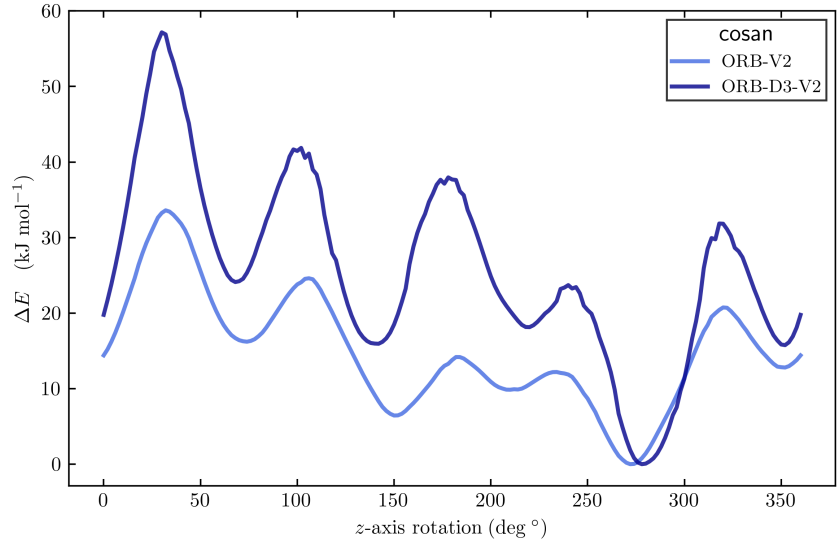


Figure 11: COSAN, a cobalt-containing anionic complex with three distinct symmetry states, labeled **A**, **B** and **C**. The overall structure (top), as well as the top view representation (bottom) of all three symmetry structures is provided. The central larger atom is cobalt (Co). Pink and grey atoms are boron (B) and carbon (C), respectively.



(a)



(b)

Figure 12: Potential energy scans along the rotation of the top-half cluster with respect to the bottom-half for **(a)** `MACE-MP` and **(b)** `ORB-v2` and `ORB-D3-v2` models. Other models cannot be used due to cobalt and boron being present in the structure. States in **(b)** could not be assigned due to the qualitatively incorrect depiction of the potential energy.

Table 5: ΔE (kJ mol^{-1}) comparison between `MACE-MP` (a general foundational NNP model) and QM calculations from Ref. [74] for all three conformations of COSAN. Energy is referenced with respect to conformation **C**.

	A conf.	**B** conf.	**C** conf.
`MACE-MP`	7.94	2.58	0.0
DFT (Ref. [74])	14.7	3.40	0.0

4.3.2 Molecular asymmetry. Cyclobutadiene.

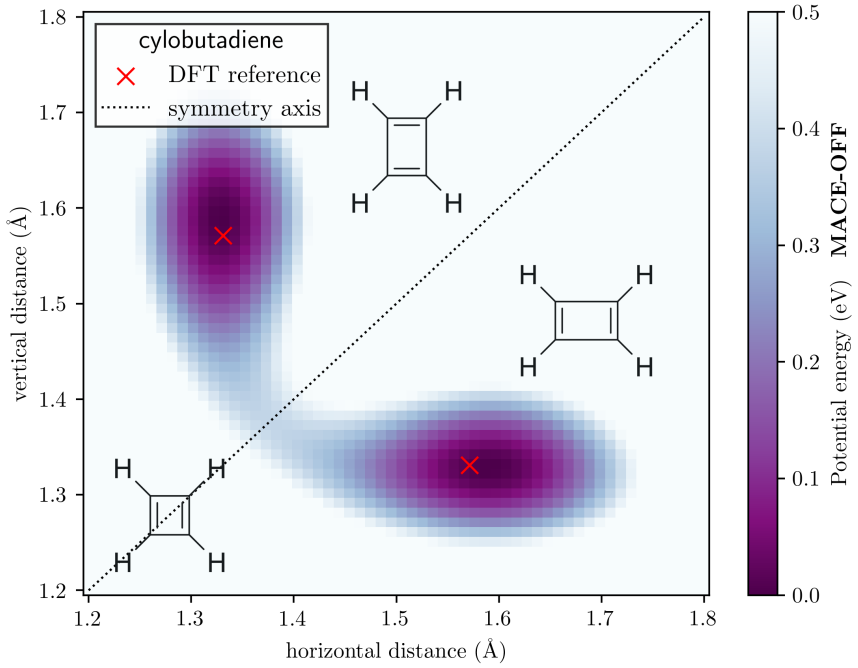
Symmetries are the source of energy degeneracy. There are some cases, however, where breaking degeneracy can lead to a more stable system. Cyclobutadiene (Fig. 13) is one of them, where the most symmetric square geometry is naturally deformed to a less symmetric rectangular one to break degeneracy and improve stability. The deformation can take

place in both x and y directions, generating two rotationally-identical, deformed, molecules. Thus, to correctly describe cyclobutadiene, models must understand its structural asymmetry (breaking the square degeneracy), but also deformation symmetry: x- and y-deformed molecules are identical.

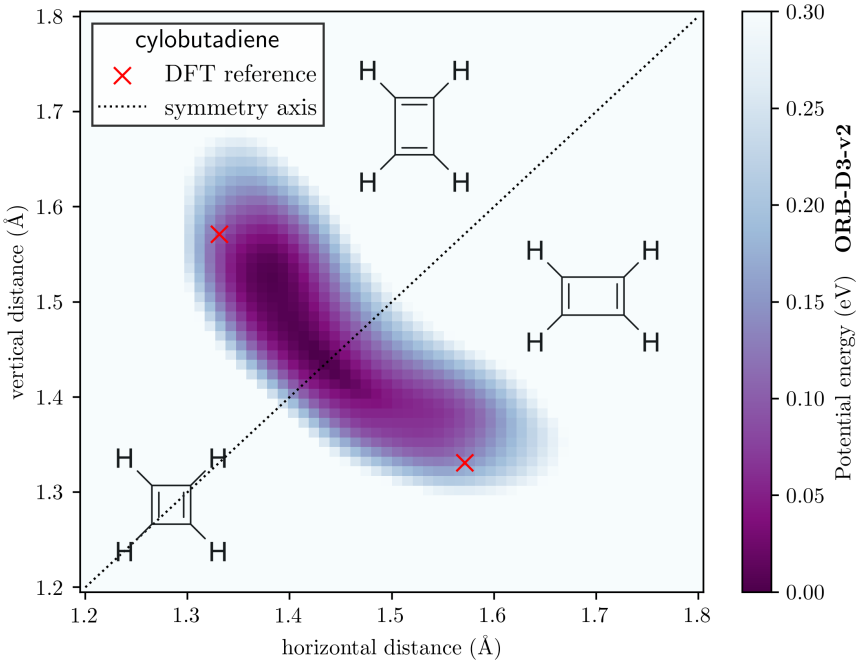
Fig. 13 shows the potential energy surface for each x and y distortion possible, with red crosses showing the reference DFT structure, optimized with wB97XD/6-31G(d) DFT functional and basis set using `Gaussian 16`[75]. Most models neglect the asymmetry of cyclobutadiene, instead predicting it to be a square molecule, as shown by the 1D deformation of Fig. G.5 and the surfaces in Figs. 13 and G.6. The only model that describes the rectangular geometry is `MACE-OFF` (Fig. 13a), with two identical minima situated exactly where the DFT reference is and the square geometry approximately 0.4 eV (9.2 kcal mol^{-1}) higher in energy.

On the other hand, `ORB-D3-v2` fails on both tasks: it predicts a symmetric square molecule, but most importantly, it favors y-axis deformations to x-axis, even though the resulting molecules are identical (rotated 90°), breaking symmetry. This bias of approximately 40 meV corresponds to the maximum energy prediction error when rotating the molecule along its z-axis, as shown in Fig. G.7d, due to the lack of rotational invariance.

These two examples remark the fact that a qualitatively correct understanding of the atomic interactions is not sufficient and a correct treatment of symmetries is necessary to achieve an adequate molecular description.



(a)



(b)

Figure 13: **(a)** `MACE-OFF` and **(b)** `ORB-D3-v2` energy surfaces for the distortion in the x- and y-axis of cyclobutadiene. Energies are represented with respect to the minimum energy structure. The symmetry axis when the distortion in both dimensions is equal is represented with a black dotted line. Red crosses represent the geometry of a reference cyclobutadiene structure (own work available at the `GitHub` repository). The x-only and y-only deformations scans can be found in Fig. G.5. Fig. G.6 showcases the same computed surface using other models.

5 Conclusions

Even though data-driven ML models have shown great applicability in data science, using them for physical tasks requires special consideration regarding their physical understanding, as motivated throughout this work. Particularly, the necessity of symmetry has led to invariant and equivariant models, that use *inductive biases* to understand symmetry at an architectural level.

The tests performed in this work show that translational and rotational invariance of energy, force and torque is satisfied in all models, with the notable exception of rotational invariance in `Orbital` models. The latter shows biases for particular atomic conformations of sometimes more than 1 kcal mol^{-1}, breaking space isotropy. These biasing effects should be studied further to assess whether rotationally augmented models can be used over invariant or equivariant ones in *in silico* simulations. Additionally, it seems that direct force prediction is an adequate, faster alternative to analytical gradients of the energy, as shown by the force adequacy of `Orbital` models.

When deforming a set of molecules, `MACE` models show a qualitatively correct description of angles, bonds and dihedrals. `ANI` models, however, fail to describe the dissociation limits of bonds, which might hinder their applicability in describing chemical reactions. Similarly, `Orbital` fails to describe the symmetries of biaryl torsions, which stems from the lack of rotational invariance. It is particularly remarkable how `MACE` and `Orbital` models capture bond deformations correctly, despite not being trained on these high-energy structures.

`MACE` equivariant models also show a qualitatively adequate description of interactions in symmetry-dependent systems such as COSAN and cyclobutadiene. `Orbital` models, on the other hand, fail to depict the symmetry-induced degeneracy of both systems, which is attributed to, once again, the lack of rotational invariance.

Overall, these results provide strong evidence to state that even though data is important in the raw performance of these models, their real-world performance is highly dependent on their physical foundations and the *inductive biases* included in the *design* of these models, as a physically-flawed model shows little applicability for physical tasks.

Finally, it is worth remarking how rapid-moving the field of ML is and its future importance as a scientific tool to accelerate research now, as seen by the 2024 Nobel Prizes in Physics and Chemistry, and in the upcoming years.

A *Ab initio* electronic structure methods

A.1 Born-Oppenheimer approximation

One of the central aspects in theoretical and computational chemistry is the development of *ab initio* (first principles) quantum chemistry methods[11–14] to solve the time-independent schrödinger equation (TISE),

$$\hat{H}\,|\Psi\rangle \equiv \left[-\frac{\hbar}{2m}\hat{\nabla}^2 + \hat{V}\right]|\Psi\rangle = E\,|\Psi\rangle \ . \tag{A.1}$$

Let t be time, N and M the number of electrons and nuclei, respectively, and $\{\mathbf{R}\}$ and $\{\mathbf{r}\}$ the position of the M nuclei and N electrons, respectively. The system wavefunction (in coordinate representation) is

$$\Psi(t, \{\mathbf{r}\}, \{\mathbf{R}\}) \equiv \Psi(t, \mathbf{r}_1, \ldots, \mathbf{r}_N, \mathbf{R}_1, \ldots, \mathbf{R}_M) \ . \tag{A.2}$$

The TISE is only valid in case the potential is independent of time, $V(t, \{\mathbf{r}\}, \{\mathbf{R}\}) \to V(\{\mathbf{r}\}, \{\mathbf{R}\})$. In that case,

$$\Psi(t, \{\mathbf{r}\}, \{\mathbf{R}\}) = \phi(t)\psi(\{\mathbf{r}\}, \{\mathbf{R}\}) \,, \tag{A.3}$$

and the TISE simply becomes $\hat{H}\psi = E\psi$. However, solving it for a general system of arbitrary number of atoms and electrons is impossible, as nuclear and electronic degrees of freedom might be coupled.

An initial approximation based on the mass ratio of the proton and electron is called the **Born-Oppenheimer approximation**[76], in which the electronic and nuclear motion is decoupled based on the different time-scale of the 'motion' of both nuclei and electrons. This approximation implies that no mixing of different electronic stationary states happens due to the interaction between the nuclei (electrons do not undergo transitions between stationary states) and hence no heat exchange between nuclei and electrons is observed. The adiabatic approximation is only valid if the energy associated to nuclear vibrational degrees of freedom is not comparable to the electronic energy.

Under the BO approximation, the wavefunction becomes

$$\psi(\{\mathbf{r}\}, \{\mathbf{R}\}) \quad \longrightarrow \quad \psi(\{\mathbf{r}\}; \{\mathbf{R}\}) \,, \tag{A.4}$$

and only depends *parametrically* on the nuclear positions. The general Hamiltonian of the system is (in atomic units)

$$\begin{aligned}\hat{H} &= -\sum_i^N \hat{\nabla}_i^2(\mathbf{r}_i) - \sum_\alpha^M \frac{m_e}{m_\alpha}\hat{\nabla}_\alpha^2(\mathbf{R}_\alpha) - \sum_i^N\sum_\alpha^M \frac{Z_\alpha}{r_{i\alpha}} + \sum_i^N\sum_{j>i}^N \frac{1}{r_{ij}} + \sum_\alpha^M\sum_{\beta>\alpha}^M \frac{Z_\alpha Z_\beta}{r_{\alpha\beta}} \\ &\equiv \hat{T}_N(\{\mathbf{R}\}) + \hat{T}_e(\{\mathbf{r}\}) + \hat{V}_{Ne}(\{\mathbf{r}\}, \{\mathbf{R}\}) + \hat{V}_{ee}(\{\mathbf{r}\}) + \hat{V}_{NN}(\{\mathbf{R}\}) \,,\end{aligned} \tag{A.5}$$

with $\hat{T}_N$ and $\hat{T}_e$ the kinetic energy of the nuclei and the electrons, respectively, and $\hat{V}_{Ne}$, $\hat{V}_{ee}$, $\hat{V}_{NN}$ the potential energy due to nucleus-electron, electron-electron and nucleus-nucleus interaction. The Hamiltonian can be separated into a nuclear and electronic contributions, separating the dependencies,

$$\begin{aligned}\hat{H} &= \left(\hat{T}_N(\{\mathbf{R}\}) + \hat{V}_{NN}(\{\mathbf{R}\})\right) + \left(\hat{T}_e(\{\mathbf{r}\}) + \hat{V}_{Ne}(\{\mathbf{r}\},\{\mathbf{R}\}) + \hat{V}_{ee}(\{\mathbf{r}\})\right) \\ &\equiv \hat{H}_{\text{nuclear}} + \hat{H}_{\text{electronic}}\,,\end{aligned} \tag{A.6}$$

where the nuclear Hamiltonian only depends on the coordinates of the nuclei. As ψ depends parametrically on $\{\mathbf{R}\}$, if nuclear coordinates are fixed, $\hat{H}_{\text{el}}$ will only depend on electronic coordinates. The potential energy of a given nuclear arrangement can be trivially determined from the nuclear repulsion term,

$$\hat{H}(\{\mathbf{R}\}) = \hat{H}_{\text{nuclear}} + \langle\hat{H}_{\text{el}}\rangle = \hat{T}_N + \left(\hat{V}_{NN} + \langle\hat{H}_{\text{el}}\rangle\right) = \hat{T}_N(\{\mathbf{R}\}) + U(\{\mathbf{R}\})\,, \tag{A.7}$$

which is represents the movement of classical nuclei under the potential due to the average field of the electrons. This provides a way to obtain the Potential Energy Surface (PES) for any given system: for a given atomic configuration (set of $\{\mathbf{R}\}$), the electronic TISE is solved and the eigenfunctions and eigenvalues are collected and the potential energy is obtained. Iteration over all nuclear arrangement provides the PES, which can be used to compute the nuclear wavefunction and eigenvalues corresponding to rotovibrational motion.

A.2 Hartree-Fock theory introduction

In order to solve the electronic TISE to obtain the wavefunction and energies, we must deal with $\hat{V}_{ee}(\{\mathbf{r}\})$, which introduces the so-called 'electron correlation', where the position of a single electron is determined by the positions of all other electrons, and these are determined likewise (many-body problem).

Hartree-Fock theory[11] tries to work around this issue to devise a method to obtain the electronic energies and wavefunctions. As a first approximation, the many-body problem is converted into a many-particle problem neglecting electron correlaction completely and introducing Self-consistent Field (SCF) theory (or mean-field theory), under which each electron is 'effectively' independent, only interacting with other electrons in an average way.

Note that, as electrons are fermions, the permutation of any pair of electrons must change the sign of the wavefunction. This, as a consequence, means that the probability density of finding two electrons of the same spin in the same space is zero (what is refered to as the Exclusion principle). Let Ψ be the N-electron wavefunction, $\Psi(\mathbf{r}_1,\ldots,\mathbf{r}_N)$, we can build it as an antisymmetrized product (Slater determinant) of a set of one-electron wavefunctions $\psi_j(\mathbf{r}_i)$,

$$\Psi(1,\ldots,N) = \frac{1}{\sqrt{N!}}\begin{vmatrix}\psi_1(1) & \psi_2(1) & \ldots & \psi_N(1)\\ \psi_1(2) & \psi_2(2) & \ldots & \psi_N(2)\\ \vdots & \vdots & \ddots & \vdots\\ \psi_1(N) & \psi_2(N) & \ldots & \psi_N(N)\end{vmatrix} \tag{A.8}$$

with $i \equiv \mathbf{r}_i$ the coordinate of electron i. One last important approximation in the HF theory is the usage of basis functions φ_i such that each one-electron wavefunction is written as a linear combination of basis functions, such that

$$\psi_i = \sum_j^K c_{ij}\varphi_j \tag{A.9}$$

The basis set $\{\varphi_i\}$ contains K basis functions that provide a functional description of the one-electron wavefunctions. Note that each ψ_i might be delocalized over the whole molecule,

as φ_j may be centered in any given atom of the molecule.

The final step to obtain the energy of the configuration (fixed $\{\mathbf{R}\}$) is to apply the variational theorem to optimize the set of coefficients $\{c_{ij}\}$ that minimize the electronic energy,

$$E_{\text{el}}^{*} \equiv \min_{\{c_{ij}\}\in\mathbb{R}} \frac{\langle\Psi(1,\ldots,N)|\,\hat{H}_{\text{el}}(1,\ldots,N)\,|\Psi(1,\ldots,N)\rangle}{\langle\Psi(1,\ldots,N)|\Psi(1,\ldots,N)\rangle} \geq E_{\text{el}}^{\text{real}} \tag{A.10}$$

which can be done solving the Roothan-Hall equations to provide the 'best' Slater Determinant for the molecule. Note that, to determine the coefficients of molecular orbital ψ_j, knowledge of all the other s is needed. This turns the SCF into an iterative procedure (Roothan-Hall equations are solved until convergence, or in other words, until the electronic density is self-consistent with the field it generates).

A.3 Electron correlation methods

In the limit of infinite basis set size (optimization over all functional space), the energy converges to the so-called Hartree-Fock limit. However, due to the lack of complete electron-electron correlation, the energy is larger than the 'exact' value due to the mean-field approximation: electrons avoid each other only on average, not as point particles. In relative terms, the correlation energy ranges from $0-2\%$, however, it is significant in absolute terms, making it a crucial energetic contribution for deriving molecular properties.

Two different approaches can be taken to introduce correlation[11] or 'recover' the missing correlation energy in the Hartree-Fock method. The computational cost of usual electronic structure methods can be found in Table 6.

- **Wavefunction methods.** Are based on expressing $\Psi(1,\ldots,N)$ as a linear combination of excited Slater Determinants, which allow for more flexibility when describing the real wavefunction and can be systematically improved including higher-order excitations. These methods include: Configuration Interaction methods (CI), Coupled Cluster methods (CC), Møller-Plesset perturbation theory methods (MP) and Multiconfiguratinal SCF methods (CASSCF, etc.)
- **Density functional methods.** Try to predict the energy of the configuration directly from the electron density using an exchange correlation functional within the Kohn-Sham formalism. Depending on what information of the density is used, several exchange-correlation types arise: Local density approximation (LDA), Generalized gradient approximation (GGA), Hybrid functionals.

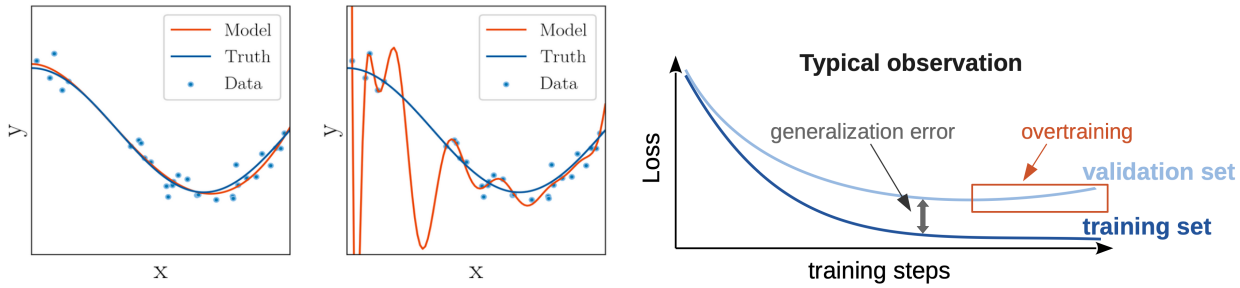
Table 6: Computational cost scaling in terms of the number of basis functions, K, for common electronic structure methods[12]. For DFT, the exact scaling depends on the exchange-correlation functional used. Scaling shown is before code-implementation optimizations.

Electronic structure method	Computational cost scaling
Hartree-Fock	$\mathcal{O}(K^4)$
DFT	$\mathcal{O}(K^3)$
MP2	$\mathcal{O}(K^5)$
CCSD(T)	$\mathcal{O}(K^7)$

B Further ML considerations

B.1 Generalization error, underfitting and overfitting

The goal of the model is to try to learn the underlying representation of the input data (e.g. for a regression task, learn the underlying true function that generates the points, as seen in Fig. B.1a). To track the generalization performance while and after training, subsets of the dataset are generated. For training, a *training* and *validation* set are used, the former to actually train the model and the later to assess its performance while training (but not to train it). The rest of the dataset is used to build a *test* set, used to assess the final performance after the model has been trained.



(a) **(b)**

Figure B.1: **(a)** Overfitting of a model. Underfitting is caused by the lack of parameter flexibility of the model, not being able to capture the true representation of the model. Overfitting, on the other hand, can be caused by excessive model complexity or overtraining. **(b)** Typical ML training observation. With training the model reaches a minimum generalization error, but this gap increases due to overtraining. Extracted from Ref. [29].

Model complexity plays a crucial role in determining the underlying representation: an overly simple model with very few trainable parameters will not be able to capture the underlying representation of the data due to the lack of 'flexibility' to represent the underlying function, even if the model is trained for many epochs[b], in which case the model is said to **underfit**. On the other hand, very complex models with a lot of trainable parameters are prone to create a complex function that minimizes training error but miss the underlying representation. In this case, the model 'memorizes' the training set and is unable to generalize to new data, which leads to **overfitting**.

Overfitting can be detected as an increase in generalization error (between the training and validation sets), where, the model starts to 'memorize' the training set due to overtraining. This can be mitigated by stopping the training where generalization error is minimum (as seen in Fig. B.1b).

[b] An epoch of training ≡ training over all examples in the training set once.

C Graph Neural Networks (GNNs)

In ML, we often need to tailor the architecture to the data format of a specific task to obtain a data-efficient model, that is, one that can take advantage and exploit the structure and symmetries in the data. A simple case is image recognition: the MNIST dataset containing hand-written digits from 0 to 9 can be 'solved' with a MLP architecture, but is not data-efficient, as MLPs do not satisfy translational invariance of the features in images nor can be applied to images of variable dimensionality. Instead, Convolutional Neural Networks[32] (CNNs) are the go-to model for image recognition, as they are able to exploit the translational symmetry of the features in images and are extendible to large images, while being parameter-efficient.

The need for GNNs is apparent when we consider how many data structures can naturally be codified as graphs. A first intuitive example is molecules, where we can think of each atom as a node and each bond as an edge connecting the nodes, as seen in Fig. 3. Other type of data that can be modeled as a graph are images (each node is a pixel, each with a feature vector containing the corresponding RGB values), the citations in scientific papers, maps (where each intersection is a node, connected by roads), social interaction in social media, etc. Hence, it is interesting to develop a ML architecture that can efficiently exploit the underlying structure of the data.

This section aims to provide a solid, although not exhaustive, description of the GNN architecture based on Refs. [77–79]. The structure will be as follows. First, some math preliminaries about GNNs are given. Second, the barebones architecture is formulated to gain intuitive insight on it. Third, the message passing neural network (MPNN) and its variants are introduced. Finally, particular considerations when applying GNNs to computational chemistry are discussed.

C.1 Mathematical preliminaries

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is formed by nodes (or vertices) $\mathcal{V}$ which are connected through edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. These edges can be directed or undirected to specify the relationship between nodes. We will first start by considering graphs with no edges, and then move on to consider the connectivity of a graph.

Graphs in ML encode the features of a given example in a particular representation (embedding) in both nodes and edges. Let k be the number of node features, we define the node feature vector of node $i = 1, \dots, N$ as $\mathbf{x}_i \in \mathbb{R}^k$ and the corresponding feature matrix $\mathbf{X}$ as a stack of them

$$\mathbf{X} = \begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ & \vdots & \\ — & \mathbf{x}_N & — \end{bmatrix} .$$

Note that, by stacking all feature vectors, we are assigning a label (a number) to each node, but these are arbitrary: all permutations of $\mathbf{X}$ are equally as valid and thus the model must return the same value for each permutation (see Fig. C.1). If we define f as a function that operates on all the features of the graph, $\mathbf{X}$, we must impose f to be **permutation**

invariant such that $f(\mathbf{PX}) = f(\mathbf{X}) \quad \forall \mathbf{P} \in \Sigma$ with $\mathbf{P}$ the permutation matrix, $\mathbf{P} \in \mathbb{R}^{N\times N}$ (and $\Sigma \subset \mathbb{R}^{N\times N}$ the group that contains all $N!$ permutation matrices). This means that if we were to change the order in which we present the nodes to f, the result would be the same. A general permutation invariant function could be

$$f(\mathbf{X}) = \phi\left(\bigoplus_{i\in\mathcal{V}} \psi(\mathbf{x}_i)\right) , \tag{C.2}$$

where ψ operates on a single node, and all that information is aggregated in a permutation-invariant way with $\bigoplus$, which can be a sum, average or maximum operation. The resulting feature vector will be passed to ϕ to make the final prediction.

This, however, is only useful if we want to determine a property of the whole graph, i.e. a property given by the node features $\mathbf{X}$. However, it might be interesting to perform node-wise predictions, where feature vectors are updated to obtain a set of latent feature vectors $\mathbf{h}_i$ (in matrix form $\mathbf{H} = g(\mathbf{X})$, with $\mathbf{H}$ the latent feature matrix of the graph, defined similarly as to $\mathbf{X}$). If we let g be permutation invariant, the ordering of 'which part of the output belongs to which node' is lost. Thus, we must require g to be **permutation equivariant**, that is, the ordering of the output must be preserved independently of whether the permutation is performed before or after the operation, i.e. $g(\mathbf{PX}) = \mathbf{P}g(\mathbf{X})$.

We can combine both invariant and equivariant operations to provide a framework to obtain global features from per-node features. This can simply be done by defining a local, shared and equivariant operation ψ on each node, that updates each node's features $\mathbf{h}_i = \psi(\mathbf{x}_i)$ to form $\mathbf{H}$, from which we can obtain per-node properties. Then, we can combine all features with a permutation invariant operation, $\bigoplus$, to aggregate all latent features and apply ϕ over the final 'pooled' feature vector to obtain the final global property.

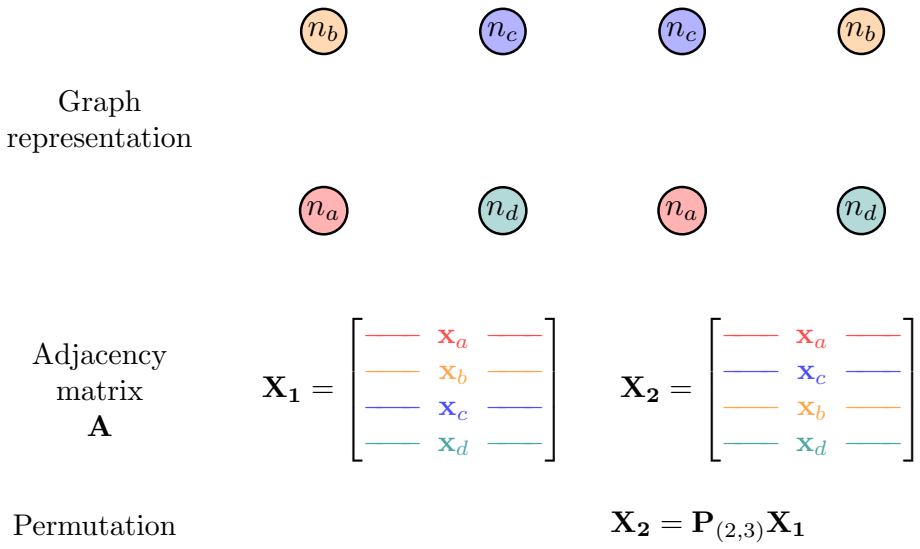


Figure C.1: Two identical graphs, with only nodes $\mathcal{V}$ and no edges $\mathcal{E}$. The permutation matrix $\mathbf{X}$ is build by stacking the feature vectors $\mathbf{x}_i$ of each node, which necessarily imposes an arbitrary ordering of the nodes. Permutations (using a permutation matrix $\mathbf{P}$) of the rows of $\mathbf{X}$ contain the same information, and as such, a GNN model should return the same output. This property of graphs give rise to **permutation invariant** transformations on all the nodes, i.e. the condition $\mathbf{v} = f(\mathbf{X}) = f(\mathbf{PX})\ \forall \mathbf{P}$ where $\mathbf{v}$ is a global property of the graph obtained from $\mathbf{X}$. In the example, the labels of nodes n_b and n_c are swapped, as a result $\mathbf{X_2}$ is a permutation of the initial feature matrix $\mathbf{X_1}$.

Until now, we have not yet considered graph connecitivity. Connectivity is determined by edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, which can be represented using an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ such that

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}, \tag{C.3}$$

which encodes which nodes are connected. We consider undirected edges, meaning that $\mathbf{A}$ is symmetric, and that each node is not connected to itself, i.e. $\mathbf{A}_{ii} = 0$. Similarly to nodes, each edge has an edge feature vector $\mathbf{e}_{ij}$, not necessarily the same dimensionality as k. If we consider a non $\mathbf{0}$ adjacency matrix (we have edges) the conditions for equivariance and invariance not only apply to nodes, but also to edges. This can be described in the following way,

$$\text{invariance} \qquad f(\mathbf{PX}, \mathbf{PAP}^\top) = f(\mathbf{X}, \mathbf{A}) \tag{C.4}$$

$$\text{equivariance} \qquad f(\mathbf{PX}, \mathbf{PAP}^\top) = \mathbf{P}f(\mathbf{X}, \mathbf{A}) \tag{C.5}$$

where $\mathbf{PAP}^\top$ means that the permutation is applied to both rows and columns of $\mathbf{A}$. Fig. C.2 shows two isomorphic graphs, and although $\mathbf{A}$ is different, the connectivity remains the same and is why permutation invariance on nodes and edges is required for any graph $\mathcal{G}$.

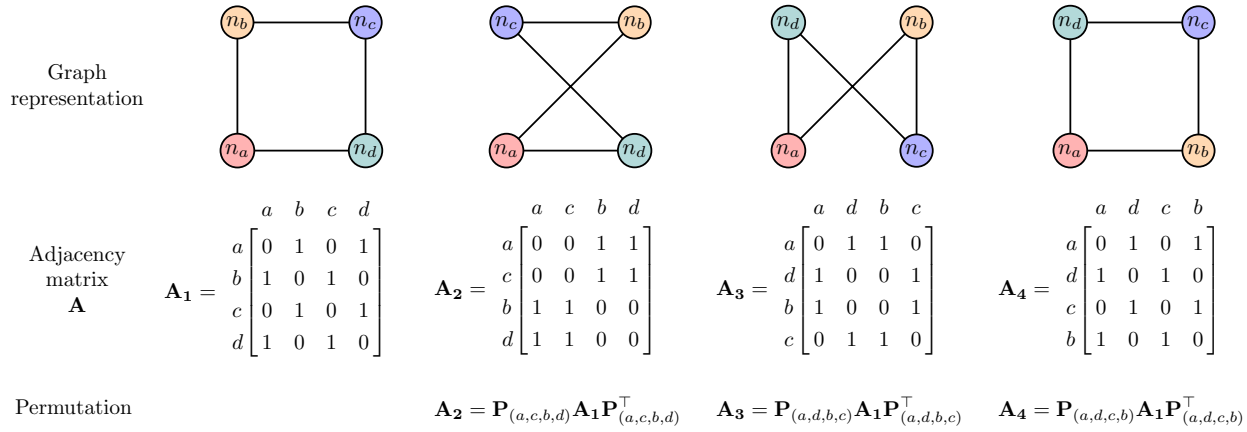


Figure C.2: Set of four isomorphic graphs, all with the same connectivity but a different choice of labels to represent the adjacency matrix $\mathbf{A}$.

Another important definition is the 1-hop (immediate) neighborhood of node i

$$\mathcal{N}_i = \{u \mid \{u, v\} \in \mathcal{E}\} . \tag{C.6}$$

We can extract a multiset of features of the neighborhood

$$\mathbf{X}_{\mathcal{N}_i} = \{\mathbf{x}_j \mid j \in \mathcal{N}_i\} \tag{C.7}$$

which excludes the own node's $\mathbf{x}_i$. This allows us to define a local, shared, permutation invariant function h that operates on each node and its neigborhood, $h(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i})$, to update each node's value as seen in Fig. C.3.

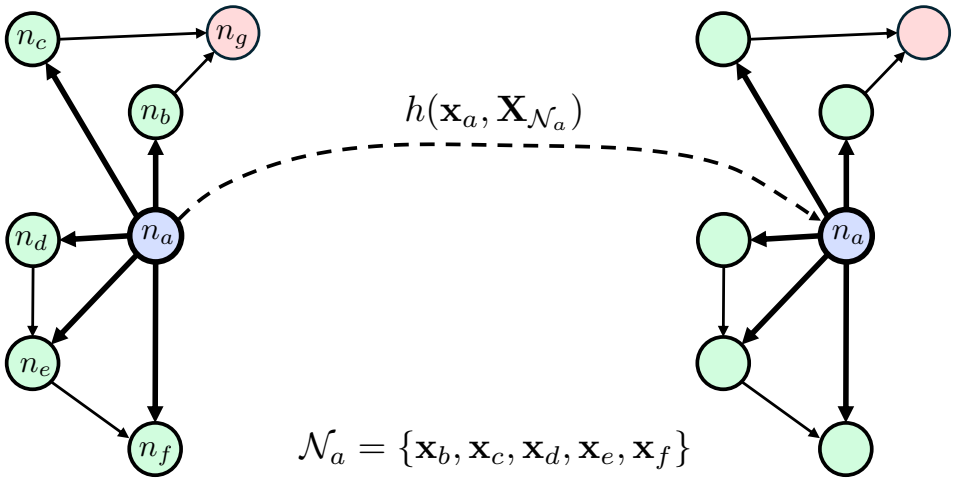


Figure C.3: Visualization of the per-node update mechanism. The feature vector $\mathbf{x}_a$ of node n_a is updated using a shared local permutation invariant function h that operates on the own node (in blue) and its 1-hop neighborhood (in green). Nodes in red do not take part in the node update. This operation must be permutation invariant, as the labeling of the neighborhood is arbitrary.

We now return to search a blueprint for obtaining per-node and global properties, as done in the node-only case. To start, we need to find a function g that updates the features of all nodes, i.e. $g : (\mathbf{X}, \mathbf{A}) \to (\mathbf{H}, \mathbf{A})$. This transformation must be equivariant as discussed when we only had nodes. To do this, we can simply build $\mathbf{H}$ by stacking the latent feature vectors obtained via $\mathbf{h}_i = h(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i})$, with h invariant and where the transformation includes its own node (as was for ψ in the only-nodes case), but also the 1-hop neighbors due to the connection through edges. This leads to

$$\mathbf{H} = g(\mathbf{X}, \mathbf{A}) = \begin{bmatrix} — & h(\mathbf{x}_1, \mathbf{X}_{\mathcal{N}_1}) & — \\ — & h(\mathbf{x}_2, \mathbf{X}_{\mathcal{N}_2}) & — \\ — & \vdots & — \\ — & h(\mathbf{x}_N, \mathbf{X}_{\mathcal{N}_N}) & — \end{bmatrix}, \tag{C.8}$$

with h the permutation invariant shared local function, and g the permutation equivariant 'update' function. Note that g returns a graph with the same connectivity as the input graph, $\mathbf{A}$.

C.2 Barebones GNN architecture

As seen in CNNs, ML models typically take rectangular or grid-like arrays as input[c]. For a graph $\mathcal{G}$, it is not obvious how we can process its data, as each node might be of different *degree* (contain a different number of nodes in its neighborhood). The GNN architectures we will introduce can take advantage of the graph structure and solve the different connectivites of each node.

The GNN architecture takes the graph connectivity ($\mathbf{A}$ or different representation), node and edge embeddings $\{\mathbf{x}_i\}$, $\{\mathbf{e}_{ij}\}$, respectively, and performs operations to update its features, from which per-node and global properties of the graph can be determined. This

[c]For non-numeric data, such as strings, a one-hot encoding is used (grid-like). If the dimensionality is large, the one-hot encoding becomes sparse and reduced representations are used (autoencoder architecture).

is a '*graph-in, graph-out*' approach that satisfies graph symmetries and exploits the graph representation, as seen in Fig. C.4.

In a GNN architecture, the input graph is transformed by a sequence of GNN layers, producing an output graph with latent node and edge embeddings. To update the features, function $g : (\mathbf{X}, \mathbf{A}) \rightarrow (\mathbf{H}, \mathbf{A})$ must be equivariant. In the simplest form, each node and edge is transformed independently by an invariant function ψ only taking information of the own's node or edge, and not the neighborhood (which makes the invariant condition trivial). It is important to note that the *same* function ψ is applied for all nodes[d] (and or edges), decreasing the number of trainable parameters substantially.

Once the nodes have been updated, the latent feature matrix $\mathbf{H}$ can be built. As happens with deep architectures, these GNN layers can be stacked to update the features in each layer. It must be noted that ψ is often a MLP, taking the node or edge features and producing a latent feature output $\mathbf{h}_i \in \mathbb{R}^m$, with m the latent feature dimension, which can be different to k the initial input dimension. Each GNN layer containts a different ψ function.

Once the output graph is produced, per-node and global properties can be computed using permutation equivariant and invariant operations, respectively. For global properties, the blueprint of Eq. (C.2) is followed: first each node output is aggregated in a permutationally-invariant way and then ϕ (usually another MLP) is used for the final prediction.

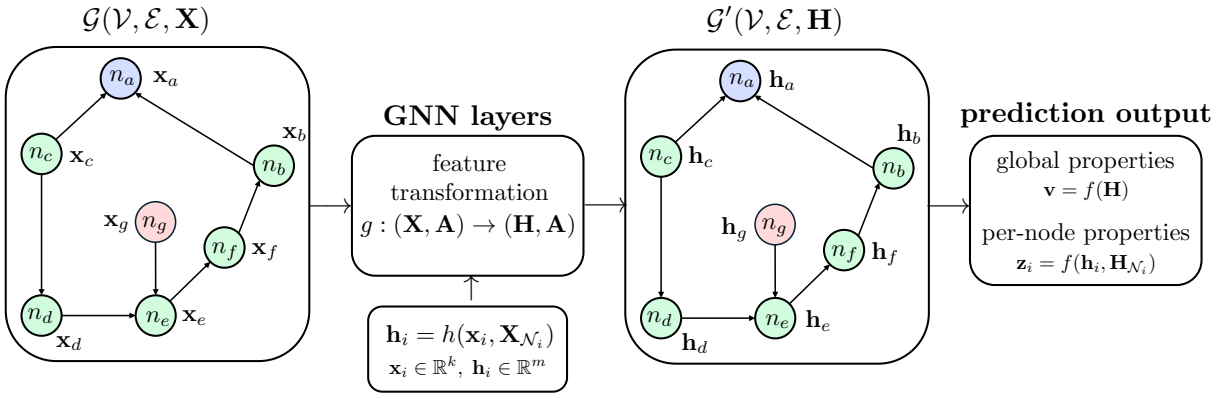


Figure C.4: General workflow of a GNN architecture. First, an initial graph $\mathcal{G}$ is defined with connectivity $\mathbf{A}$ and node and edge embeddings $\mathbf{x}_i$ and $\mathbf{e}_{ij}$, respectively. This graph is transformed to a different graph $\mathcal{G}'$ by different GNN layers, which are fundamentally formed by a *permutation invariant* ' layer' update function $g : (\mathbf{X}, \mathbf{A}) \rightarrow (\mathbf{H}, \mathbf{A})$ formed of a per-node *permutation invariant* 'message passing' function $h(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i})$. Depending on how h is defined, different types of GNNs arise. Once the output graph has been generated, global or per-node attributes can be computed, normally using an MLP for regression or classification. A more general pipeline overview can be found in Ref. [78].

The general blueprint for updating all the nodes of a graph is Eq. (C.8), with g often referred to as the 'propagation', 'update' or 'message passing' function. However, the most important step is to decide how each node is updated, i.e. define how $h(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i})$ updates node i based on the features of the neighboring nodes. Regarding this function, there are three main 'flavors' of GNNs depending on how the neighboring information is used: GCNs[80], GATs[81]

[d] This function reuse is similar to *filters* in CNNs, which were reused for different images.

and MPNNs[82] (Fig. C.5). The particular case where no neighbor information is used has been discussed in this barebones GNN architecture section.

C.3 Message-passing GNNs

The idea of message-passing GNNs is to define the function h of Eq. (C.8) to include the data (feature vectors) of neighboring nodes. This way, each node is made 'aware' of the data in its neighbors. Depending on how this mechanism is modeled we have convolutional, attentional and message-passing GNN architectures, as seen in Fig. C.5.

The core mechanism for all architectures is the same and is depicted in Fig. C.6. For each node, we prepare the features of neighboring nodes in some way ('prepare the messages') as well as the own node's features. All these features are aggregated by a permutation invariant operation and are mapped by ϕ to the updated feature vector, often by a MLP.

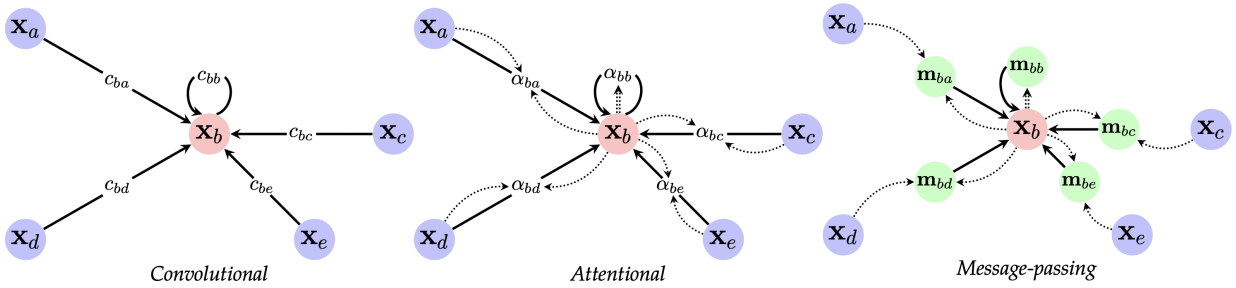


Figure C.5: Different ways of how the permutation invariant operation h can be defined. Depending on how the messages are produced, three main architecture types exist: convolutional (GCN), attentional (GAT), and message-passing (MPNN). α_{ij} refer to the normalized attention coefficients $a(\mathbf{x}_i, \mathbf{x}_j)$ over the whole neighborhood using a softmax function. Figure extracted from Ref. [77].

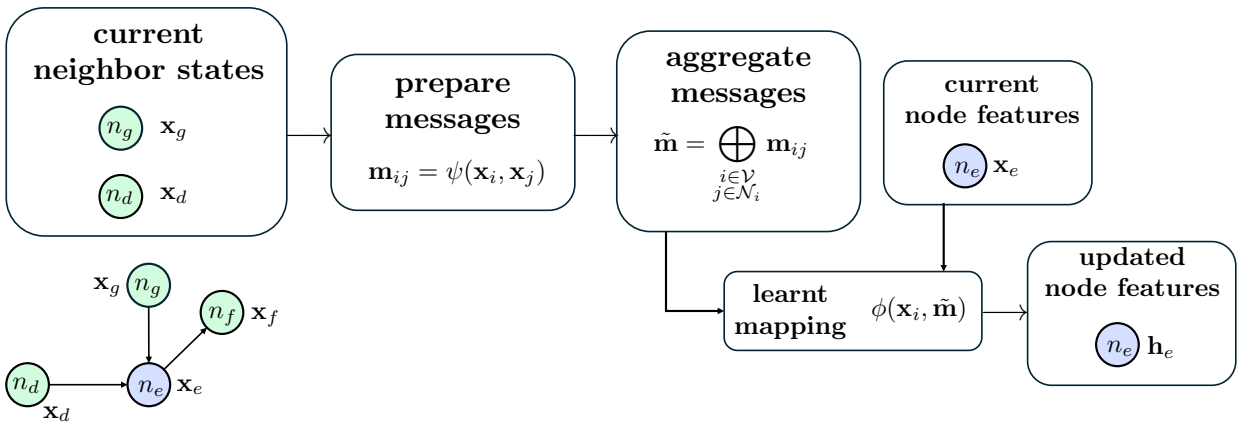


Figure C.6: Core workflow of a general message-passing GNN. The general per-node propagation step can be seen in Fig. C.3. Given a node n_e and its neighbors (n_g, n_d) with their respective features, the messages $\mathbf{m}_{ij}$ are prepared with a function ψ, taking information of the given node and its neighbors. The way messages are prepared are the main differentiating factor of different GNN architectures. The messages are then aggregated in a permutation-invariant way to ensure the whole nodeupdate process is permutation invariant.

C.3.1 Graph Convolutional Networks (GCNs)

For GCNs, h is defined as follows

$$\mathbf{h}_i = h(\mathbf{x}_i, \mathbf{X}_{\mathcal{N}_i}) = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right) \tag{C.9}$$

with ψ a shared local function that operates in isolation on each node of the neighborhood of node i, often a MLP. These 'messages' are then weighted by a fixed scalar c_{ij} and aggregated. Intuitively, the coefficients c_{ij} encode 'how much node i values the message from node j'. The node is updated using the node feature vector $\mathbf{x}_i$ and its local environment by ϕ, another MLP.

This procedure is reminiscent of a discrete convolution kernel in CNNs, where the weights of the kernel are fixed and discretized. As nodes graphs not necessarily have the same degree nor equidistant neighbors, we apply a continuous kernel to weigh the neighborhood nodes. This way, c_{ij} might be a function of node distance, $c_{ij} \sim ||\mathbf{r}_i - \mathbf{r}_j||$.

For further reading on this type of architecture, check Ref. [80].

C.3.2 Graph Attention Networks (GATs)

For GATs, h is defined as follows

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right) \tag{C.10}$$

where, compared to GCNs, the aggregation coefficient c_{ij} is replaced by a *learnable* scalar using an attention mechanism[83]. This attention function (often modeled as a MLP) is trained to take as inputs the two nodes for which the attention coefficient $a_{ij} \equiv a(\mathbf{x}_i, \mathbf{x}_j)$ is to be computed. a_{ij} are normalized among all of the neighborhood using a softmax function to qualitatively indicate how much attention node i should spend on the message from node j. Additionally, this mechanism can be extended to multiple attention heads, where different attention coefficients are computed using different attention functions.

Further information on GATs can be found in Ref. [81].

C.3.3 Message Passing Networks (MPNNs)

MPNNs are the most general type of GNNs that use the message passing mechanism, first applied to Quantum Chemistry[82]. h is defined as follows

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j\in\mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right) \tag{C.11}$$

The idea is that, for each node i, messages of nodes $j \in \mathcal{N}_i$ are prepared through ψ, usually a MLP. This messages are transformations of the neighboring node's features taking into account also node i. The message preparation provides a general procedure for message-passing GNNs. If the messages are reduced to be a fixed scalar (computed scalar) times an

isolated transformation on neighboring nodes, we obtain GCNs (GATs).

As a last comment, note that MPNNs one message must be computed per edge, compared to the other two varieties where only transformations on nodes were considered and weighted by fixed or trainable scalars. This makes MPNNs the most expensive to compute, as there are many more edges than nodes, and only scale well for small graphs compared to the other two (cheaper) alternatives.

C.3.4 Equivariant MPNNs

In the context of physics and chemistry, MPNNs used to represent molecules for tasks such as molecular energy regression or per-node force predictions, not only must be permutation equivariant (due to the inherent graph representation), but must also satisfy the symmetries of Euclidean space, $E(3)$ (translation, rotation and reflection symmetries). This equivariance could, in principle, be learnt by either using data augmentation and letting the model learn it through *data*, or using an $E(3)$-equivariant model architecture[43,45]. We will focus on the latter.

NNP models contain many information encoded in feature vectors, *inter alia* atom type, partial charge, velocities, and positions. It is desirable that, the latter are transformed in the same way as the molecule is transformed in space, i.e. are $E(3)$-equivariant[43,53,54,77,84]. To achieve this[77], positions $\boldsymbol{x}_i \in \mathbb{R}^3$ are treated independently to other scalar features $\mathbf{f}_i \in \mathbb{R}^{k-3}$. A GNN layer would transform these feature vectors independently: it would be invariant with respect to feature vectors $\mathbf{f}_i \to \mathbf{f}_i$, but equivariant with respect to positions $\boldsymbol{x}_i \to \mathcal{R}\boldsymbol{x}_i + \mathcal{T}$, with $\mathcal{R}$ an orthogonal matrix describing rotations and reflections, and $\mathcal{T}$ a vector describing translations[77].

Note that, due to invariance, the dependency of scalar features $\mathbf{f}_i$ with positions is only on the distance between nodes $||\boldsymbol{x}_i - \boldsymbol{x}_j||^2$. However, there are important features which are not scalars (vector fields such as velocities or forces) which must be transformed under $E(3)$ symmetries[53,54]. The implementation of equivaraint models that can take advantage of higher-rank information are discussed in detail in Ref. [84], and a brief introduction is provided in Appendix E.

C.4 Prediction tasks for GNNs

Once the output graph has been obtained (resulting latent representation), the architecture must allow us to extract the information we need. There are three main types of predictions. **Node classification**, in which, we extract per-node information based on the resulting embedding of each node, such as the forces applied to each atom or the partial charges of each atom in a molecule. **Graph classification**, in which we extract per-graph (global) information, for example the total molecular energy or total charge. Finally, we can do **link prediction**, where the task is to predict the existance and properties of edges.

$$\text{Node classification:} \qquad \mathbf{z}_i = f(\mathbf{h}_i) \tag{C.12}$$

$$\text{Graph classification:} \qquad \mathbf{z} = f\left(\bigoplus_{i\in\mathcal{V}} \mathbf{h}_i\right) \tag{C.13}$$

$$\text{Link prediction:} \qquad \mathbf{z}_{ij} = f(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij}) \tag{C.14}$$

with $\mathbf{z}_i$ (or $\mathbf{z}_{ij}$ the outputs of the model (predictions) and $\mathbf{h}_i$ the output graph latent feature vectors.

C.5 Particular considerations for molecular systems

GNNs have great applicability for molecular systems, as molecules can be naturally represented as graphs[1,82,85]. In the context of NNPs, we want models to be able to infer the molecular energy and forces without prior knowledge of the bonds, as is done in ES methods.

The most recent types of NNPs, such as MACE or Orbital models, are based on GNNs[e] and differ substantially with respect to NN-based NNPs. All these models use a cutoff function $f_c(r)$ to limit the short-range interactions of the model and exploit locality.

GNN NNPs define nodes as atoms, but edges are not identified with bonds (that would require prior knowledge of the type of bonds of the molecule). Instead, a node is connected to another atom if the distance is such that $||\mathbf{r}_i - \mathbf{r}_j|| \leq r_c$, with r_c the cutoff distance. The molecular energy is determined via the local contributions of each atom,

$$E_{\text{total}} = \sum_{i \in \mathcal{V}} E_i \ , \tag{C.15}$$

and as such, requires a per-node prediction: once the output node is obtained, energies are extracted from the latent features in each node. Forces follow the same per-node prediction procedure.

Graphs are particularly useful for computational chemistry as they are well suited to problems where the interaction between entities is important. Short range interaction between atoms can be modeled with high accuracy using the message-passing procedure, which provides a given atom information of its chemical environment and hence its interactions within the molecule.

C.5.1 Long-range interaction description

In principle, knowing that after ℓ GNN layers a given node obtains information about its ℓ-hop neighbors, we could naïvely stack enough layers to ensure every node receives information from every other node (with ℓ equal the graph *diameter*). This deep GNN architecture would, in principle, encode information from far away nodes, and thus describe long-range interactions. However, deep GNN architectures run into issues, namely **oversmoothing** and **oversquashing**[86,87].

Most problems, such as molecular energy prediction, must to account for interaction between not directly connected nodes, which must be achieved (in the message-passing framework) by stacking multiple GNN layers. Different learning problems require different ranges of interaction between nodes in the graph to be described properly, which is known as the *problem radius* r. To account for this radius, the GNN architecture should contain $\ell \geq r$ layers, otherwise the model will suffer from **under-reaching**: distant nodes will not be aware of each other. However, as the number of layers increases, the number of nodes in each node's *receptive field* grows exponentially, $\mathcal{N}_i \sim \mathcal{O}(\exp \ell)$. This requires information from all these

[e] Another relevant model in molecular modeling based on GNNs is AlphaFold[1].

nodes to be compressed into a fixed size feature vector $\mathbf{h}_i$, which creates a bottleneck known as **over-squashing**. In this situation, the graph fails to effectively propagate messages from far away nodes, and nodes ultimately only learn short-range messages from the training data.

Deep GNN architectures also affect short-range problems. As the number of layers increases, node representations become indistinguishable, converging to a meaningless value. This is called **over-smoothing** and is mainly caused by using more layers than the problem required: if only short-range information is required, more layers generate inputs from far away nodes, irrelevant for the problem.

In the context of Quantum Chemistry, the locality approximation and short-range nature of the models might have great implications for chemical systems where long-range electrostatic interactions are of uttermost importance for the correct description of the systems. This is an active area of research (see Refs. [88, 89] for an overview). For GNNs, over-squashing is an inherent limitation and should be mitigated if long-range interactions are to be correctly modeled.

C.5.2 Hybrid ML/MM models

Hybrid models in the context of Quantum Chemistry refer to models that use two distinct levels of theory for different regions, and are particularly useful in systems where only a system subset is of importance. This is the case of QM/MM approaches in modeling enzymatic reactions, where only the active site where the reaction takes place is described at QM level, and the rest of the protein is treated at MM level.

Hybrid ML/MM can show great applicability in theoretical studies involving modeling of large systems (such as enzymatic reactions) (see Refs. [90–92] for an introduction), metals (Refs. [25, 92, 93]) and other complex systems where parameters are unavailable or where a higher level of accuracy is needed but performing QM calculations is prohibitively expensive.

D Atomic environment representation

Given the coordinates $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ and identities $\mathbf{Z} = (Z_1, \ldots, Z_N)$ (usually the atomic number of each atom) of each component in an N-particle system, MLIPs aim to predict the energy of the configuration[6,38,50]. In this task, a ML model is trained on high-accuracy quantum mechanical results to predict energies of molecules it has not been trained on (regression task)[51]. The incentive of using an ML model is to 'skip' all the expensive quantum mechanical calculations of ES methods, while providing comparable accuracy.

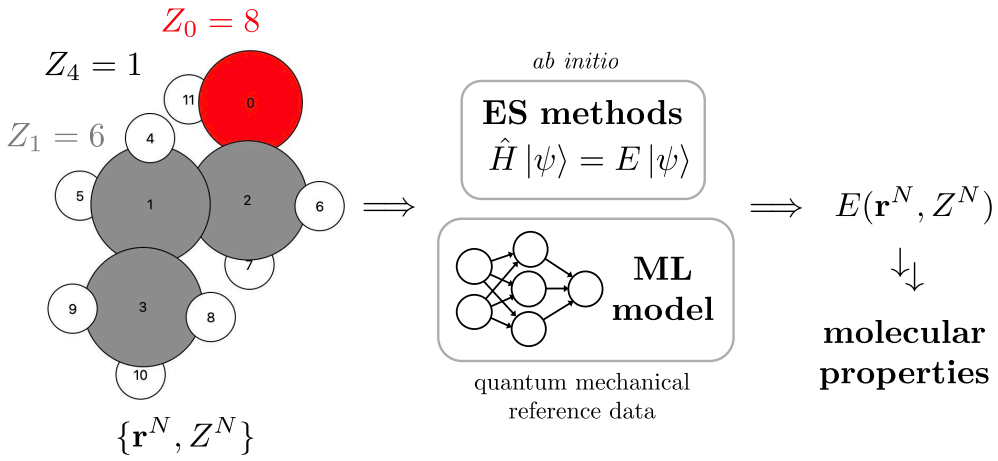


Figure D.1: An atomic arrangement of N atoms can be uniquely determined by a set of atomic coordinates, $\mathbf{r}^N$, and their 'identity', given by the atomic number Z_i. This information is parsed to ES methods, which rely on standard well-known approximations to solve the TISE and ultimately obtain the energy of the configuration. On the other hand, MLIPs aim to use the same input together with a ML model trained on reference QM results to predict a molecular energy, from which predictions on molecular properties can be performed. The model effectively skips the QM calculations, 'learning the fundamental relations of QM, without learning the postulates'. This skip, however, is not completely physics-independent, the model still must be *physically adequate* and respect the symmetries of the system[6].

However, the task is deceptively simple compared to the design of a suitable model. This section aims to introduce the core aspects that must be considered regarding basic symmetry and physical constraints to produce a physically adequate model able generalize well on a wide variety of atomic arrangements.

D.1 Physical constraints and symmetries

ML models differ substantially to traditional physics-based models in that the approach is purely data-driven, and no physical equations or priors are given to it (although one notable exception is PINNs[47,48,94]). As such, basic physics such as conservation laws or symmetries must be 'taught' to the model through data or through inductive biases in the model architecture. Conservation laws, typically derived from temporal invariance, provide strong constraints that can be used as guiding principles in search of physically adequate and plausible ML models[6]. Implications of conservation laws are as follows

- **Energy conservation** imposes restrictions on forces, namely, that forces must be the gradient of a potential. ML models can opt to predict forces as the gradient of the potential energy prediction, or alternatively, predict forces directly. In the later case, however, as they are not computed from a potential, it is difficult to obtain a model with energy conservation.

- **Linear and angular momentum conservation** imposes a dependence of the potential energy on the *relative* positions of the atoms, thus translating into rotational and translational invariance of the potential energy and rotational and translational equivariance of forces.
- Lastly, a subtile symmetry of the system is **permutation invariance**[6,51]: the potential energy must be symmetric to the exchange of two identical atoms (of the same element). Although trivial, it will have consequences when designing a model.

Failure to obey basic physics might result in models breaking symmetry, predicting a different energy for identical but rotated molecules, or breaking conservation laws, generating spurious forces under no external field[6]. In both of these cases, the consequences would prove catastrophic for *in silico* simulations and all the properties derived from them.

D.2 Special considerations of NNPs

The use of a ML model leads to practical considerations that define how NNPs are implemented. To introduce them, consider a model formed by a multi-layer perceptron (MLP), such as that of Fig. 2a. This model accepts a *fixed length*, *ordered*, input, the coordinates of the atoms $\mathbf{r}$ and outputs the potential energy prediction, $\hat{E}$. Additionally, we define the *representation* of the model as the description of the atomic arrangement that is parsed as input to the model[6,38,39,51]. In order for this model to be *physically adequate* (satisfy the basic physical constraints and system symmetries), the energy must be rotation-, translation- and permutation-invariant[38,39].

To ensure this, we must obtain a representation that is translation, rotation and permutation-invariant as well as *complete*[38] (must describe a molecule's conformation in a unique way). This would guarantee that, given any symmetry-modified coordinates, the same input is provided to the model and thus the same output (energy) is returned, satisfying invariance. In order to prove useful, the representation and model should be transferable[57]: independent to system size and applicable to very different systems. A cartesian representation of the system, while simple and complete, is not suitable[39,51,55]: the list of coordinates may be arbitrarily ordered, and they are not rotation- and translation-invariant. Another option would be using internal coordinates (distances, angles, dihedrals, etc.), however, they may be arbitrarily ordered and the number of them increase drastically with the body-order, thus requiring a large and expensive model to train and evaluate[39]. In these two cases, the same but rotated molecule may be represented in a substantially different way, which would provide a different input to the model (representation not invariant), and consequently, the energy would not be invariant[6].

Most importantly, however, is the consequence of using a *fixed length* in the input of the model (Fig. 2a). As the input length is inherent to the model (cannot be changed), the applicability of the model is limited to a fixed number of atoms[6,38,50]. To make the model independent of system size, the system energy, E, is decomposed into local (site) contributions[38] as follows

$$E(\mathbf{r}_1, \ldots, \mathbf{r}_N) = \sum_{i=1}^{N} E_i(\mathbf{r}_1, \ldots, \mathbf{r}_N)\,, \tag{D.1}$$

which additionally makes E permutation-invariant with respect to atomic permutations. The problem transforms into a different one: rather than predicting the energy of the whole

molecule (considering all possible interactions), we predict the energy of an atom in the molecular environment. As such, closer atoms will have much more contribution to the local energy than atoms far away. This naturally introduces a *cutoff* on the distance, $f_c(r)$ to determine the locality approximation, i.e. 'how many interactions should be accounted for' at the expense of computational effort[6]. Interactions between atoms separated by more than the cutoff radius are truncated (Fig. D.2). Still, there might be interactions beyond this radius that should be included, like long-range electrostatic interactions[55] (see section C.5.1.)

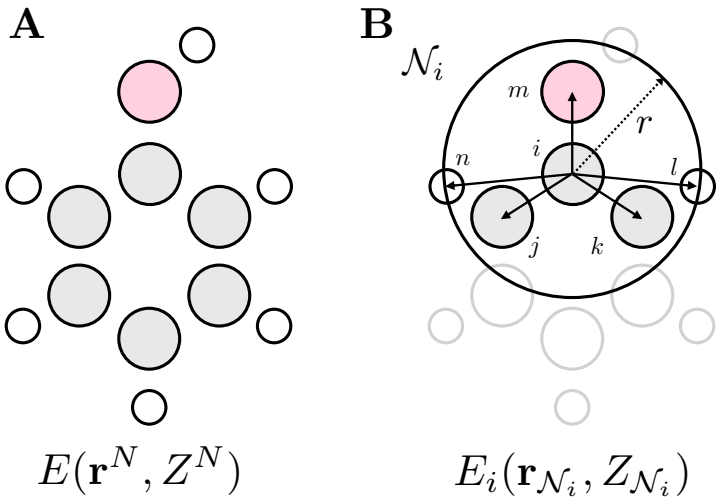


Figure D.2: **(A)** atomic arrangement (positions and identites), of the phenol molecule. Its molecular energy is denoted by E. A severe limitation of using fixed length models, such as a MLPs, is that the number of atoms must remain fixed. To circumvent this issue, the energy is computed as site (local) contributions (see Eq. (D.1)). Each atom i is considered to only interact with its chemical neighborhood $\mathcal{N}_i$, defined by a certain cutoff radius r. The representation of the local environment is used to compute atomic energies E_i **(B)**, which are then aggregated to obtain the total energy E. This locality approximation, however, might have great impact when long-range interactions are present, as they are truncated[55].

D.3 Description of the atomic environment

Once the permutation invariance and locality have been introduced, the next step is to design a suitable *representation* able to *describe* the atomic environment of each atom, and from it, derive the local atomic contributions E_i to the total energy.

The most important part of a NNP is the way the local atomic environment is represented (Fig. D.2). This representation must be translation-, rotation- and permutation-invariant, *complete*, and independent of the number of atoms in the environment, due to fix length limitations. Depending on how this representations is performed, there are two types of models[6,50,55], described in the following sections.

D.3.1 Descriptor-based NNs

In order to obtain the different site contributions, E_i, a first approach is to convert atomic coordinates $\mathbf{r}^N$ into an appropiate representation through the use of *descriptors*. These are many-body invariant symmetry functions that describe the chemical environment in a unique way[6,38,39,51,55], from which a model is able to predict the local energy contributions to the total energy. A general depiction of a descriptor-based model is shown in Fig. D.3.

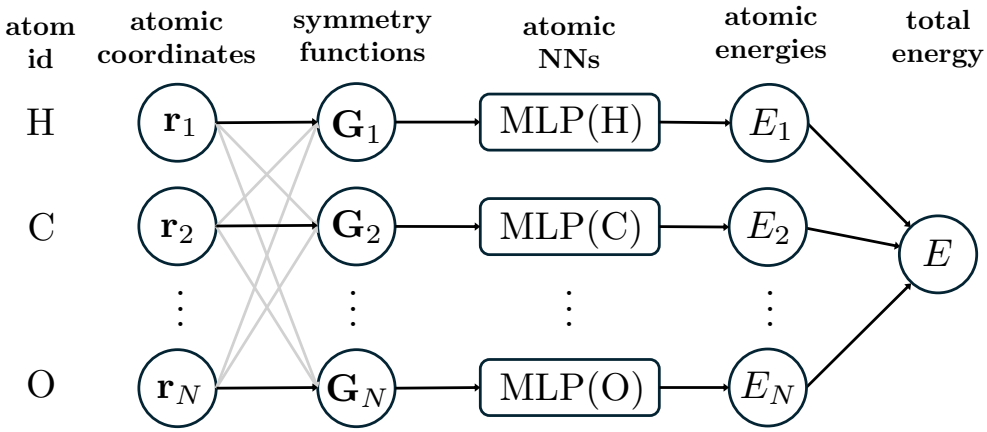


Figure D.3: General view of the descriptor-based NNP workflow. The atomic environment (up to a certain cutoff) every atom is encoded into a fixed set of descriptors, $\mathbf{G}_i$ for atom i, which serve as input to a NN architecture to predict the local atomic energies E_i. The descriptors $\mathbf{G}_i$ are built using the cartesian coordinates $\mathbf{r}$ of the atoms inside the cutoff radius (see Fig. D.2), as indicated by the grey arrows. To improve the model, we might choose to have different MLPs for each atomic element, i.e. one MLP for H, C, O, N, etc. The final molecular energy is determined from the local contributions. Adapted from Ref. [51].

In this sense, we can think of the representation of the atomic environment through a set of functions (descriptors) as a preprocessing step to obtain an invariant model, in which the predicted energy E is invariant to translations, rotations and permutations[39]. The next two sections provide an introduction to two sets of descriptors, atomic centered symmetry functions[38,51,55] (ACSFs), and the idea behind the atomic cluster expansion[40,64] (ACE).

D.3.2 Atomic centered symmetry functions (ACSFs)

The first use of chemical descriptors to obtain an invariant model used atomic centered symmetry functions[38] (ACSFs). In order to describe atomic environments, a fixed number M of symmetry functions is used (Fig. D.3), which describe the 'fingerprint' of each atomic neighborhood i, $\mathbf{G}_i = (G_{i1}, \ldots, G_{iM})$, with G_{ij} the j-th symmetry function corresponding to the environment of atom i. Note that M is independent on the number of atoms inside the cutoff (neighborhood) to ensure that the input to the NN is fixed in length[55].

ACSFs (explained in detail in Refs. [38, 39, 51, 55]) consist of two types of transferable functions to describe the environment: *radial* symmetry functions, which describe the radial distribution of neighbors up to the cutoff radius, and *angular* symmetry functions specifying their angular arrangement. Radial functions depend on the distances (2-body information) to all neighboring atoms, while angular functions depend on all possible angles (3-body information). The dependence on internal coordinates (distances and angles) guarantees translation-, and rotation-invariance, while the aggregation of all internal features guarantee permutation invariance.

In practice, a limited number of radial and angular functions are used to 'examine' the chemical environment, which imposes a limit on its resolution: more symmetry functions provide larger resolution to distinguish similar environments, but higher computational cost, as the NN model becomes larger[51].

ACSFs however, are just one type of descriptors used to represent atomic environments,

and other have been developed[39–41,63], which are summarized in Table 4 of Ref. [50]. A notable transferable NNP model that uses modified ACSF functions is the ANI family of models (ANI-1x[57], ANI-1ccx, ANI-2x[26]).

D.3.3 Atomic cluster expansion (ACE)

Recently, a new representation called atomic cluster expansion (ACE) has been introduced and proven to be a broad generalization of all before-mentioned sets of descriptors[40,64]. The idea behind ACE is to estimate the atomic energy E_i as a many-body expansion

$$E_i = V^{(1)}(\mathbf{r}_i) + \frac{1}{2}\sum_j V^{(2)}(\mathbf{r}_i, \mathbf{r}_j) + \frac{1}{3!}\sum_{j,k} V^{(3)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k) + \frac{1}{4!}\sum_{j,k,l} V^{(4)}(\mathbf{r}_i, \mathbf{r}_j, \mathbf{r}_k, \mathbf{r}_l) + \dots \ , \quad \text{(D.2)}$$

with $V^{(K)}$ the contribution to the atomic energy E_i, which depends on $K-1$ neighbors. This is, at its core, what is used to express classical FFs in Eq. (2.2). Note that, as the body-order increases, the computational cost raises dramatically: there are many more angles (three-body) than distances (two-body), and in turn, many more dihedrals (four-body) than angles. Specific systems, such as metals, may require a many-body expansion of up to $K \sim 15$[40]. ACE is a generalization of ACSFs that generates a symmetry-aware ν-body expansion at low computational cost. In fact, when ACE is expanded up to three-body contributions, it can be shown that it is equivalent to ACSFs[64].

The importance of many-body descriptors is fundamental and has been proven a limiting factor in certain sets of descriptors, as there might exist different chemical environments which can only be distinguished by using higher-order descriptors[65,66], as depicted in Fig. D.4.

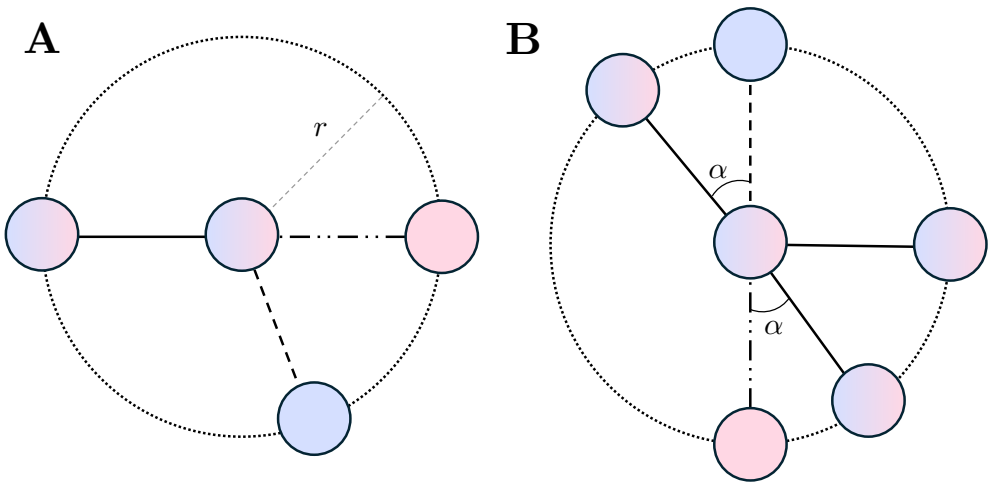


Figure D.4: Different structures comprised of blue or red atoms. Atoms colored with a gradient are shared between both structures. (**A**) Both red and blue molecules are 3 atoms, the red one is linear while the blue one is angular-shaped. If only 2-body correlations are considered to represent the environment of the central atom, i.e. only distances, both structures would be identically described. Higher body-order correlations are needed to distinguish between them, such as angles (3-body order), which would show the linear vs. angular disparity. (**B**) Both red and blue structures are 5 atoms. From the central atom perspective, the environment is described by the same distances and angles (2 and 3-body order correlations). If distances and angles are used to describe the environment of the central atom, the structures are indistinguishable. These two examples **A** and **B** show the importance of the body-order of the interactions by a model, and its implications. Restricting the body-order limits the expessability of the model, ultimately hurting energy prediction. Example **B** is adapted from Refs. [65, 66]

D.3.4 End-to-end NNs

The need to use hand-picked symmetry functions in descriptor-based models to describe a given chemical environment has several downsides, as we (the designers) do not really know what functions will provide the best representation of the atomic environment: many functions can be used (with the invariance constraint), but not all can uniquely describe the environments[6,39].

To skip these decisions, a completely different end-to-end approach can be considered. In it, rather than using hand-picked symmetry functions, we let the model *learn* a suitable invariant representation through data[6]. As such, the model figures out the best invariant representation of the chemical environment for the task, without the need for a hand-crafted functional forms of the descriptors[6,43]. This learnt representation is then used as input to predict site energies, E_i. Fig. D.5 shows a comparison between to two architectures.

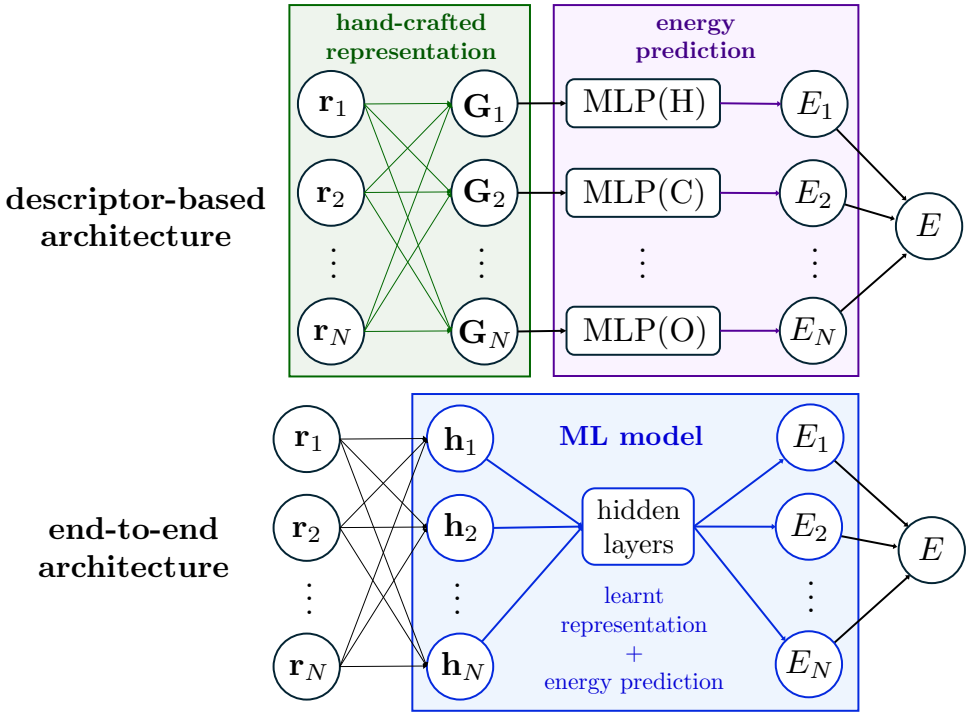


Figure D.5: Visual comparison between descriptor-based and end-to-end models. (Top) Descriptor-based architectures rely on hand-crafted features to convert the initial atomic coordinates $\mathbf{r}^N = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$ into descriptors $\{\mathbf{G}_i\}$, which can then be used as input to the NN-based model to obtain atomic energies. (Bottom) End-to-end architectures take a different approach and the representation is learnt by the model to produce internal representation feature vectors $\{\mathbf{h}_i\}$, which are then internally used by the model to predict the atomic energies, in an input-output closed fashion.

To obtain a learnt invariant representation, the model should ideally use only invariant internal features, such as distances or angles to ensure the invariance of the learnt model. Alternatively, we can lift this restriction and train the model through a data augmentation approach, where the training set is artificially enlarged by performing modifications such as rotations, in the hopes that the model learns a rotation-invariant representation. This last approach, however, besides being computationally costly (increasing training time substantially), it does not guarantee that the resulting representation is, in fact, rotation-invariant.

End-to-end models are usually MPNN[82], a type of GNN (explained in Appendix C) where a molecule is represented as a graph with atoms representing nodes. All atoms within the radial cutoff are connected through edges, representing interactions, and define the atomic environment of each node. In an end-to-end MPNN, each node stores the chemical information of the environment through the learnt representation, and is updated as in the message-passing formalism, using information of neighboring nodes to increase the receptive field to not just the local environment (1-hop neighbors), but beyond. A noteworthy invariant convolutional GNN model is SchNet[42].

D.3.5 Mixed approaches: MACE

A particularly interesting descriptor MPNN is the MACE (multi-ACE) architecture[25,45,67,68]. It structures the molecule as a GNN would, where nodes contain its atomic coordinates $\mathbf{r}_i$, fixed features $\boldsymbol{\theta}_i$ and the atomic environment description through generalized-ACE, $\mathbf{h}_i$. What truly makes it powerful is its ability to combine information of neighboring environments (within the message-passing formalism) to generate very-high order descriptions of the molecule, which provides an efficient way to encode the atomic interactions, and hence provide accurate energy predictions. MACE is, additionally, an *equivariant* model with respect to translations and rotations, which is a generalization of invariant models that allows it to use 3D structured data (vectors) for both training and prediction. For further details on equivariance and how it differs with respect to invariance, refer to Appendix E.

E Equivariant Models. Spherical Tensor Embeddings

Neural network architectures that incorporate and process symmetry information have become recently very popular in the geometric deep learning space[43–45,52–54]. Correctly capturing the symmetry of a system is especially important when dealing with molecules (as is the case for NNPs), as all of their properties must transform in a predictable way under a transformation of the coordinate system through translations, rotations or reflections (Fig. E.1). The inadequate description of the system symmetries can lead to severe consequences such as breaking fundamental physical constraints (e.g. conservation laws), thus resulting in fundamentally flawed models.

This section aims to introduce the importance of using rotation-equivariant models, able to identify features in any rotation, and discuss how equivariant models differ over invariant ones. Additionally, a brief overview on how equivariant models are implemented in the message-passing framework is provided.

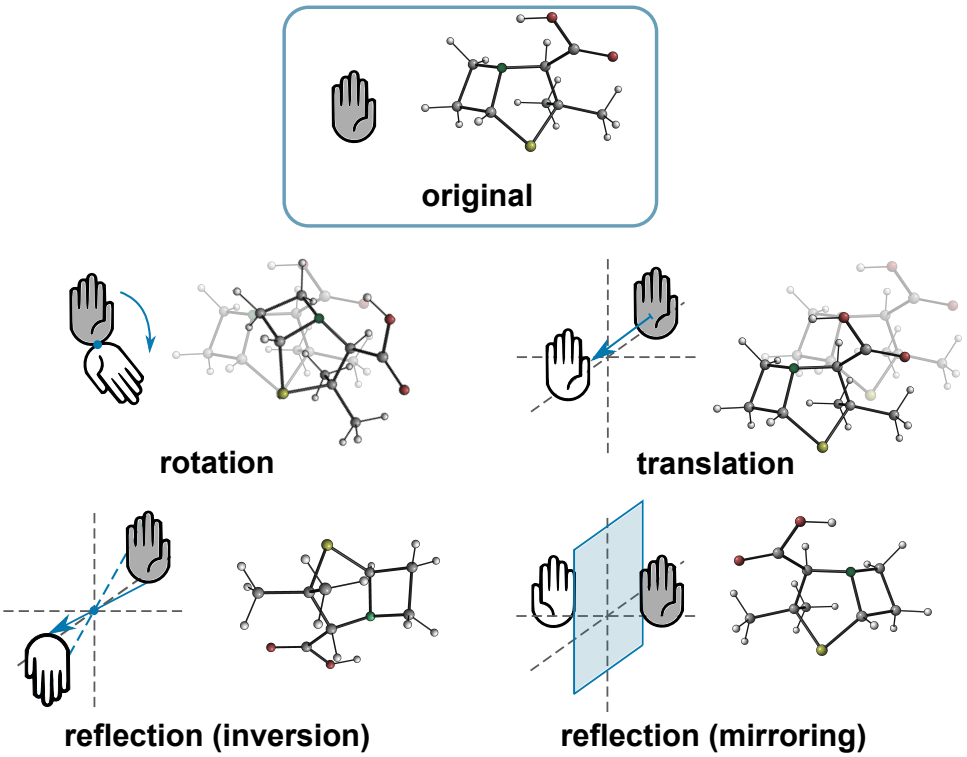


Figure E.1: Symmetry elements of $E(3)$ (Euclidean symmetry group in 3D). Other relevant symmetry subgroups are $SE(3)$ (special Euclidean group in 3D), consisting of only translations and rotations, and $SO(3)$ (special Orthogonal group in 3D), consisting of only rotations. Extracted from Ref. [52]

E.1 The need for equivariant models

Consider an isolated molecule in the absence of any external field. Properties such as the potential energy of a molecular configuration, global charge, and other scalar attributes of the system should be *invariant* with respect to changes in relative orientation in space. Other order properties, e.g. vectorial properties such as dipole moment, forces and velocities of each atom, rather than remaining *invariant* with respect to, e.g. a rotation, should transform according to it: the prediction of the dipole moment vector should change if the atomic

coordinates are rotated, while its scalar value remains invariant with respect to the rotation.

Statements such as 'the potential energy of a molecule is invariant to rotations of the molecule' might appear trivial for physics-based models, where a system is modeled through a series of equations. For these models, the equations are such that basic physical constraints, such as conservation laws, and symmetries are always fundamentally respected. ML models, on the other hand, are not based on any physical priors, meaning that basic physical constraints (e.g. energy conservation) and how symmetry operations affect the system properties must be learnt through data. To circumnavigate this, a different approach has been to introduce these constraints as *inductive biases* in the architecture design[38–45].

The first models were designed such that the molecular energy prediction was *invariant* with respect to translation and rotation (thus a physically adequate model). Such models (descriptor-based[38–41] or GNN-based[42]) are made invariant through the use of scalar internal features (such as distances and angles), which do not change under rotations or translations and ensures the invariance of the output. However, limiting the internal representation with scalars means that they are unable to predict geometry-dependent objects, such as vectors, which change according to the rotation of the atomic coordinates.

Even though invariant models can only use scalar data for training, many molecular attributes are 3D-geometric data (vectors, matrices, etc.). This data, however, is sensitive to transformations of 3D space and as such, only models that 'understand' how they are transformed under rotations can use it[54]. The 'understanding' of symmetries comes from changing their internal representation according to rotations of the atomic coordinates. In this sense, equivariant models are more general than invariant ones, the later being a specific case of the former.

Thus, equivariant models lift the limitation of using scalar-only features by using internal features that transform under symmetry operations of the input, hence 'capturing'/'understanding' the rotation of the system[52]. The rotation-equivariant internal representation allows the model to use and predict higher-order data (e.g. vectors).

There are several benefits when it comes to using *equivariant* models rather than *invariant* ones[52–54]. First, as they are not limited to using scalar information, they are more information-rich and faithful in their predictions. Secondly, the use of scalars, vectors, and higher-rank geometric tensors in the internal representations, allow it to be more expressive in predictions and more efficient capturing features of the data. Additionally, it allows the model to be more data-efficient than invariant models[54], as the mapping to be learnt is restricted by rotation-equivariant functions.

E.2 Spherical tensors in equivariant models

The most important practical choice when implementing an equivariant neural network is to determine how the features can be embedded (expressed) and modified in an equivariant way under rotations. This section aims to provide an overview on the spherical tensor formalism and show how they can be used to create equivariant MPNNs[44,53,54].

Let f be a function such that $f : X \to Y$ with $\mathbf{r} \in X$ and G be a group with $g \in G$

the symmetry elements of that group. $\mathcal{D}(g)$ and $\mathcal{D}'(g)$ are the representations of g acting on X and Y, respectively. The equivariance condition[52] can be written as follows

$$f(\mathcal{D}(g)\mathbf{r}) = \mathcal{D}'(g)[f(\mathbf{r})] \,, \tag{E.1}$$

that is, predicting vector properties such as the dipole moment on the rotated set of atomic coordinates $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_N)$, $\mathcal{D}(g)\mathbf{r}$, is the same as performing the prediction on the unrotated coordinates and then rotating the output of f. Note that the *representation* of the symmetry element can be very different. If g is a rotation, $\mathcal{D}(g)$ will be given by the usual rotation matrix in 3D, $\mathcal{R}$. However, if f returns the Hamiltonian matrix for $\mathbf{r}$, the representation of that exact rotation $\mathcal{D}'(g)$ will look very different and will be represented by a higher-rank geometric tensor. Note that the definition of invariance can be recovered from Eq. (E.1) by setting $\mathcal{D}'(g) \equiv \mathbb{I}$

Even though we are interested in E(3)-equivariance, we will only focus on SO(3) (3D rotations). Translation-equivariance can be trivially obtained using relative atomic displacement vectors[44], $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$.

The general idea to introduce equivariance into a model is to decompose the node feature vector into a set of rotationally equivariant geometric tensors called *spherical tensors*[54]. These are useful objects for dealing with rotation-equivariance due to their properties. In order to understand how higher dimensional spherical tensors transform under rotations, irreducible representations (irreps) of SO(3), a subset of rotation matrices that can be used to construct larger rotation matrices that operate on higher-dimensional tensors, are used. The irreps of SO(3) are the Wigner D matrices. Each type-ℓ Wigner D matrix, $\mathcal{D}^\ell$ with $\ell \geq 0$, defines how a type-ℓ spherical tensor rotates. These matrices provides a direct mapping between any rotation $\mathcal{R}$ of atomic coordinates, $\mathbf{r}$, in 3D space, $\mathcal{R}\mathbf{r}$, and a rotation in the embedded spherical tensor space via $\mathcal{D}^\ell_\mathcal{R} \in \mathbb{R}^{(2\ell+1)\times(2\ell+1)}$, the matrix corresponding to rotation $\mathcal{R}$.

Given a node feature vector, information might be stored as scalars, vectors or higher-rank cartesian tensors. These can be reexpressed spherical tensors, which can be directly rotated using $\mathcal{D}^\ell_\mathcal{R}$ matrices. The basis used for the projection are the (real) spherical harmonics, which represent a complete and orthonormal basis for rotations in SO(3).

Spherical harmonics $Y^m_\ell(\hat{\mathbf{r}})$, in this context, are functions defined on a sphere $Y^m_\ell : S^2 \to \mathbb{R}$ that project 3-dimensional vectors into spherical tensors that transform equivariantly and directly under Wigner-D matrices. Spherical tensors are classified acording to their degree ℓ (type-ℓ spherical tensors). Each type-ℓ spherical tensor transforms under the type-ℓ irreducible representation $\mathcal{D}^\ell$. Given a vector $\mathbf{r} \in \mathbb{R}^3$, we can project that vector into a type-ℓ spheric tensor $\mathbf{Y}^\ell$ via

$$\mathbf{Y}^\ell(\hat{\mathbf{r}}) = \begin{bmatrix} Y^{-\ell}_\ell(\hat{\mathbf{r}}) \\ \vdots \\ Y^0_\ell(\hat{\mathbf{r}}) \\ \vdots \\ Y^\ell_\ell(\hat{\mathbf{r}}) \end{bmatrix} \in \mathbb{R}^{2\ell+1} \,, \tag{E.2}$$

which transforms equivariantly via the corresponding Wigner D matrix

$$\mathbf{Y}^\ell(\mathcal{R}\hat{\mathbf{r}}) = \mathcal{D}^\ell_\mathcal{R}\mathbf{Y}^\ell(\hat{\mathbf{r}}) \,. \tag{E.3}$$

For $\ell = 0$, $\mathcal{D}^0_{\mathcal{R}}$ corresponds to 1, which describes the rotations of scalars (invariant under rotations). For $\ell = 1$, $\mathcal{D}^1_{\mathcal{R}}$ corresponds to the 3D rotation matrix R associated with the rotation $\mathcal{R}$ of vectors (equivariant under rotation). Higher degree Wigner D matrices describe the rotation of higher degree spherical tensors.

Interestingly, as spherical harmonics form a complete orthonormal basis, they are particularly useful to decompose any rotationally symmetric (SO(3)-equivariant) feature on the unit sphere, similarly to a Fourier decomposition of a periodic signal

$$f(\hat{\mathbf{r}}) \sim \sum_{\ell} \sum_{m=-\ell}^{\ell} \hat{f}^m_\ell Y^m_\ell(\hat{\mathbf{r}}) \,. \tag{E.4}$$

This function can be rotated identifying the rotation $\mathcal{R}$ on coordinates with its corresponding Wigner D matrix $\mathcal{D}$,

$$f(\mathcal{R}\hat{\mathbf{r}}) = \mathcal{D}_{\mathcal{R}}[f(\hat{\mathbf{r}})] \,, \tag{E.5}$$

which provides a blueprint to create rotationally equivariant functions. Overall, in order to produce an equivariant model, each operation should be equivariant, i.e.

$$\mathcal{F}(\mathbf{Y}^\ell(\mathcal{R}\hat{\mathbf{r}})) = \mathcal{D}^\ell_{\mathcal{R}}[\mathcal{F}(\mathbf{Y}^\ell(\hat{\mathbf{r}}))] \,, \qquad \forall \mathcal{R} \in SO(3) \,, \tag{E.6}$$

with any function $\mathcal{F}$ on spherical tensors of any degree.

E.3 Tensor product

Once we have a broad understanding of spherical tensors, we are interested in knowing how these tensors can interact among them and how they can be manipulated in an equivariant way. Spherical tensors 'interact' with each other through a tensor product[54] $\mathbf{s}^{\ell_1} \otimes \mathbf{t}^{\ell_2}$, with $\mathbf{s}^{\ell_1}$ and $\mathbf{t}^{\ell_2}$ spherical tensors of type-ℓ_1 and type-ℓ_2, respectively. This tensor product is bilinear and equivariant

$$\mathcal{T}^L_M \equiv \left(\mathbf{s}^{\ell_1} \otimes \mathbf{t}^{\ell_2}\right) \equiv \sum_{|m_1|\leq \ell_1} \sum_{|m_2|\leq \ell_2} \mathcal{C}^{(L,M)}_{(\ell_1,m_1),(\ell_2,m_2)} s^{\ell_1}_{m_1} t^{\ell_2}_{m_2} \,. \tag{E.7}$$

The resulting tensor, however, is not a spherical tensor but can be reduced (decomposed) into a set of spherical tensors of type ranging from $L = |\ell_1 - \ell_2|, \ldots, \ell_1 + \ell_2$, and $|M| \leq L$. This decomposition can be performed using Clebsch-Gordan coefficients, which appear naturally in the coupling of angular momentum in QM. This decomposition allows tensor transformations without breaking equivariance: if we have an L-equivariant function, it allows us to combine information of the two tensors, and retreive only the resulting L-degree tensor, which 'plays well' for the symmetry of the function, such that equivariance is mantained.

As an example[53], consider two spherical tensors of $\ell_1 = 1$ and $\ell_2 = 1$ (spherical harmonics projection of two 3D vectors). The resulting $L = 0$ and $L = 1$ spherical tensor comes from $1 \otimes 1 \to 0$ and $1 \otimes 1 \to 1$ which correspond to the scalar and cross product, respectively,

$$\mathcal{C}^{(L,M)}_{(\ell_1,m_1),(\ell_2,m_2)} \qquad \text{becomes} \qquad \mathcal{C}^{(0,0)}_{(1,i),(1,j)} \sim \delta_{ij} \,, \qquad \mathcal{C}^{(1,i)}_{(1,j),(1,k)} \sim \epsilon_{ijk} \,. \tag{E.8}$$

The tensor product is, thus, a way to multiply spherical tensors and obtain another set of tensors of specific symmetry, labeled by L. A more general expression of Eq. (E.7) exists for

products of k spherical harmonics that use generalized Clebsch-Gordan coefficients.

Finally, it is important to understand the *parametrization* of the tensor product. In equivariant models, features might be described by a set of tensors up to a certain degree $L_{\max}$, for example $\mathbf{s} = \{\mathbf{s}^0, \dots, \mathbf{s}^{L_{\max}}\}$ and $\mathbf{t} = \{\mathbf{t}^0, \dots, \mathbf{t}^{L_{\max}}\}$. The corresponding tensor product of these features, $\mathbf{s} \otimes \mathbf{t}$, must produce tensors of the same order to preserve equivariance i.e. up to degree $L_{\max}$. However, for a given degree of the resulting tensor ℓ, there might be different combinations of $\ell_1 = 0, \dots, L_{\max}$ and $\ell_2 = 0, \dots, L_{\max}$. Each of these *paths* are then parametrized by a given learnable weight, and tensors $L > L_{\max}$ are discarded.

E.4 Equivariant MPNNs

To discuss equivariant message-passing, we will use MACE[45] as architecture of reference and the ideas of Refs. [43–45]. Consider the usual MPNN (explained in Appendix C) where each node i at a layer t is denoted with a tuple $\sigma_i^{(t)} = (\mathbf{r}_i, \boldsymbol{\theta}_i, \mathbf{h}_i^{(t)})$, with $\mathbf{r}_i$ the Cartesian position vector of atom i, $\boldsymbol{\theta}_i$ a set of fixed attributes (such as the atomic number) and $\mathbf{h}_i^{(t)}$ its internal learnable features, updated at each layer t. We denote by $\mathbf{f}_i^{(t)}$ the features of node i (fixed, $\boldsymbol{\theta}_i$ and learnt, $\mathbf{h}_i^{(t)}$). Each message-passing layer consists of three steps.

First, in a message-passing phase, a learnable function $\mathcal{M}_t$ generates a message for each neighbor node $j \in \mathcal{N}_i$, $\mathbf{m}_{ij}^{(t)}$, which are aggregated in a permutationally invariant way

$$\mathbf{m}_i^{(t)} = \bigoplus_{j \in \mathcal{N}_i} \mathbf{m}_{ij}^{(t)} = \bigoplus_{j \in \mathcal{N}_i} \mathcal{M}_t(\sigma_i^{(t)}, \sigma_j^{(t)}) \, . \tag{E.9}$$

Note that these messages $\mathbf{m}_{ij}^{(t)}$ are two-body in nature (depend on two nodes). In a second phase, the aggregated message is used to update the features of the receiving node with a learnable update function $\mathcal{U}_t$

$$\sigma_i^{(t+1)} = (\mathbf{r}_i, \boldsymbol{\theta}_i, \mathbf{h}_i^{(t+1)}) = (\mathbf{r}_i, \boldsymbol{\theta}_i, \mathcal{U}_t(\sigma_i^{(t)}, \mathbf{m}_i^{(t)})) \, . \tag{E.10}$$

After all the message passing layers, the node embeddings at each iteration t are used to predict the local energy contribution of each node via a learnable readout function $\mathcal{R}_t$

$$E_i = \sum_t \mathcal{R}_t(\sigma_i^{(t)}) \, . \tag{E.11}$$

An equivariant model is one where its internal representation is able to transform in an equivariant way with rotations of the atomic coordinates, $\mathbf{r}$,

$$\mathbf{h}_i^{(t)}(\mathcal{R}\mathbf{r}) = \mathcal{D}(\mathcal{R})\mathbf{h}_i^{(t)}(\mathbf{r}) \, . \tag{E.12}$$

To ensure this, symmetry must be preserved throughout the MPNN procedure. The equivariant message-passing architecture is divided into three steps.

1. Node features $\mathbf{h}_i^{(t)}$ (determined by an ACE approach) are first expressed as spherical tensors. Tensors are segregated according to symmetry L, and same-symmetry tensors are stacked into different non-interacting channels k (the number of channels may depend on L).

2. For each neighboring node, a message of the same structure as $\mathbf{h}_i^{(t)}$ is created. This is done by combining the features of the two nodes using a parametrized tensor product.

3. Finally, to update the node, we allow same-symmetry channels k of the message to mix. The resulting tensors become the updated nodes features.

Internal node feature vectors, $\mathbf{h}_i^{(t)}$, can be embedded into different degree spherical tensors that transform in an equivariant through Wigner D matrices as seen in Eq. (E.3) (e.g. scalars and vectors transform under $L = 0$ and $L = 1$ Wigner D matrices, respectively). Thus, $\mathbf{h}_i^{(t)}$ can be decomposed into L-symmetry tensors, of which there may be k of them, stacked into k non-interacting (uncoupled) channels. The $2L + 1$ components of each spherical tensor indexed by the symmetry L and channel k, $\mathbf{h}_{i,kL}^{(t)} \in \mathbb{R}^{2L+1}$, are labeled with M, each transforming like a spherical harmonic of degree L and order M, Y_L^M.

To satisfy equivariance, each spherical tensor $\mathbf{h}_{i,kL}^{(t)}$ must transform according to their symmetry

$$\mathbf{h}_{i,kL}(\mathcal{R}\mathbf{r}) = \mathcal{D}^L(\mathcal{R})\mathbf{h}_{i,kL}(\mathbf{r}) \quad \text{or,} \quad h_{i,kLM}(\mathcal{R}\mathbf{r}) = \sum_{M'=-L}^{L} \mathcal{D}^L_{M'M}(\mathcal{R}) h_{i,kLM'}(\mathbf{r}) \,. \tag{E.13}$$

Messages are generated by combining features of two nodes through the use of a learnt parametrization of the tensor product. As we can combine different L symmetries of each feature in each of the nodes, there may be different combinations of the features that contain the same resulting symmetry. All these combinations are gathered and weighted to produce a final L-symmetry tensor, for each channel k. The details can be found in Refs. [44, 45].

Finally, to update the nodes features, all the k channels of messages of the same symmetry, i.e. $\mathbf{m}_{i,kL}^{(t)}$ (and their components $m_{i,kLM}^{(t)}$), are mixed. This is done with a learnable linear transformation $W_{Lk\tilde{k}}^{(t)}$, a square matrix for each symmetry L, each of dimension $k \times k$, with k the number of channels of that symmetry.

$$h_{i,kLM}^{(t+1)} = U_t(\sigma_i^{(t)}, \mathbf{m}_{i,L}^{(t)}) \equiv \sum_{\tilde{k}} W_{k\tilde{k}L}^{(t)} m_{i,\tilde{k}LM}^{(t)} \tag{E.14}$$

This can be reexpressed using a block diagonal matrix $W^{(t)}$, composed of $W_{Lk\tilde{k}}^{(t)}$ according to the symmetry and number of channels of the messages,

$$h_{i,kLM}^{(t+1)} = W^{(t)} \mathbf{m}_{i,kL}^{(t)} = \begin{bmatrix} W_0^{(t)} & & & 0 \\ & W_1^{(t)} & & \\ & & \ddots & \\ 0 & & & W_L^{(t)} \end{bmatrix} \begin{bmatrix} \mathbf{m}_{i,k0}^{(t)} \\ \mathbf{m}_{i,k1}^{(t)} \\ \vdots \\ \mathbf{m}_{i,kL}^{(t)} \end{bmatrix} \tag{E.15}$$

This framework ensures that, given a rotation of the atomic coordinates $\mathbf{r}$, thanks to the symmetry label segregation of the embedding, the model understands how internal features and messages should be modified, resulting in an equivariant message-passing model.

F Pretrained models and datasets

This section provides general information on the models considered in this TFG, such as the architecture (chemical environment description), training dataset used and general applicability[95]. References and GitHub pages and documentation (if available) are also provided. All models are easily integrated with `ASE` simulation environment[59]. Additional details on the installation for each model are provided in this project's own `GitHub` repository.

It is important to note that in this project, only pretrained models have been used, as in order to generate a NNP, a lot of data for a wide variety of systems (if the model is to be transferable) is needed, which in turn requires a lot of computational time to generate it. Pretrained models are trained once on computationally 'expensive' and diverse datasets with the goal of capturing broad interaction patterns. The usage of these models are two: use pretrained models directly to make predictions on similar systems the model was trained on, or, instead, tweak the model with system-specific additional training data to improve its predictive performance on particular systems, process known as 'fine-tunning'. In this sense, pretrained models are also called 'foundational models', as they serve as a versatile foundation for fine-tunning.

F.1 Training datasets and applicability

`ANI` family of models consist of three models: `ANI-1x`[57], `ANI-1ccx`[58] and `ANI-2x`[26]. The pretrained models have been made available in Python by TorchANI[59] (`GitHub`, `Documentation`). These models are trained using `ANI-1` and `ANI-2` datasets[26,58,60–62], containing small organic molecules, more specifically:

- `ANI-1x`. Trained on 5 million perturbed small and mid-sized organic molecules, containing total energies, forces, dipoles and quadrupoles, among other information at DFT wB97X/6-31G(d) level of theory (see Appendix A). Atomic coverage includes H, C, N, O.
- `ANI-1ccx`. Fine-tuned model based on `ANI-1x`, with additional training data of the higher-level CCSD(T)/CBS calculations (see Appendix A). Same atomic coverage as `ANI-1x`.
- `ANI-2x`. Refined `ANI-1x` model with an expanded training set of 8.9 million perturbed molecules, with special focus on torsions, at DFT wB97X/6-31G(d) level of theory. Dataset properties are the same as in `ANI-1x`, with an expanded coverage to F, S and Cl.

These are invariant descriptor-based models (see Appendix D) that use a modified version of ACSF (see Ref. [38]), which contain a two- and three-body order description of the chemical environment based on distances and angles. Forces are predicted in a conservative way as the analytical gradient of the potential energy with respect to nuclear displacements.

`MACE` family of models[25,45,67,68] consist of two sets of pretrained models, `MACE-MP`[25] and `MACE-OFF`[67], each with several subvariants (`GitHub`, `Documentation`). More specifically:

- `MACE-MP`. A foundational model focused on materials trained on the `MPtrj` dataset[70], with $\sim$ 145 thousand periodic DFT calculations including energies, forces and magnetic moments.

- `MACE-OFF`. Focused on drug-like small organic molecules trained on partly on the `SPICE` dataset[69], containing 1.1 million DFT calculations on small molecules interacting with proteins, with formation and total energies, forces and multipole expansion up to octupoles.

These are equivariant models (see Appendix E) based on an high-order generalized ACE[40] description of the chemical environment, with a MPNN architecture. Forces are predicted the as in `ANI` models.

`ORB` (or `Orbital`) family of models[27] consist of two pretrained models, `ORB-v2` and `ORB-D3-v2` (`GitHub`).

- `ORB-v2`. A foundational model focused on materials trained on `MPtrj` and `Alexandia`, the later dataset consisting of 2.5 million DFT calculations of several materials.
- `ORB-D3-v2`. The same model as `ORB-v2` but trained with a dispersion-corrected D3 version of its datasets.

`ORB` models are not invariant/equivariant, instead they opt to learn rotational invariance through augmenting data during training rather than strictly enforcing it as an inductive bias in the model. This increases prediction speed while using less memory, although at the expense of physical adequacy. Additionally, it predicts forces in a non-conservative way: rather than using the gradient of the potential energy, forces are predicted by the model itself, resulting in a near but not strict conservation of energy, increasing speed at the expense of adequacy.

At the time of writing (April 17th 2024), a new set of `ORB` pretrained models have been made available (`v3` version), which improve on `v2` models and are trained on the `OMat24` dataset[96]. Most importantly, these models can both predict forces and torques directly or as the gradient of the potential, instead of being limited to only the former.

G Additional results

This section aims to list additional results following the same protocols and discussions as in Sec. 4. A more complete and organized set of results for this project is available at the GitHub repository of this project.

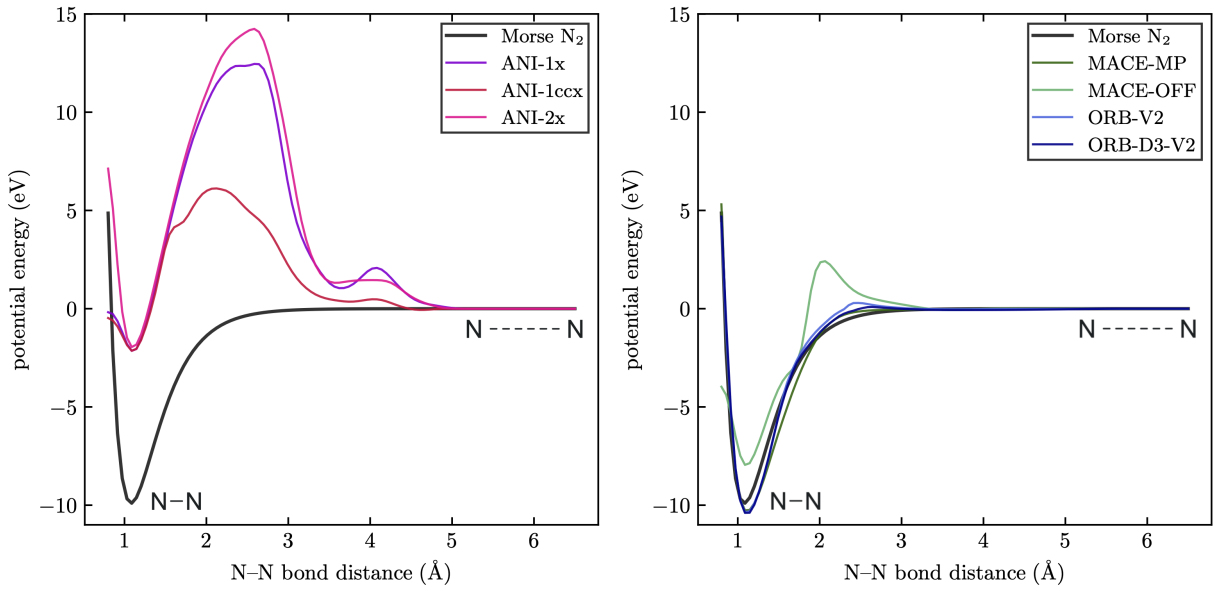


Figure G.1: Dissociation potential for N_2, analogous to Fig. 7. The black line represents the qualitatively correct Morse potential for N_2.

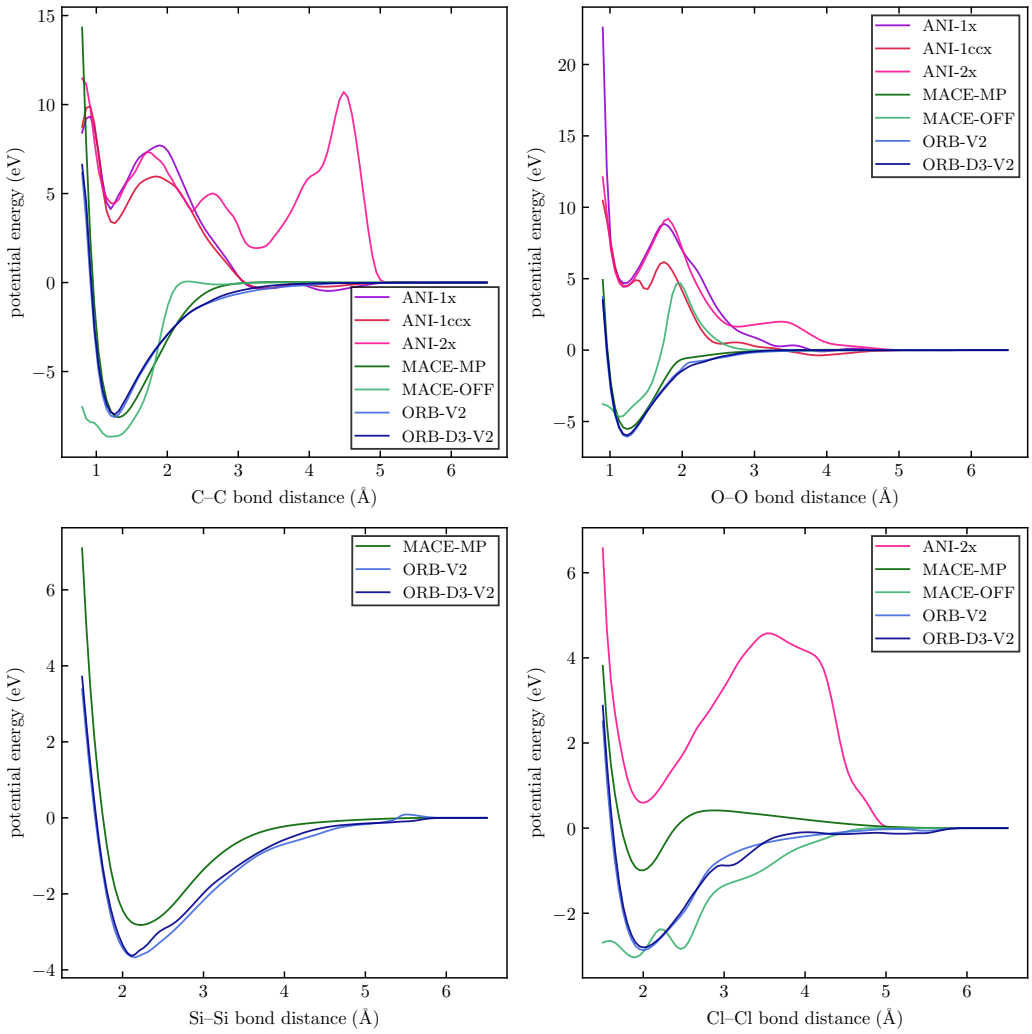


Figure G.2: Additional bond dissociation potentials for C_2, O_2, Si_2 and Cl_2, Analogous to Figs. 7 and G.1. Note that not all models support silicon and chlorine.

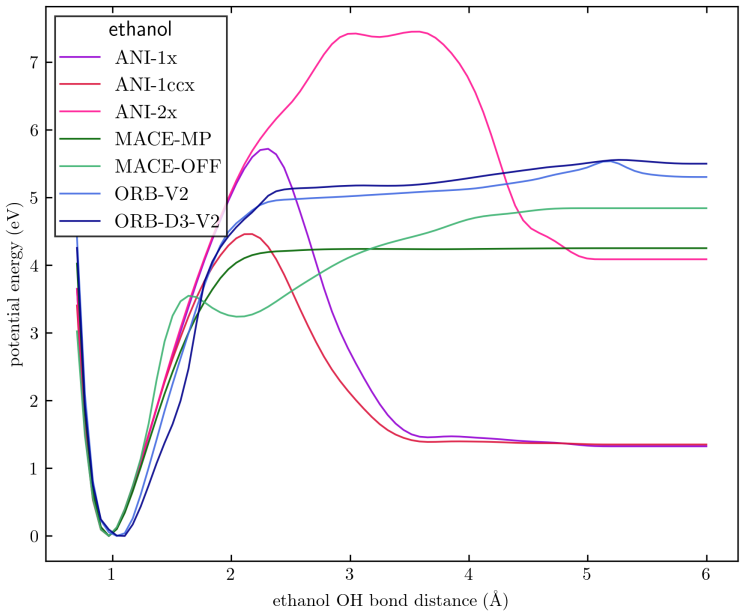
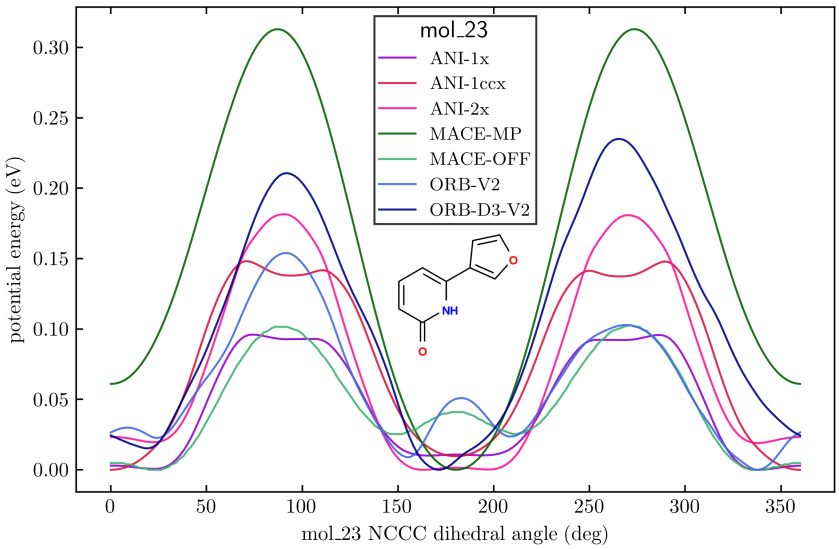
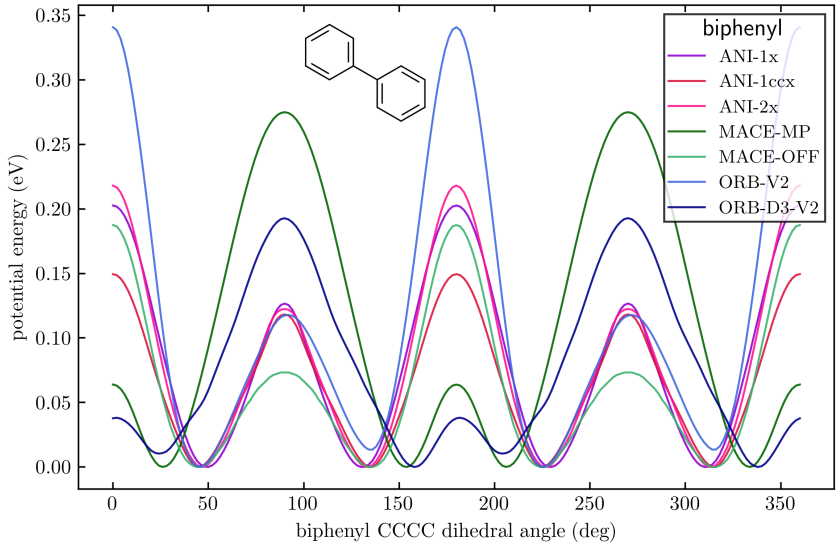


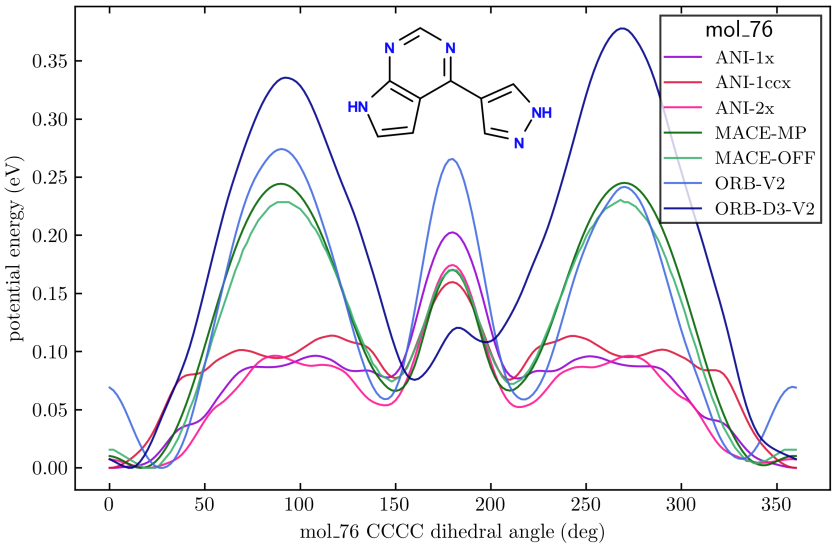
Figure G.3: Ethanol OH hydroxyl group bond dissociation, analogous to Fig. 8.



(a)

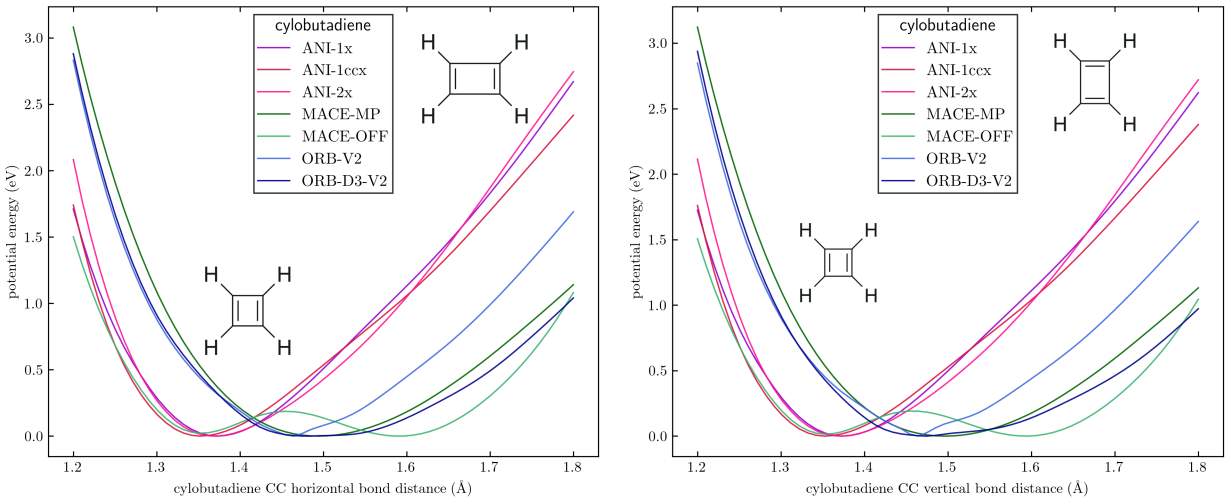


(b)



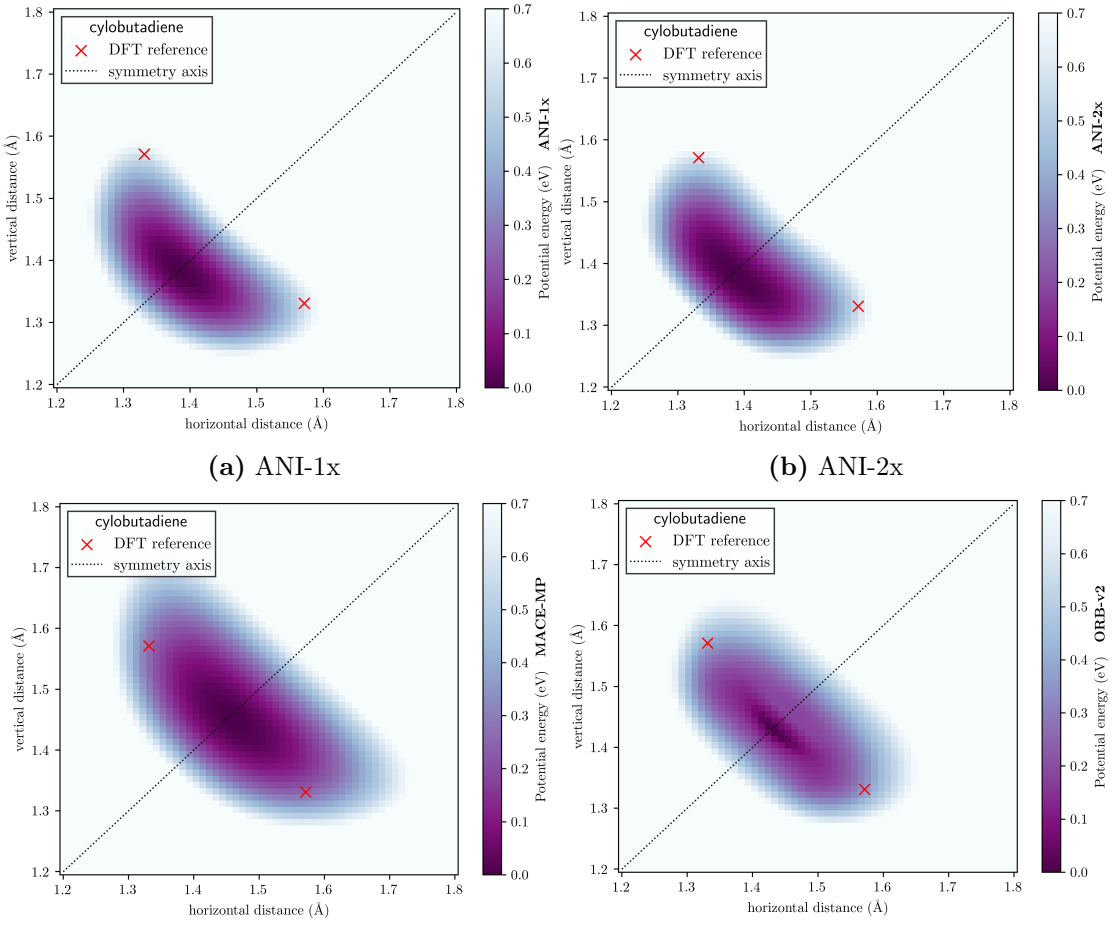
(c)

Figure G.4: Torsional barriers of several byaryl-like compounds. Extension of Fig. 10.



(a) horizontal **(b)** vertical

Figure G.5: Cyclobutadiene **(a)** x-only and **(b)** y-only deformations around the equilibrium distance. Structural depictions are provided to help understand the compression/elongation process. Only `MACE-OFF` provides two minima, accounting for the rectangular symmetry of cyclobutadiene.



(a) ANI-1x **(b)** ANI-2x

(c) MACE-MP **(d)** ORB-v2

Figure G.6: Potential energy surface for the x and y deformations of cyclobutadiene using additional models to those used in Fig. 13. Note that all these models predict a square cyclobutadiene (which differs from reality), but the prediction is qualitatively correct, without any bias toward the elongation of one axis over the other.

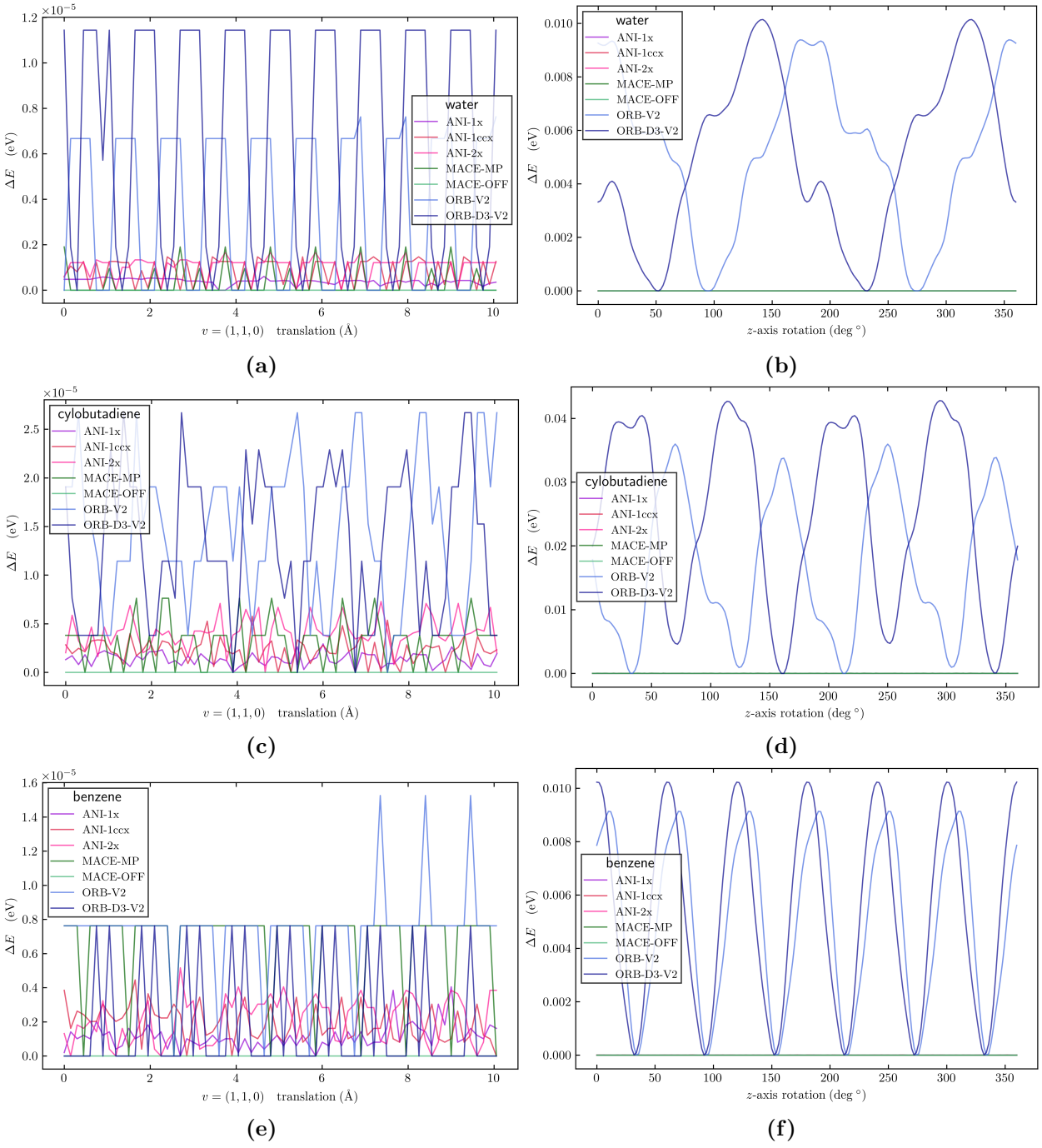


Figure G.7: Energy differences when a water **(a)**–**(b)**, cyclobutadiene **(c)**–**(d)** and benzene **(e)**–**(f)** molecule is rotated and translated. Note that for translations, the scale of the fluctuations is amplified by a 10^5 factor. Variations in translations (**(a)**, **(c)** and **(e)**) are attributed to numerical fluctuations, and translational invariance is observed across all models as the scale of the fluctuations is negligible. For rotations (**(b)**, **(d)** and **(f)**), all except for `Orbital` models are observed to be invariant under rotations. Fig. G.8 contains other test molecules. This is the corresponding graphical representation of the data found in Tables 1 and 2.

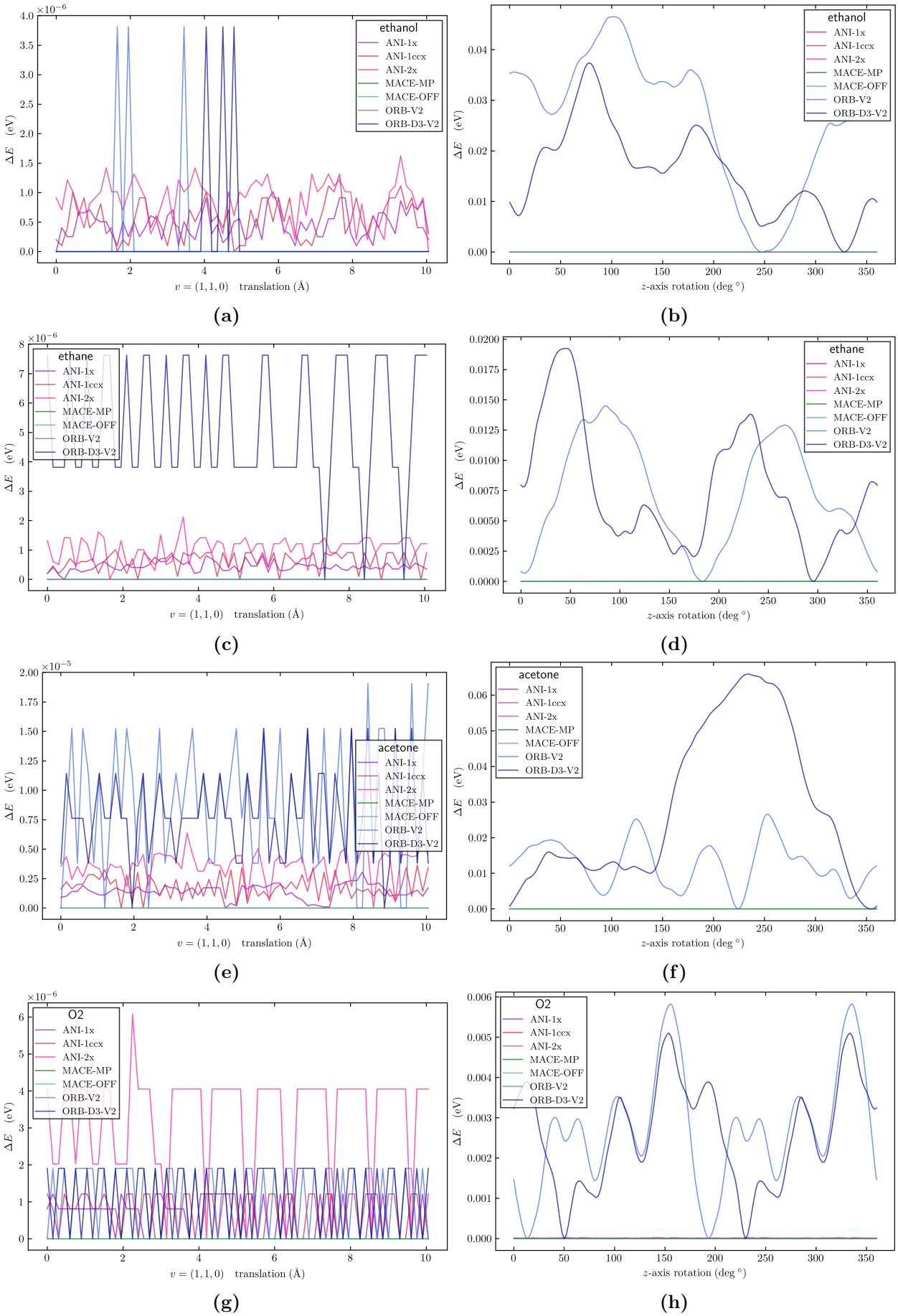


Figure G.8: Continuation of Fig. G.7

Table 7: Maximum force deviation (eV/Å) when predicting the total force for z-axis rotations of three simple molecules. Deviations beyond numerical fluctuations are highlighted in bold. Results for additional systems are showcased in Figs. G.9 and G.10.

System	ANI-1x	ANI-1ccx	ANI-2x	MACE-MP	MACE-OFF	ORB-v2	ORB-D3-v2
water	1×10^{-9}	1×10^{-9}	1×10^{-9}	3×10^{-8}	6×10^{-8}	1×10^{-7}	2×10^{-7}
acetone	5×10^{-7}	7×10^{-7}	8×10^{-7}	5×10^{-7}	5×10^{-7}	8×10^{-7}	1×10^{-6}
benzene	1×10^{-5}	1×10^{-5}	1×10^{-6}	4×10^{-7}	4×10^{-7}	7×10^{-7}	9×10^{-7}

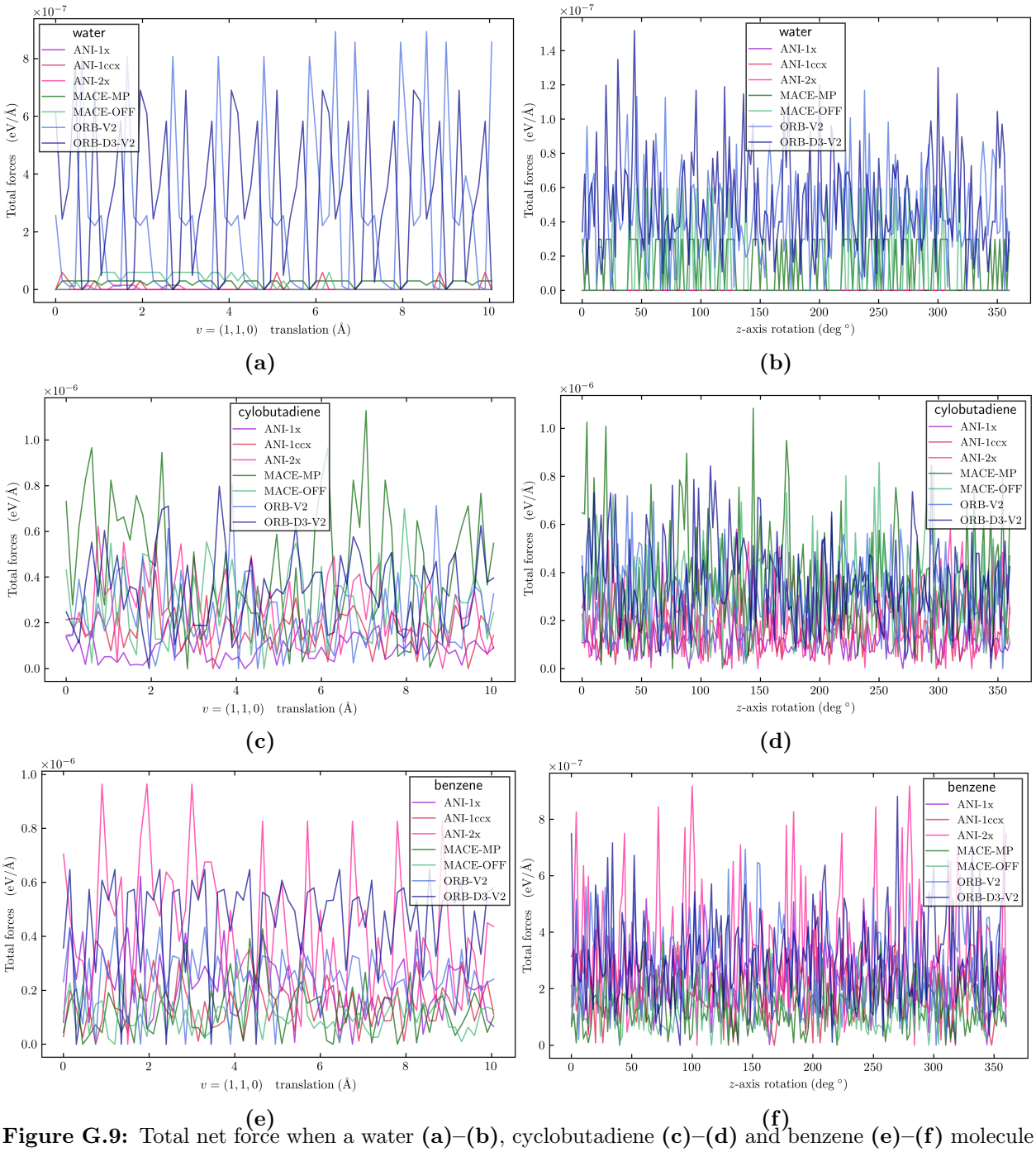


Figure G.9: Total net force when a water **(a)**–**(b)**, cyclobutadiene **(c)**–**(d)** and benzene **(e)**–**(f)** molecule is rotated and translated. Note that for translations and rotations, the scales have been augmented by a factor larger than 10^5. The fluctuations in the net force are attributed to numerical fluctuations. As such, all models satisfy the conservation of linear momentum, even though not because of the same reasons (see results discussion). Fig. G.10 contains other test molecules. This is the corresponding graphical representation of the data found in Tables 3 and 7.

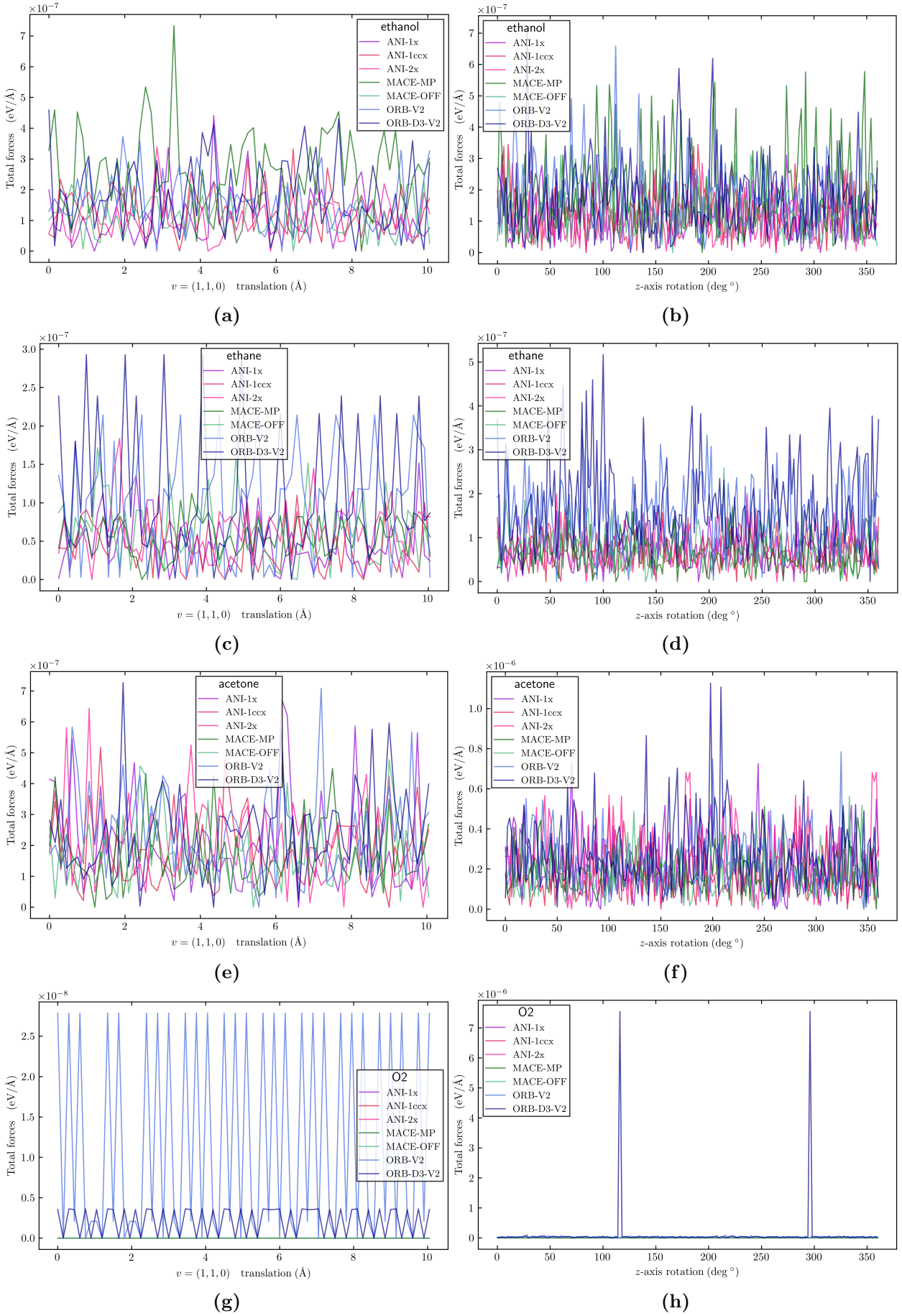


Figure G.10: Continuation of Fig. G.9.

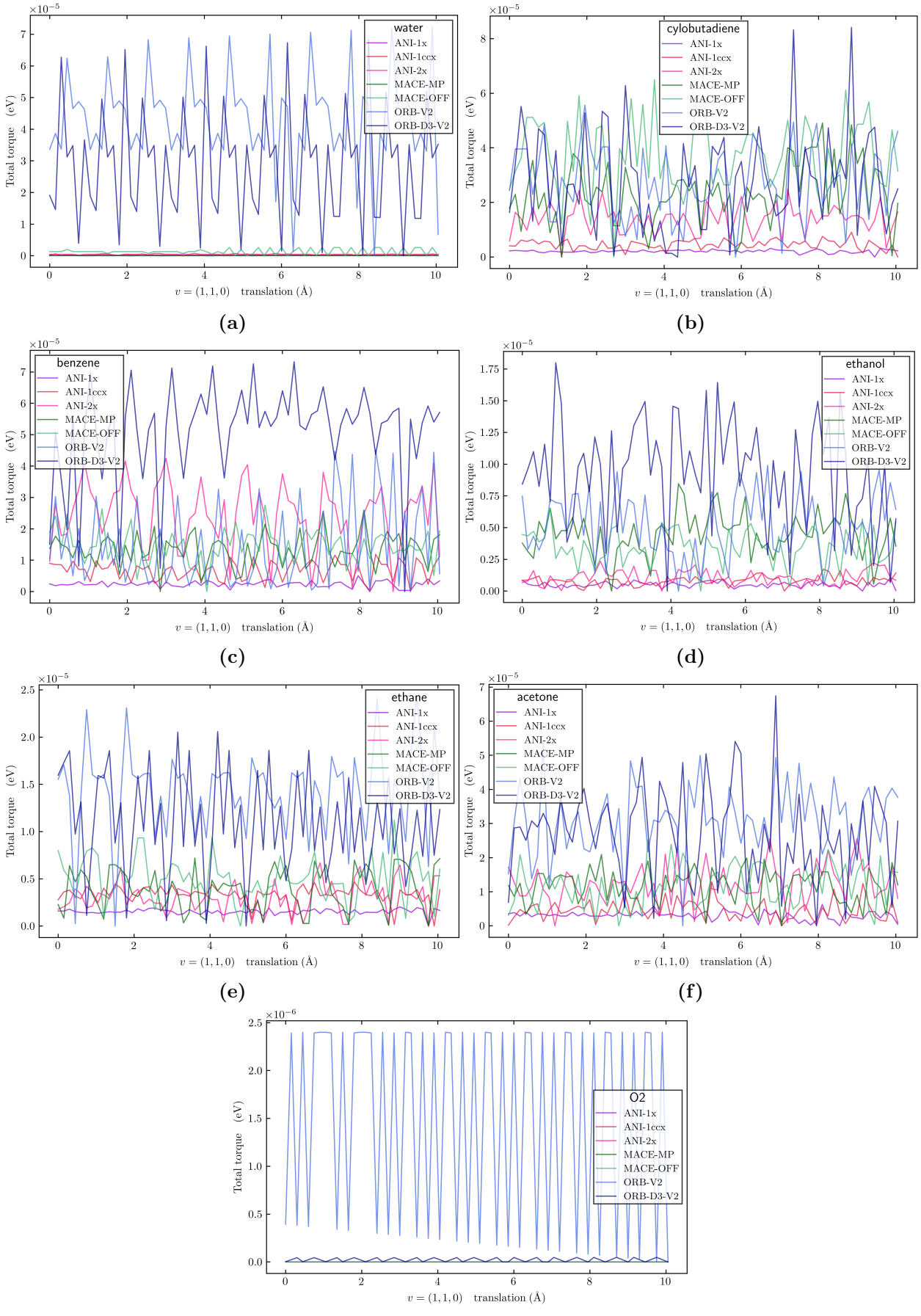


Figure G.11: Total net torque for a variety of compounds along translations, analogously to Fig. G.9 with the total net force. This is the corresponding graphical representation of the date in Table 4.

References

[1] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A. J. Ballard, J. Bambrick, S. W. Bodenstein, D. A. Evans, C.-C. Hung, M. O'Neill, D. Reiman, K. Tunyasuvunakool, Z. Wu, A. Žemgulytė, E. Arvaniti, C. Beattie, O. Bertolli, A. Bridgland, A. Cherepanov, M. Congreve, A. I. Cowen-Rivers, A. Cowie, M. Figurnov, F. B. Fuchs, H. Gladman, R. Jain, Y. A. Khan, C. M. R. Low, K. Perlin, A. Potapenko, P. Savy, S. Singh, A. Stecula, A. Thillaisundaram, C. Tong, S. Yakneen, E. D. Zhong, M. Zielinski, A. Žídek, V. Bapst, P. Kohli, M. Jaderberg, D. Hassabis, and J. M. Jumper, Nature **630**, 493 (2024).

[2] M. Ragoza, J. Hochuli, E. Idrobo, J. Sunseri, and D. R. Koes, Journal of Chemical Information and Modeling **57**, PMID: 28368587, 942 (2017).

[3] M. Popova, O. Isayev, and A. Tropsha, Science Advances **4**, eaap7885 (2018).

[4] J. Benavides-Hernández and F. Dumeignil, ACS Catalysis **14**, 11749 (2024).

[5] E. Schrödinger, Phys. Rev. **28**, 1049 (1926).

[6] O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, Chemical Reviews **121**, PMID: 33705118, 10142 (2021).

[7] J. Hermann, Z. Schätzle, and F. Noé, Nature Chemistry **12**, 891 (2020).

[8] G. Carleo and M. Troyer, Science **355**, 602 (2017).

[9] F. Sabanés Zariquiey, R. Galvelis, E. Gallicchio, J. D. Chodera, T. E. Markland, and G. De Fabritiis, Journal of Chemical Information and Modeling **64**, 1481 (2024).

[10] P. A. M. Dirac and R. H. Fowler, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character **123**, 714 (1929).

[11] A. Szabo and N. S. Ostlund, Dover Books on Chemistry (Dover Publications, Mineola, NY, Jan. 1996).

[12] F. Jensen, 3rd ed. (John Wiley & Sons, Nashville, TN, Feb. 2017).

[13] C. J. Cramer, 2nd ed. (John Wiley & Sons, Chichester, England, Sept. 2004).

[14] A. R. Leach, 2nd ed. (Prentice-Hall, London, England, Jan. 2001).

[15] J. E. Jones and S. Chapman, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character **106**, 463 (1924).

[16] F. Vitalini, A. S. J. S. Mey, F. Noé, and B. G. Keller, The Journal of Chemical Physics **142**, 084101 (2015).

[17] M. J. S. Dewar, E. G. Zoebisch, E. F. Healy, and J. J. P. Stewart, Journal of the American Chemical Society **107**, 3902 (1985).

[18] M. J. S. Dewar and W. Thiel, Journal of the American Chemical Society **99**, 4899 (1977).

[19] J. J. P. Stewart, Journal of Molecular Modeling **15**, 765 (2009).

[20] M. Elstner, Theoretical Chemistry Accounts **116**, 316 (2006).

[21] J. W. Ponder, C. Wu, P. Ren, V. S. Pande, J. D. Chodera, M. J. Schnieders, I. Haque, D. L. Mobley, D. S. Lambrecht, R. A. J. DiStasio, M. Head-Gordon, G. N. I. Clark, M. E. Johnson, and T. Head-Gordon, The Journal of Physical Chemistry B **114**, 2549 (2010).

[22] C. Tian, K. Kasavajhala, K. A. A. Belfon, L. Raguette, H. Huang, A. N. Migues, J. Bickel, Y. Wang, J. Pincay, Q. Wu, and C. Simmerling, Journal of Chemical Theory and Computation **16**, 528 (2020).

[23] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmüller, and A. D. MacKerell, Nature Methods **14**, 71 (2017).

[24] M. J. Robertson, J. Tirado-Rives, and W. L. Jorgensen, Journal of Chemical Theory and Computation **11**, 3499 (2015).

[25] I. Batatia, P. Benner, Y. Chiang, A. M. Elena, D. P. Kovács, J. Riebesell, X. R. Advincula, M. Asta, M. Avaylon, W. J. Baldwin, F. Berger, N. Bernstein, A. Bhowmik, S. M. Blau, V. Cărare, J. P. Darby, S. De, F. Della Pia, V. L. Deringer, R. Elijošius, Z. El-Machachi, F. Falcioni, E. Fako, A. C. Ferrari, A. Genreith-Schriever, J. George, R. E. A. Goodall, C. P. Grey, P. Grigorev, S. Han, W. Handley, H. H. Heenen, K. Hermansson, C. Holm, J. Jaafar, S. Hofmann, K. S. Jakob, H. Jung, V. Kapil, A. D. Kaplan, N. Karimitari, J. R. Kermode, N. Kroupa, J. Kullgren, M. C. Kuner, D. Kuryla, G. Liepuoniute, J. T. Margraf, I.-B. Magdău, A. Michaelides, J. H. Moore, A. A. Naik, S. P. Niblett, S. W. Norwood, N. O'Neill, C. Ortner, K. A. Persson, K. Reuter, A. S. Rosen, L. L. Schaaf, C. Schran, B. X. Shi, E. Sivonxay, T. K. Stenczel, V. Svahn, C. Sutton, T. D. Swinburne, J. Tilly, C. van der Oord, E. Varga-Umbrich, T. Vegge, M. Vondrák, Y. Wang, W. C. Witt, F. Zills, and G. Csányi, 2024, arXiv:2401.00096.

[26] C. Devereux, J. S. Smith, K. K. Huddleston, K. Barros, R. Zubatyuk, O. Isayev, and A. E. Roitberg, Journal of Chemical Theory and Computation **16**, 4192 (2020).

[27] M. Neumann, J. Gin, B. Rhodes, S. Bennett, Z. Li, H. Choubisa, A. Hussey, and J. Godwin, 2024, arXiv:2410.22570.

[28] Y. Shi, Z. Xia, J. Zhang, R. Best, C. Wu, J. W. Ponder, and P. Ren, Journal of Chemical Theory and Computation **9**, PMID: 24163642, 4046 (2013).

[29] Y. Bengio, Adaptive Computation and Machine Learning series (MIT Press, London, England, Nov. 2016).

[30] P. Baldi (Cambridge University Press, Cambridge, England, July 2021).

[31] M. Erdmann, J. Glombitza, G. Kasieczka, and U. Klemradt (World Scientific Publishing, Singapore, Singapore, June 2021).

[32] Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).

[33] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).

[34] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, in *Neural networks: tricks of the trade: second edition*, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012), pp. 9–48.

[35] C. M. Bishop (Oxford University Press, Nov. 1995).

[36] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, Distill, 10.23915/distill.00033 (2021).

[37] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, Cell **180**, 688 (2020).

[38] J. Behler and M. Parrinello, Phys. Rev. Lett. **98**, 146401 (2007).

[39] A. P. Bartók, R. Kondor, and G. Csányi, Phys. Rev. B **87**, 184115 (2013).

[40] R. Drautz, Phys. Rev. B **99**, 014104 (2019).

[41] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, Phys. Rev. Lett. **104**, 136403 (2010).

[42] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller, in Proceedings of the 31st international conference on neural information processing systems (2017), pp. 992–1002.

[43] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, Nature Communications **13**, 2453 (2022).

[44] I. Batatia, S. Batzner, D. P. Kovács, A. Musaelian, G. N. C. Simm, R. Drautz, C. Ortner, B. Kozinsky, and G. Csányi, 2022, arXiv:2205.06643.

[45] I. Batatia, D. P. Kovács, G. N. C. Simm, C. Ortner, and G. Csányi, 2022, arXiv:2206.07697.

[46] B. Moseley, https://github.com/benmoseley/harmonic-oscillator-pinn.

[47] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Nature Reviews Physics **3**, 422 (2021).

[48] M. Raissi, P. Perdikaris, and G. Karniadakis, Journal of Computational Physics **378**, 686 (2019).

[49] G. P. P. Pun, R. Batra, R. Ramprasad, and Y. Mishin, Nature Communications **10**, 2339 (2019).

[50] J. A. Keith, V. Vassilev-Galindo, B. Cheng, S. Chmiela, M. Gastegger, K.-R. Müller, and A. Tkatchenko, Chemical Reviews **121**, PMID: 34232033, 9816 (2021).

[51] J. Behler, The Journal of Chemical Physics **134**, 074106 (2011).

[52] K. Atz, F. Grisoni, and G. Schneider, Nature Machine Intelligence **3**, 1023 (2021).

[53] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, 2018, arXiv:1802.08219 [cs.LG].

[54] M. Geiger and T. Smidt, 2022, arXiv:2207.09453 [cs.LG].

[55] J. Behler, International Journal of Quantum Chemistry **115**, 1032 (2015).

[56] L. Himanen, M. O. Jäger, E. V. Morooka, F. Federici Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke, and A. S. Foster, Computer Physics Communications **247**, 106949 (2020).

[57] J. S. Smith, O. Isayev, and A. E. Roitberg, Chem. Sci. **8**, 3192 (2017).

[58] J. S. Smith, B. T. Nebgen, R. Zubatyuk, N. Lubbers, C. Devereux, K. Barros, S. Tretiak, O. Isayev, and A. E. Roitberg, Nature Communications **10**, 2903 (2019).

[59] X. Gao, F. Ramezanghorbani, O. Isayev, J. S. Smith, and A. E. Roitberg, Journal of Chemical Information and Modeling **60**, PMID: 32568524, 3408 (2020).

[60] R. Zubatyuk, J. S. Smith, J. Leszczynski, and O. Isayev, Science Advances **5**, eaav6490 (2019).

[61] J. S. Smith, B. Nebgen, N. Lubbers, O. Isayev, and A. E. Roitberg, The Journal of Chemical Physics **148**, 241733 (2018).

[62] J. S. Smith, R. Zubatyuk, B. Nebgen, N. Lubbers, K. Barros, A. E. Roitberg, O. Isayev, and S. Tretiak, Scientific Data **7**, 134 (2020).

[63] M. Gastegger, L. Schwiedrzik, M. Bittermann, F. Berzsenyi, and P. Marquetand, The Journal of Chemical Physics **148**, 241709 (2018).

[64] D. P. Kovács, C. v. d. Oord, J. Kucera, A. E. A. Allen, D. J. Cole, C. Ortner, and G. Csányi, Journal of Chemical Theory and Computation **17**, PMID: 34735161, 7696 (2021).

[65] S. N. Pozdnyakov, M. J. Willatt, A. P. Bartók, C. Ortner, G. Csányi, and M. Ceriotti, Phys. Rev. Lett. **125**, 166001 (2020).

[66] Y. Zhang, J. Xia, and B. Jiang, Phys. Rev. Lett. **127**, 156002 (2021).

[67] D. P. Kovács, J. H. Moore, N. J. Browning, I. Batatia, J. T. Horton, Y. Pu, V. Kapil, W. C. Witt, I.-B. Magdău, D. J. Cole, and G. Csányi, 2023, arXiv:2312.15211.

[68] D. P. Kovács, I. Batatia, E. S. Arany, and G. Csányi, The Journal of Chemical Physics **159**, 044118 (2023).

[69] H. Moore, D. P. Kovacs, N. J. Browning, I. Batatia, J. T. Horton, V. Kapil, W. Witt, I. Magdau, D. Cole, and G. Csanyi, 10.17863/CAM.107498 (2024).

[70] B. Deng, P. Zhong, K. Jun, J. Riebesell, K. Han, C. J. Bartel, and G. Ceder, (2023), arXiv:2302.14231.

[71] S. Grimme, A. Hansen, J. G. Brandenburg, and C. Bannwarth, Chemical Reviews **116**, 5105 (2016).

[72] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, Journal of Physics: Condensed Matter **29**, 273002 (2017).

[73] S.-L. J. Lahey, T. N. Thien Phuc, and C. N. Rowley, Journal of Chemical Information and Modeling **60**, 6258 (2020).

[74] D. C. Malaspina, C. Viñas, F. Teixidor, and J. Faraudo, Angewandte Chemie International Edition **59**, 3088 (2020), eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/anie.201913257.

[75] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, G. A. Petersson, H. Nakatsuji, X. Li, M. Caricato, A. V. Marenich, J. Bloino, B. G. Janesko, R. Gomperts, B. Mennucci, H. P. Hratchian, J. V. Ortiz, A. F. Izmaylov, J. L. Sonnenberg, D. Williams-Young, F. Ding, F. Lipparini, F. Egidi, J. Goings, B. Peng, A. Petrone, T. Henderson, D. Ranasinghe, V. G. Zakrzewski, J. Gao, N. Rega, G. Zheng, W. Liang, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, K. Throssell, J. A. Montgomery Jr., J. E. Peralta, F. Ogliaro, M. J. Bearpark, J. J. Heyd, E. N. Brothers, K. N. Kudin, V. N. Staroverov, T. A. Keith, R. Kobayashi, J. Normand, K. Raghavachari, A. P. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, J. M. Millam, M. Klene, C. Adamo, R. Cammi, J. W. Ochterski, R. L. Martin, K. Morokuma, O. Farkas, J. B. Foresman, and D. J. Fox, Gaussian Inc. Wallingford CT, 2016.

[76] M. Born and R. Oppenheimer, Annalen der Physik **389**, 457 (1927).

[77] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, 2021, arXiv:2104.13478 [cs.LG].

[78] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, AI Open **1**, 57 (2020).

[79] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, IEEE Transactions on Neural Networks **20**, 61 (2009).

[80] T. N. Kipf and M. Welling, 2017, arXiv:1609.02907 [cs.LG].

[81] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, 2017, arXiv:1710.10903.

[82] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, in Proceedings of the 34th international conference on machine learning - volume 70 (2017), pp. 1263–1272.

[83] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, 2023, arXiv:1706.03762 [cs.CL].

[84] V. G. Satorras, E. Hoogeboom, and M. Welling, 2022, arXiv:2102.09844 [cs.LG].

[85] C. W. Coley, W. Jin, L. Rogers, T. F. Jamison, T. S. Jaakkola, W. H. Green, R. Barzilay, and K. F. Jensen, Chemical Science **10**, 370 (2019).

[86] T. K. Rusch, M. M. Bronstein, and S. Mishra, 2023, arXiv:2303.10993 [cs.LG].

[87] U. Alon and E. Yahav, 2021, arXiv:2006.05205 [cs.LG].

[88] T. Jaffrelot Inizan, T. Plé, O. Adjoua, P. Ren, H. Gökcan, O. Isayev, L. Lagardère, and J.-P. Piquemal, Chem. Sci. **14**, 5438 (2023).

[89] D. M. Anstine and O. Isayev, The Journal of Physical Chemistry A **127**, PMID: 36802360, 2417 (2023).

[90] S.-L. J. Lahey and C. N. Rowley, Chem. Sci. **11**, 2362 (2020).

[91] R. Galvelis, A. Varela-Rial, S. Doerr, R. Fino, P. Eastman, T. E. Markland, J. D. Chodera, and G. De Fabritiis, Journal of Chemical Information and Modeling **63**, PMID: 37694852, 5701 (2023).

[92] K. Zinovjev, L. Hedges, R. Montagud Andreu, C. Woods, I. Tuñón, and M. W. van der Kamp, Journal of Chemical Theory and Computation **20**, PMID: 38804055, 4514 (2024).

[93] K. Zinovjev, Journal of Chemical Theory and Computation **19**, PMID: 36821513, 1888 (2023), eprint: https://doi.org/10.1021/acs.jctc.2c00914.

[94] F. Musil, A. Grisafi, A. P. Bartók, C. Ortner, G. Csányi, and M. Ceriotti, Chemical Reviews **121**, PMID: 34310133, 9759 (2021).

[95] M. Kulichenko, B. Nebgen, N. Lubbers, J. S. Smith, K. Barros, A. E. A. Allen, A. Habib, E. Shinkle, N. Fedik, Y. W. Li, R. A. Messerly, and S. Tretiak, Chemical Reviews **124**, PMID: 39572011, 13681 (2024).

[96] L. Barroso-Luque, M. Shuaibi, X. Fu, B. M. Wood, M. Dzamba, M. Gao, A. Rizvi, C. L. Zitnick, and Z. W. Ulissi, 2024, arXiv:2410.12771.