Advancing Computational Quantum Chemistry with Machine Learning

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Christian Hölzer
aus
Köln

Bonn, 2025

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

Gutachter / Betreuer:	Prof. Dr. Stefan Grimme
Gutachter:	Prof. Dr. Thomas Bredow
Tag der Promotion:	05.06.2025
Erscheinungsjahr:	2025

Dedicated to the Prüfungsbüro.

Da steh ich nun, ich promovierter Tor,
Bin jedenfalls ein bisschen klüger als wie zuvor!

adapted from **Goethe, Faust I**

Abstract

This thesis focuses on advancing quantum chemistry using machine learning methods. For this purpose, a dataset for the lanthanoid elements is generated, conformer ranking is addressed leveraging novel machine learning architectures and the widely used extended tight-binding model is enhanced with automatic differentiation.
The LnQM dataset, a comprehensive benchmark of 17 269 mono-lanthanoid complexes optimized at PBE0-D4/def2-SVP level, enables systematic evaluations of quantum chemical and machine learning methods across the lanthanoid series. It features geometric, energetic, molecular and electronic properties at ωB97M-V/def2-SVPD level, granting insights into lanthanoid chemistry and highlighting limitations of current atomic charge models.
The ConfRank ansatz improves conformer ranking through pairwise training of state-of-the-art machine learning models. Utilizing the DimeNet++ architecture, the accuracy of relative energy prediction on GMKTN55 conformational subsets is improved by 29 % on average. Moreover, a considerable 100-fold computational speed-up compared to the currently used GFN2-xTB method is achieved using GPU infrastructure.
The dxTB model, a PyTorch implementation of GFN-xTB, demonstrates a novel integration of quantum chemical algorithms into machine learning frameworks. It allows for differentiation of any input parameters to arbitrary order, achieving similar runtimes as the original Fortran implementation, which in turn lacks automatic differentiation. Moreover, parameter optimization can now be conducted using backpropagation, harnessing the extensive existing machine learning infrastructure, opening up possibilities to investigate new functional forms of internal xTB procedures and to develop individual, problem-specific GFN parametrizations.
Together, these contributions chart new directions across different dimensions of computational research, ranging from data science aspects to model development. This thesis conduces to the ongoing advancement of machine learning in the domain of computational quantum chemistry and aims to offer a valuable contribution on the path to improved material sciences, healthcare and beyond.

Contents

CHAPTER 1

Introduction

Along the search for artificial intelligence, the field of machine learning (ML) has evolved since the development of an artificial neuron by Warren McCulloch und Walter Pitts in 1943 [1] and the perceptron by Frank Rosenblatt in 1958 [2]. After a couple of "AI winters" [3] the topic gained accelerated traction in the 2000s when algorithms, compute power and data availability had sufficiently advanced.
The advent of ML in the modern age has led to many improvements in natural science research. From protein structure prediction [4] to particle identification at the Large Hadron Collider [5], ML has been adopted to countless applications and has become indispensable in contemporary research. In chemistry, ML-based methods have enabled rapid predictions of molecular energies, forces, and reaction pathways, drastically reducing computational costs compared to traditional quantum mechanical calculations [6–8].
The field of quantum chemistry (QC) emerged with the postulation of the Schrödinger equation in 1926 [9] and the first calculation of the diatomic hydrogen molecule using quantum mechanics by Heitler and London in 1927 [10]. With the development of density functional theory and the groundbreaking Kohn-Sham equations [11], computational methods gained more and more attention for the calculation of atoms and molecules in the 1990s [12].
Nowadays, computational methods have become an integral part of QC and an indispensable tool for the theoretical description of chemical processes and systems. Computational chemistry offers a unique contribution to research where experimental realizations are difficult due to toxicological hazards, resource constraints, or time limitations. In this respect, the application of computational chemistry in drug discovery[13], catalyst development [14] and material science [15] has significantly transformed research in the respective areas. Since recently, general-purpose methods in the family of force fields [16] and semiempirical methods [17, 18] have facilitated calculations of molecular properties. Thereby opening up unprecedented large-scale analyses of massive amounts of chemical data, supporting theoretical researchers and researchers in the lab alike. In turn, this data can be used for parameterizing and benchmarking novel methods or studying chemical space. As a result of the increased data abundance, aspects of data management, augmentation, and extension have made data science increasingly important in the field of QC research. Furthermore, since vast amounts of data are the very context in which ML excels, its methods have become powerful tools for improving and complementing traditional computational approaches.

This thesis is chapterized into sections that explore different levels of ML integration with QC. Starting from an introduction to the theoretical foundations of QC and ML, it presents scientific work ranging from dataset generation to the native embedding of ML into QC algorithms. To provide a comprehensive overview of the theoretical background relevant to the projects presented in this thesis, chapter 2 outlines the foundations of QC and ML. In this context, section 2.1 contains a derivation of the Hartree-Fock energy expression from the time-independent Schrödinger equation. Furthermore, the section addresses the principles of wavefunction theory (WFT) and density functional theory (DFT), leading up to the fundamental considerations of semiempirical quantum mechanical (SQM) methods. Further, in section 2.2 the fundamentals of neural networks, along with essential optimization techniques such as gradient descent and backpropagation are explained. Additionally, the section highlights the importance of graph neural networks in computational QC, as they allow for an intrinsic embedding of molecular structures and their associated information. Based on the core idea behind DFT, the parallels between function approximation in DFT and ML are explored.

Subsequent chapters focus on specific research questions along the integration of ML in the QC domain. The ensuing chapter 3 explores data generation for lanthanoids. Thereby, the LnQM dataset is presented – a novel collection of several thousand mono-lanthanoid complexes. This dataset constitutes the first comprehensive resource covering the entire lanthanoid series, providing electronic, energetic, and geometric trends across the series. As part of this study, the performance of various charge models within the lanthanoid regime is analyzed.

In chapter 4 an application of ML for conformer ranking is featured. For this task accurate ranking of relative energies is essential for the identification of the most favorable conformer. Conformer ranking is of critical importance for drug design and molecular biology, yet existing methods are either computationally expensive or lack accuracy for robust predictions in large-scale applications. The ML-assisted ConfRank approach developed in this research project demonstrates improved ranking accuracy while significantly reducing computational costs compared to currently employed methods.

The following chapter 5 deep dives into the development of dxTB, a framework designed for the integration of ML into the widely used xTB method. Using this new framework, it is possible to compute nuclear gradients of arbitrary order, facilitating the determination of molecular spectra, for instance. Additionally, ML techniques can be applied to optimize individual parameters within the GFN parametrization. Moreover, the framework permits for the native use of xTB as a component in novel ML architectures. In a sense, the dxTB framework allows the reinterpretation of existing QC methods as ML models. This maintains the accuracy and robustness of traditional QC methods, i.e. adherence to fundamental physical principles, while facilitating the adaptability of ML models. This synergistic combination of knowledge enables the use of ML methods without discarding the rich knowledge in physics and chemistry accumulated over centuries of research.

The thesis concludes with a summary and outlook in chapter 6, which highlights the key findings and outlines possible future research avenues.

CHAPTER 2

Theoretical background

This chapter provides an overview of the most important theoretical concepts relevant to this thesis, starting with a thematic contextualization of quantum chemistry in section 2.1. Thereby, special focus is put on wavefunction theory (subsection 2.1.1) and density functional theory (subsection 2.1.2) concluding with a brief introduction of semiempirical quantum mechanical methods (subsection 2.1.3). Subsequently, an introduction to machine learning is given in section 2.2, focusing on the basics such as optimization techniques (subsection 2.2.1) and molecular representations (subsection 2.2.2). A general overview over supervised machine learning methods in the field of quantum chemistry is provided in subsection 2.2.3 including the foundations of neural networks. Due to their considerable importance in the field of computational quantum chemistry, graph neural networks are explored in greater detail in subsection 2.2.4. For more information on the quantum chemical background, see also Szabo et al. (1996) and Jensen (2017) [19, 20]. Except for the introduction into quantum mechanics, atomic units will be used throughout this chapter.

2.1 Quantum Chemistry

Quantum Chemistry (QC) is, as the name suggests, the field in which principles of quantum mechanics are applied to chemistry. Granted that for many macroscopic phenomena quantum mechanics can be neglected, its application becomes crucial when addressing small structures. To understand the different length scales, figure 2.1 provides an overview of the characteristic length scales encountered in natural science disciplines. Note that the schematically indicated research areas may extend depending on the specific subtopics within each field.

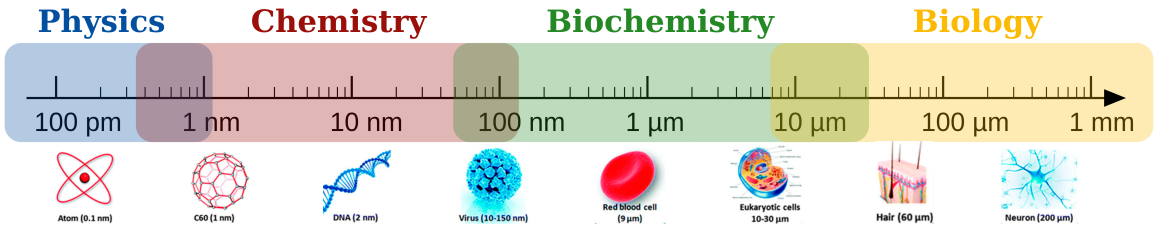

Figure 2.1: Overview over length scales in nature and a schematic indication on the different research domains in natural sciences. Adapted from reference [21].

In the quantum mechanical picture, particles are treated as waves and therefore cannot be described by Newtonian dynamics anymore. Instead of Newton's second law in classical mechanics [22], the Schrödinger equation [9] governs the equation of motion for non-relativistic particles:

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{\mathrm{H}}\Psi(\mathbf{r},t) \tag{2.1}$$

In this equation, i is the imaginary unit, $\hbar$ denotes the reduced Planck constant, and $\frac{\partial}{\partial t}$ represents the partial derivative with respect to time t, capturing the time evolution of the system. The wavefunction $\Psi(\mathbf{r},t)$ describes the quantum state of the system as a function of the position vector $\mathbf{r}$ and time t, while the Hamiltonian operator $\hat{\mathrm{H}}$ encodes the total energy (both kinetic and potential) of the system. The relation for stationary properties, such as the total energy of the system E or discrete energy levels in atoms or molecules, can be derived from the time-independent formulation of the Schrödinger equation:

$$\hat{\mathrm{H}}\Psi(\mathbf{r}) = E\Psi(\mathbf{r}) \tag{2.2}$$

The resulting eigenvectors Ψ_i give rise to different states of the system, whereby the eigenvalues E_i determine the energy of each state. In order to ensure particle number conservation, the wavefunction of a particle needs to be normalized $\langle\Psi|\Psi\rangle = 1$. Furthermore, as $\hat{\mathrm{H}}$ is Hermitian, its non-degenerate eigenfunctions are orthogonal. The orthonormality condition on the wavefunction can be formulated in Dirac notation [23] as follows:

$$\langle\Psi_i|\Psi_j\rangle = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{else} \end{cases} \tag{2.3}$$

The energy of a given eigenstate (or a superposition of eigenstates) can be obtained as the expectation value of the Hamiltonian

$$\langle E\rangle = \langle\Psi|\hat{\mathrm{H}}|\Psi\rangle = \int_{-\infty}^{+\infty}\Psi^*\hat{\mathrm{H}}\Psi\, d\mathbf{r} \tag{2.4}$$

For a specific eigenstates the corresponding energy is retrieved from the main diagonal of the Hamiltonian matrix

$$E_i = \langle\Psi_i|\hat{\mathrm{H}}|\Psi_i\rangle = E_i\langle\Psi_i|\Psi_i\rangle = \mathrm{H}_{ii} \tag{2.5}$$

For any Hermitian operator with a discrete spectrum, such as $\hat{\mathrm{H}}$, the variational principle holds, resulting in the ground state wavefunction Ψ_0 minimizing the energy to E_0, while any other admissible wavefunction $\tilde{\Psi}$ produces an expectation value exceeding $E_0 \leq \tilde{E}$:

$$\langle\tilde{\Psi}|\hat{\mathrm{H}}|\tilde{\Psi}\rangle = \sum_{\lambda_1,\lambda_2}\langle\tilde{\Psi}|\Psi_{\lambda_1}\rangle\langle\Psi_{\lambda_1}|\hat{\mathrm{H}}|\Psi_{\lambda_2}\rangle\langle\Psi_{\lambda_2}|\tilde{\Psi}\rangle \tag{2.6}$$

$$= \sum_{\lambda}\lambda|\langle\Psi_\lambda|\tilde{\Psi}\rangle|^2 \geq \sum_{\lambda}E_0|\langle\Psi_\lambda|\tilde{\Psi}\rangle|^2 = E_0\langle\tilde{\Psi}|\tilde{\Psi}\rangle \tag{2.7}$$

Note that for an accurate description of relativistic systems the Dirac equation [24] is required.

However, in many organic chemistry applications that typically involve only light elements, relativistic effects can generally be neglected. This is because electrons in these systems are subject to only moderate electric fields and accelerations, and thus rarely approach speeds near that of light. In contrast, for heavier elements relativistic corrections become significant [25]. To address these effects, several approaches are available, for example using effective core potentials (ECPs) [26]. ECPs replace inner (core) electrons with an effective pseudopotential to reduce computational effort while efficiently incorporating relativistic effects by treating only the valence electrons explicitly. Relativistic effects become particularly relevant in systems containing heavy elements, as demonstrated by examples such as liquid mercury [27], the color of caesium [28], and lead-acid batteries [29].
For a single-electron system, such as an isolated hydrogen atom, the Schrödinger equation can be solved analytically. In this case, the wavefunction corresponding to the quantum numbers n, l, and m is given by

$$\Psi_{nlm}(r, \theta, \phi) = R_{nl}(r)\, Y_{lm}(\theta, \phi) \tag{2.8}$$

where $R_{nl}(r)$ represents the radial component using generalized Laguerre polynomials, and $Y_{lm}(\theta, \phi)$ the angular component expressed by spherical harmonics. The associated energy levels E_n are determined by

$$E_n = -\frac{\mu c^2 \alpha^2}{2n^2} \approx \frac{13.6\,\text{eV}}{n^2} \tag{2.9}$$

with $\mu = \frac{m_\text{H} m_\text{e}}{m_\text{H} + m_\text{e}} \approx m_\text{e}$ being the reduced mass, c the speed of light, and $\alpha \approx \frac{1}{137}$ the fine-structure constant. In this formulation, corrections due to fine structure, the Lamb shift, and hyperfine structure are neglected [30–34]. This analytical solution allows for simplified treatment of multiple systems such as alkali metals and ions [35].

However, for many-particle systems, analytical solutions are impossible. Even though the underlying physics and mathematical equations are exactly known, the emerging complexity for many-particle system is prohibitively high and by far exceeds the expected growth in computational power in the upcoming decades (c.f. Moore's law [36]). Therefore, in chemistry, where molecules often contain hundreds or even thousands of electrons, approximation methods are needed. The development of suitable approximations that allow to calculate solutions for the many-particle Schrödinger equation within finite computational time is *the* fundamental challenge in QC. In a nutshell, QC applies quantum mechanical approximations to model the complex many-particle interactions problem present in molecular systems.

2.1.1 Wavefunction Theory

Wavefunction theory (WFT) comprises methods in QC that explicitly compute a wavefunction by approximating the Schrödinger equation. Thereby, WFT offers a detailed treatment of quantum mechanical effects and typically yields highly accurate results. However, this accuracy comes at the cost of substantial computational demand, limiting the scalability of pure quantum mechanical approaches. To overcome these limitations, electronic structure theory provides strategies for efficiently calculating molecular geometries, energy gradients, and other physical properties in larger chemical systems. For this purpose, the following section will first discuss how the Hamiltonian for general molecular systems is constructed and simplified using the Born-Oppenheimer approximation. It will

subsequently explain how one-electron orbitals are build from basis functions and combined into a many-electron wavefunction via a Slater determinant.

Electronic Hamiltonian

Starting from the time-independent Schrödinger equation 2.2 the wavefunction and corresponding energy of a molecule can be determined using a Hamiltonian constructed from kinetic $\hat{\mathrm{T}}$ and potential energy terms $\hat{\mathrm{V}}$ for electrons and nuclei (subscripts e and n):

$$\hat{\mathrm{H}} = \hat{\mathrm{T}}_e + \hat{\mathrm{T}}_n + \hat{\mathrm{V}}_{ee} + \hat{\mathrm{V}}_{ne} + \hat{\mathrm{V}}_{nn} \tag{2.10}$$

In the Born-Oppenheimer approximation [37], the significant mass difference between nuclei and electrons is exploited. Since electrons change momentum on much shorter timescale than the heavy nuclei, the equations of motion for the fast (electronic) degrees of freedom can be solved independently from those of the slow (nuclear) ones. In the adiabatic limit, the kinetic energy of the nuclei, $\hat{\mathrm{T}}_n$, can be neglected while the potential between the nuclei becomes a constant, $\hat{\mathrm{V}}_{nn} = E_{nn} = \text{const.}$, simplifying Equation 2.10 to the electronic Hamiltonian, which depends only on nuclear positions but not their momenta:

$$\hat{\mathrm{H}}_e = \hat{\mathrm{T}}_e + \hat{\mathrm{V}}_{ee} + \hat{\mathrm{V}}_{ne} + E_{nn} \tag{2.11}$$

For a N-electron system the electronic kinetic energy operator is given by

$$\hat{\mathrm{T}}_e = -\frac{1}{2}\sum_{i}^{N} \nabla_i^2 \tag{2.12}$$

with ∇_i being the partial differential with respect to the coordinates of electron i. The electronic interaction between the particles is given by the Coulomb potentials

$$\hat{\mathrm{V}}_{ee} = \sum_{i}^{N}\sum_{\substack{j \\ i>j}}^{N} \frac{1}{|\vec{r}_i - \vec{r}_j|} = \frac{1}{2}\sum_{i\neq j}\frac{1}{r_{ij}} \tag{2.13}$$

$$\hat{\mathrm{V}}_{ne} = -\sum_{i}^{N}\sum_{\mathrm{k}}^{K} \frac{Z_{\mathrm{k}}}{|\vec{r}_i - \vec{R}_{\mathrm{k}}|} \tag{2.14}$$

whereby the repulsive electron potential $\hat{\mathrm{V}}_{ee}$ is constituted of the pairwise interaction given the electron coordinates $\vec{r}_{\{i,j\}}$ and the resulting distance $r_{ij} = |\vec{r}_i - \vec{r}_j|$ between electrons i and j. The attractive nucleus-electron potential $\hat{\mathrm{V}}_{ne}$ for the K nuclei in the system and their coordinates $\vec{R}_{\mathrm{k}}$ and nuclear charge Z_{k} is composed respectively. For practicality reasons, grouping all terms into one- and two-electron operators yields

$$\hat{\mathrm{h}} = \hat{\mathrm{T}}_e + \hat{\mathrm{V}}_{ne} \quad \text{and} \quad \hat{\mathrm{g}} = \hat{\mathrm{V}}_{ee} \tag{2.15}$$

Having defined the required Hamiltonian, the electronic energy of a molecular system can be obtained by solving the Schrödinger equation as shown in equations 2.2 to 2.5.

Basis Sets

The exact electronic wavefunction $\Psi(\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_N)$ of an N-electron system depends on $3N$ spatial coordinates and is generally too complex to be determined directly. To make the problem tractable, an approximate form of the wavefunction is constructed, typically using an orbital-based ansatz. In this approach, the many-electron wavefunction is built from simpler, one-electron functions called orbitals, which in turn are expressed as linear combinations of fixed basis functions. These orbitals are usually chosen to be centered on atomic nuclei to reflect the localized nature of electrons around atoms. For this reason they are commonly referred to as atomic orbitals (AOs). The full set of the utilized basis functions constitutes the basis set, which plays a central role for the accuracy and efficiency of electronic structure methods.

A natural starting point for constructing AOs are the analytical solutions of the hydrogen atom. These functions, such as $1s$, $2p$, etc., describe the spatial distribution of a single electron bound to a single nucleus and are well understood both analytically and physically. These so called Slater-type orbitals (STOs) [38] are solutions to the Schrödinger solution for a hydrogen-like atom (c.f. Equation 2.8)

$$\chi_{\mathrm{STO}}(r, \theta, \phi) = N\, r^{n-1} e^{-\zeta |r|} Y_{lm}(\theta, \phi) \tag{2.16}$$

Slater functions correctly model the cusp behavior at the nucleus and the exponential decay at large distances. However, their use in multi-electron integrals leads to computational challenges due to the complexity of evaluating the necessary multi-center integrals. Hence, despite physical correctness, their high computational cost renders them impractical for numerical integration. In contrast, Gaussian-type orbitals (GTOs) [39] are computationally significantly more efficient. The general form of a GTO in Cartesian coordinates is

$$\chi_{\mathrm{GTO}}(r, \theta, \phi) = N\, r^{2(n-1)-l} e^{-\zeta r^2} Y_{lm}(\theta, \phi) \tag{2.17}$$

where N is a normalization constant, n is the principal quantum number, l the orbital angular momentum quantum number and ζ is the Gaussian exponent. The angular part is given by the spherical harmonics $Y_{lm}(\theta, \phi)$ accordingly [20]. Despite lacking the correct cusp at the nucleus and decaying too rapidly at large distances, GTOs are employed by most modern QC programs to approximate the "more physical" STOs [40, 41]. On the one hand, this is due to the fact that integrals of Gaussian functions possess an analytical closed-form solution, speeding up numerical calculations. Furthermore, the Gaussian product theorem, which states that the product of two GTOs centered on different atoms can be expressed as a finite sum of Gaussians along the interatomic axis, facilitates the calculations further. As a result, multi-center integrals can be reduced to sums over one-center integrals, achieving a speedup of four to five orders of magnitude compared to Slater functions – even though a larger number of basis functions is required in general [42].

To flexibly describe the electronic structure of atoms, a wide variety of basis sets, i.e. sets of basis functions and coefficients, have been developed, each with different advantages and areas of focus. Thereby, as the basis set size increases, the variational principle ensures an improvement in the total energy (Equation 2.6). Consequently, the quality of a calculation can be evaluated by performing computations with progressively larger basis sets [20]. In the limit of infinite number of basis functions, a complete basis set (CBS) is obtained, which in principle represents the exact wavefunction of a system. While such a basis set cannot be used in practice due to computational constraints, it serves as a theoretical benchmark. Many high-level methods aim to systematically approach the CBS limit

by using increasingly larger and more flexible basis sets, allowing for extrapolation techniques to estimate the results one would obtain with a truly complete basis set. On the other hand, a minimal basis set is a type of basis set in which each AO is represented by exactly one basis function. As a result, it employs the smallest possible number of basis functions needed to describe all electrons in the ground state of an atom. For example, one function for the $1s$- and $2s$-orbitals, and three functions for the $2p$-orbitals are used. Due to the limited number of functions, calculations using minimal basis sets are computationally very efficient. Although computationally inexpensive, such minimal basis sets offer limited flexibility and are generally mostly suitable for qualitative or educational purposes. To make basis sets practically useful, they require optimized coefficients to ensure accuracy and robustness. This is typically achieved by optimizing the basis sets to reference data. Again, the variational principle (c.f. Equation 2.6) ensures that any improvement in the basis set results in a lower total energy, ensuring strict convergence from above toward the true ground-state energy. This property enables the systematic optimization of basis set parameters, as improved parameterizations necessarily yield lower energy expectation values.

Typically for the generation of basis sets, multiple GTOs are combined into contracted GTOs to mimic STO-like behavior to compensate for their physical shortcomings [43]. A widely used minimal basis is the STO-nG family [43], where each STO is represented by n primitive Gaussians. For this purpose in STO-3G, three GTOs are fitted to each STO, providing a computationally cheap basis set that is useful for qualitative studies. For improved accuracy, split-valence basis sets have been developed. One such widely used basis is the Pople basis set 6-31G [44], where core orbitals are modeled with six contracted Gaussians, while valence orbitals are split into two parts – one represented by three Gaussians, the other by a single uncontracted Gaussian. This separation into core and valence orbitals increases variational flexibility in the chemically important valence region.

Extensions to a basis set involve the addition of polarization and/or diffuse functions. Polarization functions increase the flexibility of the basis by allowing the electron density to become more asymmetric about the nucleus, which is important for hydrogen bonding involving otherwise isotropic s-orbitals, for instance. For polarization always function of higher angular momentum are used, i.e. p-functions are used to polarize s-orbitals, d for p-orbitals, and so on. Diffuse functions are additional s- and p-type basis functions designed to describe electron density far from the nucleus. They are characterized by small Gaussian exponents, which extend the tail of the AOs and provide the flexibility needed to model long-range interactions. Diffuse functions are especially important for accurately describing anions, dipole moments and can also play a crucial role in modeling intra- and intermolecular bonding. For Pople basis sets polarization functions are indicated after the "G" and diffuse functions by a "+" or "++" such as seen in the 6-31+G(d) basis set, which is a split valence basis set with one set of diffuse sp-functions and a single d-type polarization function on non-hydrogen atoms [20].

An improvement over minimal basis sets is achieved by increasing the number of basis functions per atomic orbital. For instance, so-called "Double-Zeta" (DZ) basis sets use two functions for each orbital, i.e. two s-functions for hydrogen, four s-functions and two sets of p-functions for second row elements [20]. This concept can be extended to Triple-Zeta (TZ) and Quadruple-Zeta (QZ) basis sets accordingly. Well-known examples of making use of zeta-type basis sets include the Ahlrichs def2 basis set family [45], which are widely used due to their computational efficiency and broad elemental coverage. Moreover, correlation-consistent polarized valence n-zeta (cc-pVnZ) basis sets [46] are specifically designed to systematically recover electron correlation energy. They do not impose the same constraints as Pople-style sets (e.g. equal exponents for s- and p-functions), making them more

flexible and accurate for correlated methods. The correlation-consistent design ensures that functions contributing similar amounts to the correlation energy are included at the same function stage. These basis sets can be further augmented with diffuse functions (aug-cc-pVnZ) [47].
For heavier elements, the large number of core electrons requires many basis functions to describe them accurately, even though those core electrons are chemically less important. Without this, the valence orbitals cannot be properly represented due to poor treatment of electron-electron repulsion. Additionally, relativistic effects become significant for heavier atoms (see above). Both issues can be addressed by using ECPs, which replace the core electrons with a simplified potential, allowing only the valence electrons to be treated explicitly [20].
Above described basis sets are focused on molecular calculations. For solid state applications where periodic boundary conditions are required, plane-wave basis sets based on the lattice wave vector $e^{i\vec{k}\cdot\vec{r}}$ can be employed [48].

In molecular systems, where electrons are influenced by multiple nuclei, the localized AOs alone are not sufficient to represent the full electronic structure. To describe electron distributions that extend over an entire molecule, molecular orbitals (MOs) are introduced. In the context of computational chemistry, a MO is a mathematical function that describes the spatial distribution and wave-like behavior of a single electron within a molecule. A common approach to constructing MOs is via a linear combination of atomic orbitals (LCAO), in which each molecular orbital ϕ_i is expressed as a linear combination of AOs χ_μ:

$$\phi_i = \sum_{\mu}^{N_{\text{AO}}} c_{\mu i}\chi_\mu \tag{2.18}$$

The coefficients $c_{\mu i}$ determine the contribution of each basis function to the MO and are subject to optimization in later electronic structure methods. As shown in the next section, the coefficients $C_{\mu i}$ are typically determined variationally in an iterative procedure whilst N_{AO} and χ_μ are pre-defined. The LCAO approach approximates the full multi-electron wavefunction of a molecule by expressing it in terms of MOs, which are constructed from AOs. Since these AOs are defined by the chosen basis set, the quality and design of the basis set has great influence on the quality in the accuracy of the resulting wavefunction approximation. Importantly, this idea precedes the formal introduction of the Hartree-Fock method, which will in the following provide a principled way to determine the optimal set of MOs for a given system.
Note that in addition to basis sets, alternative methods exist to obtain MOs such as grid-based [49] or wavelet-based methods [50].

Basis Set Errors

Before diving into how to obtain a molecular wavefunction from a given basis set using the Hartree-Fock Method, error sources of using basis set representations are briefly addressed. Despite the use of large and flexible basis sets, finite basis sets inevitably introduce two significant sources of error: the basis set superposition error (BSSE) and the basis set incompleteness error (BSIE).
BSSE arises when two interacting fragments in a molecular complex can use each other's basis functions, leading to an artificially low total energy. Especially with large, flexible basis sets, the BSSE becomes apparent when calculating the energy of complexes such as hydrogen-bonded dimers.

In these cases, the basis functions of one monomer contribute additional variational freedom to the other monomer, leading to an artificial lowering of the complex energy and an overestimation of the interaction strength. The conventional method to estimate and correct for BSSE is the CounterPoise (CP) correction [20]. Initially, the complexation energy of the dimer is calculated as

$$\Delta E_{\text{complexation}} = E(AB)^*_{ab} - E(A)_a - E(B)_b \tag{2.19}$$

where $E(AB)^*_{ab}$ represents the energy of the dimer computed with the full combined basis set ab, and $E(A)_a$ and $E(B)_b$ are the energies of the isolated monomers calculated with their respective basis sets. Typically, the geometries of molecules A and B in the complex differ from those of the isolated molecules, therefore the complex's geometry is indicated by an asterisk (*). To correct for BSSE, additional calculations are performed in which each monomer is computed in the presence of the "ghost orbitals" of the other fragment, i.e. the basis functions of the other monomer without its nuclei. The CP correction is then defined as

$$\Delta E_{\text{CP}} = \left[E(A)^*_{ab} - E(A)^*_a\right] + \left[E(B)^*_{ab} - E(B)^*_b\right] \tag{2.20}$$

and the corrected complexation energy is obtained by subtracting this correction from the initially calculated complexation energy $\Delta E_{\text{corrected}} = \Delta E_{\text{complexation}} - \Delta E_{\text{CP}}$. Intramolecular BSSE can occur when non-bonded parts of a molecule, though spatially close, are not bonded and thus benefit artificially from the additional variational flexibility provided by neighboring basis functions, leading to errors in computed relative conformational energies. A parameterized correction for intramolecular BSSE, known as geometrical CP (gCP), requires only geometric information and straightforwardly computed overlap integrals [51, 52].
In addition to BSSE, the basis set incompleteness error (BSIE) arises from the inherent limitations of any finite basis set: for an atom, the BSIE is defined as the difference between the value obtained with a specific basis set and the CBS limit. While in a molecular system the error is a combination of BSIE and BSSE effects, it is difficult to disentangle both effects [20]. Various approaches have been developed to minimize both BSSE and BSIE, including the use of larger basis sets to approach the complete basis set limit or the CP correction method described above. Ultimately, the careful selection of a balanced basis set and the implementation of correction schemes such as the CP method are essential for obtaining accurate interaction energies, especially when high precision is required for weak intermolecular interactions.

Hartree-Fock Method

One of the most important methods to determine a surrogate wavefunction for molecules was developed by Hartree and Fock around 1930 [53, 54]. Albeit relatively concise to formulate, its importance can barely be overstated, laying the foundation of quantum chemical computational calculations and giving rise to many modern QC methods.
In the Hartree-Fock (HF) ansatz, for a system of N electrons, the electronic ground state wavefunction $\Psi_{e,0}$ is described by a single normalized Slater determinant [55–57] composed of N independent

one-electron wavefunctions $\phi_i = \sigma_i \cdot \psi_i$ with spin component $\sigma_i \in \{\alpha, \beta\}$ and spatial orbital ψ_i:

$$\Psi_{e,0} \approx \Phi_{\mathrm{HF}}(1, 2, \ldots, N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_1(1) & \phi_2(1) & \cdots & \phi_N(1) \\ \phi_1(2) & \phi_2(2) & \cdots & \phi_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(N) & \phi_2(N) & \cdots & \phi_N(N) \end{vmatrix} \tag{2.21}$$

The Slater determinant represents an antisymmetrized product of one-particle functions, abiding the Pauli principle $\Phi_{\mathrm{HF}}(i, j) = -\Phi_{\mathrm{HF}}(j, i)$ [58]. Moreover, if two electrons were to occupy the same quantum state, the determinant would vanish, reflecting the impossibility for fermions to share identical quantum numbers $\Phi_{\mathrm{HF}}(i, i) = 0$. In the case of non-interacting fermions, a single Slater determinant represents the exact wavefunction.
Following the variational principle from Equation 2.6, in the HF ansatz a set of MOs ϕ_i is sought that minimizes the energy expectation value. Using the expression for the electronic Hamiltonian from Equation 2.15

$$\hat{\mathrm{H}}_e = \sum_i^N \hat{\mathrm{h}}_i + \sum_{ij}^N \hat{\mathrm{g}}_{ij} \tag{2.22}$$

and inserting relation Equation 2.13 yields the energy expectation value with respect to $\Psi_{e,0}$

$$E = \langle \Psi_{e,0} | \hat{\mathrm{H}}_e | \Psi_{e,0} \rangle = \sum_i^N \langle \phi_i | \hat{\mathrm{h}}_i | \phi_i \rangle + \frac{1}{2} \sum_{ij}^N \left(\langle \phi_i \phi_j | \frac{1}{r_{ij}} | \phi_i \phi_j \rangle - \langle \phi_i \phi_j | \frac{1}{r_{ij}} | \phi_j \phi_i \rangle \right) \tag{2.23}$$

To minimize the energy E the Lagrange multipliers ϵ_{ij} are introduced to enforce the constraint of orbital orthonormality $\langle \phi_i | \phi_j \rangle = \delta_{ij}$ and form the Lagrangian

$$\mathcal{L} = E - \sum_{ij}^N \epsilon_{ij} \left(\langle \phi_i | \phi_j \rangle - \delta_{ij} \right) \tag{2.24}$$

The stationarity condition $\delta \mathcal{L} = 0$ with respect to variations in the spin-orbitals ϕ_i leads to the HF equations:

$$\hat{\mathrm{f}}_i \, \phi_i = \epsilon_i \, \phi_i \quad i = 1, \ldots, N, \tag{2.25}$$

where the one-electron Fock operator $\hat{\mathrm{f}}_i$, which combines one-electron and effective two-electron interactions, is defined as

$$\hat{\mathrm{f}}_i = \hat{\mathrm{h}}_i + \sum_j^N \left(\hat{\mathrm{J}}_{ij} - \hat{\mathrm{K}}_{ij} \right) \tag{2.26}$$

Thereby, the two-electron part is expressed by Coulomb operator $\hat{\mathrm{J}}_j$ which represents the electron-electron Coulomb repulsion, approximated by the electron interaction with a mean field generated by

all other electrons

$$\hat{\mathrm{J}}_{ij}|\phi_i\rangle = \langle\phi_j|\frac{1}{r_{ij}}|\phi_j\rangle|\phi_i\rangle \tag{2.27}$$

and the Exchange operator $\hat{\mathrm{K}}_j$ which originates from the wavefunction anti-symmetry due to the Pauli principle and occurs only for two electrons of the same spin

$$\hat{\mathrm{K}}_{ij}|\phi_i\rangle = \langle\phi_j|\frac{1}{r_{ij}}|\phi_i\rangle|\phi_j\rangle \tag{2.28}$$

To solve the eigenvalue problem in Equation 2.25, the MOs ϕ_i can be expanded as linear combinations of AOs, as shown in Equation 2.18. Note that the canonical MOs ϕ_i serve as a convenient basis for performing the variational energy minimization. However, the total energy depends only on the full wavefunction, which is represented by a Slater determinant built from the occupied MOs (Equation 2.21). Given that the value of a determinant is invariant under unitary transformations of its rows and columns, the wavefunction and thus the total energy, remains unchanged under unitary rotations among the occupied MOs. Since the Coulomb and Exchange contribution cancel for an identical electron $\hat{\mathrm{K}}_{ii}|\phi_i\rangle = \hat{\mathrm{J}}_{ii}|\phi_i\rangle$, no artificial self-interaction of an electron with itself occurs. This exact cancellation ensures that the Hartree-Fock method remains free from self-interaction error (SIE).

The Hartree-Fock equations 2.25 in matrix form, known as the Roothaan-Hall equations [59, 60], can be formulated as

$$\mathbf{FC} = \mathbf{SC}\boldsymbol{\epsilon} \tag{2.29}$$

which involves the Fock matrix $\mathbf{F}$, the LCAO coefficient matrix $\mathbf{C}$ to represent MOs as linear combination of AOs, the overlap matrix $\mathbf{S}$ and the matrix of orbital energies $\boldsymbol{\epsilon}$. The overlap matrix accounts for possible non-orthogonality of the orbitals $S_{ij} = \langle\phi_i|\phi_j\rangle$. Thereby, the Roothaan-Hall equations require an iterative solution due to the dependence of the Fock matrix $\mathbf{F}$ on the MOs themselves (c.f. Equation 2.27 and 2.28) as a given orbital can only be determined if all other occupied orbitals are known. This leads to a self-consistent field (SCF) procedure, which updates the Fock matrix $\mathbf{F}$ and MO coefficients $\mathbf{C}$ iteratively until convergence is achieved. The process commences with an initial guess for the MO coefficient matrix $\mathbf{C}^{t=0}$, often based on the core Hamiltonian (i.e. no electron-electron interactions) or a superposition of atomic densities [61]. From these, the electron density matrix is calculated $\rho^{t=0} = 2\sum_i^{\mathrm{occ}} C_{\mu i}^{t=0} C_{\nu i}^{t=0}$, the Fock matrix is constructed $\mathbf{F}^t$ and the Roothaan-Hall eigenvalue problem is solved to yield new coefficients $\mathbf{C}^{t+1}$, which are then in turn used to update the density matrix ρ^{t+1}. This cycle continues until the change in energy or density falls below a defined threshold. Once convergence is reached, the resulting orbitals represent a variationally optimized single-determinant approximation to the ground-state wavefunction of the system. The self-consistent set of molecular orbitals obtained from this procedure is then used to compute the HF energy according to

$$E_{\mathrm{HF}} = \sum_i^N \langle\phi_i|\hat{\mathrm{h}}_i|\phi_i\rangle + \frac{1}{2}\sum_{ij}^N \left(\langle\phi_i|\hat{\mathrm{J}}_{ij}|\phi_i\rangle - \langle\phi_i|\hat{\mathrm{K}}_{ij}|\phi_i\rangle\right) \tag{2.30}$$

So far, identical orbitals ϕ_i have been used for both spin-up and spin-down electrons. Whereas for open-shell species the unrestricted HF method is often employed, allowing electrons of different spins to occupy distinct spatial orbitals $\phi_i^{\uparrow}$ and $\phi_i^{\downarrow}$, resulting in different LCAOs for each spin channel.
It is noteworthy that the computational effort for HF calculations formally scales as $\mathcal{O}(N_{\mathrm{AO}}^4)$ with respect to the number of atomic orbitals N_{AO}. The primary limitation of the HF method lies in its mean-field nature: each electron experiences only the averaged potential of all other electrons. This approximation does not account for the full instantaneous electron-electron interaction $\hat{\mathrm{V}}_{ee}$. Moreover, the use of a single Slater determinant includes electron repulsion only in an averaged sense. Consequently, even in the CBS limit, the HF method does not yield the exact solution to the electronic Schrödinger equation within the Born-Oppenheimer approximation. Rather it provides the best possible wave function that can be obtained using a single determinant [20]. The discrepancy between the exact non-relativistic energy and the HF energy is known as the electron correlation energy, defined as $E_{\mathrm{corr}} = E - E_{\mathrm{HF}}$ [62, 63]. Electron correlation is commonly divided into dynamic and static correlation. Dynamic correlation accounts for the instantaneous motion of electrons to avoid one another. While HF includes Fermi correlation for same-spin electrons through antisymmetrization, it neglects Coulomb correlation between electrons of opposite spin. Static correlation becomes significant when a single Slater determinant fails to describe the electronic ground state adequately, e.g. in systems with near-degenerate configurations requiring a multi-determinant description. Although the correlation energy is often small in absolute terms ($E_{\mathrm{corr}}/E < 1\,\%$), it is essential for accurately capturing relative energies, reaction barriers and molecular properties [20].

2.1.2 Density Functional Theory

Density functional theory (DFT) is a quantum mechanical method family in which the ground-state properties of a molecular system are determined using the electron density $\rho(\mathbf{r})$ as the fundamental variable rather than the many-electron wavefunction [64]. In doing so, DFT transits from using individual particle coordinates to using the total density, circumventing the need to solve the many-electron Schrödinger equation directly. The theoretical foundation for DFT is laid by the Hohenberg-Kohn theorems [65]: The first theorem asserts that the ground-state electron density uniquely determines the external potential $\mathrm{V}_{\mathrm{ext}}(\mathbf{r})$ (up to a constant), and thus all observables of the system. Note that for an isolated system, the external potential equates to the nucleus-electron potential V_{ne}. The second theorem establishes a variational principle: for any trial density $\tilde{\rho}(\mathbf{r})$ that corresponds to a valid external potential, the energy functional obeys

$$E[\tilde{\rho}] \geq E_0 \tag{2.31}$$

where E_0 is the true ground-state energy. Thus, given the energy functional $E[\rho]$ the ground state and its energy can be determined by systematically varying the electron density [66]. Early attempts of DFT development tried to express all energy contributions as a functional of the electron density

$$E_{\mathrm{DFT}}[\rho] = \mathrm{T}_e[\rho] + \mathrm{V}_{ne}[\rho] + \mathrm{V}_{ee}[\rho] \tag{2.32}$$

$$= \mathrm{T}_e[\rho] + \mathrm{V}_{ne}[\rho] + \mathrm{J}[\rho] + \mathrm{K}[\rho] \tag{2.33}$$

where $\mathrm{T}_e[\rho]$ signifies the kinetic energy of the electrons and $\mathrm{V}_{ne}[\rho]$ the nuclear-electron interaction. The electron-electron interaction $\mathrm{V}_{ee}[\rho]$ is divided into the classical Coulomb repulsion $\mathrm{J}[\rho]$ and the

Exchange contribution $\mathrm{K}[\rho]$. However, these orbital-free approaches had poor performance due to inadequate description of the kinetic energy $\mathrm{T}_e[\rho]$ [20].
In Kohn-Sham DFT (KS-DFT)[11] the concept of orbitals is reintroduced to improve upon the determination of the kinetic energy and find a variational ansatz for optimizing $\rho(\mathbf{r})$. Thereby, the KS formalism decomposes the complex many-body interactions into a set of self-consistent single-particle equations, rendering DFT practical for applications in QC [67]. For this purpose, in KS-DFT the kinetic energy is calculated using an auxiliary set of orbitals that reproduces the exact ground-state electron density.
The kinetic energy functional is divided into two parts: one that can be calculated exactly and a smaller correction term. For this purpose, a systematic framework for separating the kinetic energy is provided by introducing a modified Hamiltonian. A λ-dependent Hamiltonian can be formulated to establish an adiabatic connection between the non-interacting reference system and the fully interacting electron system

$$\hat{\mathrm{H}}(\lambda) = \hat{\mathrm{T}} + \hat{\mathrm{V}}_{\mathrm{ext}}(\lambda) + \lambda\, \hat{\mathrm{V}}_{ee} \tag{2.34}$$

where $\hat{\mathrm{T}}$ is the kinetic energy operator, $\hat{\mathrm{V}}_{\mathrm{ext}}(\lambda)$ is an external potential that is adjusted for each value of λ to ensure that the ground-state electron density remains constant, and $\hat{\mathrm{V}}_{ee}$ represents the electron-electron interaction. At $\lambda = 1$, $\hat{\mathrm{V}}_{\mathrm{ext}}(\lambda)$ corresponds to the actual electron-nuclear potential $\hat{\mathrm{V}}_{ne}$, yielding the Hamiltonian of the real interacting system. In contrast, at $\lambda = 0$ the electrons do not interact, and the KS orbitals ϕ_i allow to express the kinetic energy of the fictitious non-interacting electrons exactly as

$$\mathrm{T}_{\mathrm{SD}}[\rho] = \sum_{i=1}^{N_{\mathrm{elec}}} \langle \phi_i \mid -\frac{1}{2}\nabla^2 \mid \phi_i \rangle \tag{2.35}$$

which is equivalent to the HF theory using a Slater determinant utilizing KS orbitals. Note that KS-DFT is thereby closely related to the HF method, sharing the same formulas for kinetic energy, electron-nuclear interactions, and Coulomb electron-electron interactions. Despite being only an approximation for the $\lambda = 1$ case, this formulation has two major advantages. First, by computing $\mathrm{T}_{\mathrm{SD}}[\rho]$ exactly from the orbitals, the method captures about 99 % of the kinetic energy correctly. Second, because $E_{\mathrm{XC}}[\rho]$ is typically much smaller than the kinetic energy, the errors introduced by approximate treatments of exchange and correlation are significantly mitigated. The remaining part, the kinetic correlation energy, is incorporated into the exchange-correlation energy $E_{\mathrm{XC}}[\rho]$, which remains the only unknown functional in the KS energy expression. The exchange-correlation energy is defined by the residual term

$$E_{\mathrm{XC}}[\rho] = \Big(\mathrm{T}_e[\rho] - \mathrm{T}_{\mathrm{SD}}[\rho]\Big) + \Big(\mathrm{V}_{ee}[\rho] - \mathrm{J}[\rho]\Big) \tag{2.36}$$

whereby the first parenthesis can be interpreted as the kinetic correlation energy and the second considers potential correlation and exchange energy [20]. In a sense, the postulation of the theoretically exact $E_{\mathrm{XC}}[\rho]$ functional in DFT theory is analogous to the restriction to a single Slater determinant in HF, where both theories assume non-interacting electrons. Generally, $E_{\mathrm{XC}}[\rho]$ accounts for only a small portion of the total energy, and even relatively simple approximations yield remarkably accurate

computational results [20]. The total energy expression in KS-DFT is given by

$$E_{\text{KS-DFT}}[\rho] = \mathrm{T}_{\text{SD}}[\rho] + \mathrm{V}_{ne}[\rho] + \mathrm{J}[\rho] + E_{\text{XC}}[\rho] \tag{2.37}$$

Furthermore, $E_{\text{XC}}[\rho]$ encapsulates the exchange-correlation effects that account for the difference between the true and the non-interacting kinetic energies stemming from the auxiliary reference system. Thereby, $E_{\text{XC}}[\rho]$ includes for all many-body interactions beyond the classical electrostatic contributions. However, the exact form of $E_{\text{XC}}[\rho]$ is unknown [68, 69], presenting a fundamental challenge in DFT, as it prevents the systematic improvement of known DFT models. Consequently, a hierarchy of approximations has been developed to approach this exact form, balancing computational efficiency with accuracy. Figure 2.2 gives an overview over the different levels of DFT functionals and the approximations used on every level.

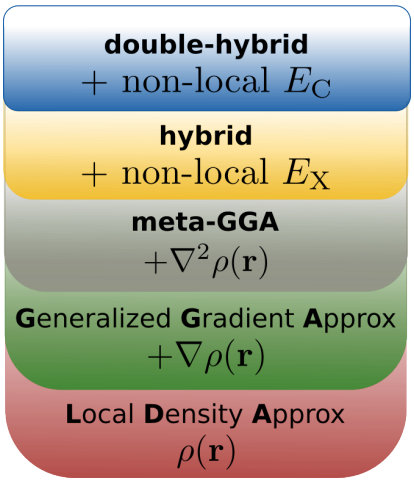

Figure 2.2: Overview over the different density functional approximation classes.

Local Density Approximation

The simplest approximation is the local density approximation (LDA), derived from the uniform electron gas (UEG) [12, 70, 71]. Analytic expressions for the correlation energy of the UEG are available in the high- and low-density limits [72–74], while accurate quantum Monte Carlo simulations [75, 76] at intermediate densities yield precise values for the correlation energy density [77]. Furthermore, the LDA exchange part is given by [78, 79]

$$E_{\text{X}}^{\text{LDA}}[\rho] = -\frac{3}{4}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} \int \rho(\mathbf{r})^{\frac{4}{3}} d\mathbf{r} \tag{2.38}$$

Although LDA is computationally efficient, by assuming a uniform electron density its usability is limited to systems exhibiting pronounced electron density homogeneities, such as metals. LDA implementations include the VWN functional [80] for instance.

Generalized Gradient Approximation

To improve upon LDA, the generalized gradient approximation (GGA) not only incorporates the local electron density $\rho(\mathbf{r})$ but also its gradient $\nabla\rho(\mathbf{r})$ [81]. For a GGA the exchange-correlation energy is

given by

$$E_{\mathrm{XC}}^{\mathrm{GGA}}[\rho] = \int \epsilon_{\mathrm{XC}}^{\mathrm{LDA}}[\rho(\mathbf{r})]\, F_{\mathrm{XC}}^{\mathrm{GGA}}\left[\rho(\mathbf{r}), \nabla\rho(\mathbf{r})\right] d\mathbf{r} \tag{2.39}$$

where $\epsilon_{\mathrm{XC}}^{\mathrm{LDA}}$ is the local exchange-correlation energy density and $F_{\mathrm{XC}}^{\mathrm{GGA}}$ is an enhancement factor that accounts for density gradients [82]. As $F_{\mathrm{XC}}^{\mathrm{GGA}}$ incorporates information from the immediate vicinity of the local electron density, GGAs are classified as semi-local DFT functionals. Nevertheless, GGAs still lack sufficient non-local exchange to cancel self-interaction, hence resulting in an over-delocalization of the electron density (as electrons spuriously repel themselves). Nonetheless, GGAs such as PBE [82] or BLYP [74], are widely used for molecular systems.

Meta-Generalized Gradient Approximation

The meta-generalized gradient approximation (meta-GGA) extends the GGA approach by including additional information on the curvature of electron density $\nabla^2\rho(\mathbf{r})$ [81]. Due to its numerical instability, $\nabla^2\rho(\mathbf{r})$ is usually substituted by the kinetic orbital energy $\tau(\mathbf{r})$ [83], which is defined as

$$\tau(\mathbf{r}) = \frac{1}{2}\sum_{\sigma}\sum_{i=1}^{N_\sigma}\left|\nabla\phi_i(\mathbf{r})\right|^2 \tag{2.40}$$

with $\phi_i(\mathbf{r})$ denoting the KS orbitals and N_σ the number of electrons with spin $\sigma \in \{\alpha, \beta\}$. Meta-GGA functionals (e.g. TPSS [84], r^{2}SCAN [85]) use the most semi-local information among the common DFT functionals and only lack description of non-local effects. For the calculation of thermochemistry meta-GGAs provide generally good results [86]. The formal scaling of GGAs and meta-GGAs can be reduced to $\mathcal{O}(N_{\mathrm{AO}}^3)$ using approximations such as the resolution-of-identity method [87].

Hybrid Functionals

Hybrid functionals improve upon semi-local approximations by incorporating a fraction of non-local exact exchange from HF theory ("Fock exchange"), thereby also including information about the occupied KS orbitals [68]. The theoretical foundation for hybrid functionals is provided by the Adiabatic Connection Formula (ACF) [88], which expresses the exchange-correlation energy as an integral over the coupling parameter λ from Equation 2.34 that continuously connects the non-interacting KS reference system ($\lambda = 0$) with the fully interacting physical system ($\lambda = 1$)

$$E_{\mathrm{XC}} = \int_0^1 \left\langle \Psi_\lambda \middle| \hat{\mathrm{V}}_{\mathrm{XC}}(\lambda) \middle| \Psi_\lambda \right\rangle d\lambda \tag{2.41}$$

This formulation allows for the systematic inclusion of exact exchange at the non-interacting end of the integration path, where the exchange energy matches that of HF theory. Hybrid functionals can thus be viewed as approximations to the ACF, where the exchange energy is constructed as a weighted average of exact and approximate contributions. The general expression for a hybrid XC functional is

$$E_{\mathrm{XC}}^{\mathrm{hybrid}} = E_{\mathrm{C}}^{\text{(meta-)GGA}} + (1 - a_{\mathrm{X}})\, E_{\mathrm{X}}^{\text{(meta-)GGA}} + a_{\mathrm{X}}\, E_{\mathrm{X}}^{\mathrm{HF}} \tag{2.42}$$

where a_{X} is the mixing parameter and $E_{\mathrm{X}}^{\mathrm{HF}}$ represents the non-local exchange energy. The inclusion of non-local exchange reduces the SIE, and enhances the accuracy of reaction barriers [89]. An overlarge

addition of non-local exchange may lead to overestimated orbital energy gaps and could worsen thermochemical predictions that are already well captured by (meta-)GGA functionals. Conversely, a lower HF exchange fraction tends to better preserve the accurate thermochemistry of semilocal functionals while providing a less aggressive correction of delocalization errors. The optimal fraction of exact exchange depends on the property of interest, and therefore varies between different functionals. Because exchange is evaluated non-locally by integrating $\rho(\mathbf{r}, \mathbf{r}')$ over two spatial coordinates, the computational cost increases to a formal $\mathcal{O}(N_{\mathrm{AO}}^4)$ scaling [81]. Examples for hybrid DFT functionals include B3LYP [68, 74, 90], PBE0 [91, 92] or TPSSh [93, 94].

Range-Separated Hybrid Functionals

Range-separated hybrid (RSH) functionals address inherent limitations in conventional hybrid approaches by partitioning the electron-electron Coulomb operator into short-range and long-range components. This partitioning is typically achieved using the error function

$$\frac{1}{r_{12}} = \frac{1 - a_{\mathrm{X}}' - a_{\mathrm{X}}'' \operatorname{erf}(\omega r_{12})}{r_{12}} + \frac{a_{\mathrm{X}}' + a_{\mathrm{X}}'' \operatorname{erf}(\omega r_{12})}{r_{12}} \tag{2.43}$$

where ω controls the range separation and a_{X}', a_{X}'' are parameters that weight the short- and long-range contributions, respectively. In practice, the short-range part is treated with a density-based exchange functional, while the long-range part is evaluated using exact HF exchange. The motivation for using RSH functionals arises from systematic deficiencies in traditional DFT methods, which often underestimate excitation energies for charge-transfer and Rydberg states. Such errors are linked to self-interaction and an overdelocalization of the electron density, resulting in an unphysical lowering of energies for delocalized systems [20]. By incorporating 100 % HF exchange in the long-range, RSH functionals restore the correct asymptotic behavior of the exchange-correlation potential and effectively eliminate SIE at large distances. A key aspect of the RSH approach is the free parameter ω, which defines the crossover between the short- and long-range interactions. This parameter can be tuned either empirically [95, 96] against experimental data or non-empirically [97] by enforcing conditions such as Koopmans' theorem [98]. The latter being a so-called "optimally tuned" approach that yields a distinct ω for each system.

This tunability of RSH functionals allows for improved predictions in systems with loosely bound electrons, such as anions and zwitterions, and helps to avoid issues like the dissociation into fractionally charged fragments. Additionally, this range-separation concept has been widely incorporated into various DFT functionals, leading to the development of methods such as ωB97X [95, 96] and ωB97X-V [99]. Alternative strategies, like the inverse separation used in the HSE functional [100, 101], which employs HF exchange at short distances and density exchange at longer ranges, offer different approaches to range separation. In addition, the HISS functional [102] introduces a modulation in which the HF exchange is initially increased and then decreased as a function of r_{12}, providing another means of fine-tuning the exchange contributions.

Double-Hybrid Functionals

Double-hybrid functionals further extend the concept of hybrid functionals by incorporating information from both occupied and virtual KS orbitals, thereby capturing non-local correlation effects [103–106]. They typically include a second-order Møller-Plesset perturbation theory (MP2)-like term to account for dynamic electron correlation [107]. Thereby, the non-local correlation correction is computed

using the MP2 formulation, employing DFT orbitals and orbital energies. The general form of a double-hybrid functional is

$$E_{\mathrm{XC}}^{\text{double-hybrid}} = (1 - a_{\mathrm{C}})E_{\mathrm{C}}^{\text{(meta-)GGA}} + a_{\mathrm{C}}\,E_{\mathrm{C}}^{\mathrm{MP2}} + (1 - a_{\mathrm{X}})E_{\mathrm{X}}^{\text{(meta-)GGA}} + a_{\mathrm{X}}\,E_{\mathrm{X}}^{\mathrm{HF}} \tag{2.44}$$

with a_{C} controlling the amount of MP2 mixing and $E_{\mathrm{C}}^{\mathrm{MP2}}$ representing the perturbative, non-local second-order correlation energy. Double-hybrid functionals were first introduced by Neese and Grimme in 2006 [105], since then, a multitude of variants have emerged [108]. Although double-hybrid functionals, like B2-PLYP [104], achieve high accuracy in QC calculations [89, 109], their MP2-like scaling of $\mathcal{O}(N_{\mathrm{AO}}^5)$ limits their applicability to moderately sized systems. Moreover, due to the presence of the MP2 expression, basis set convergence is slower compared to pure DFT methods [20].

2.1.3 Semiempirical Quantum Mechanical Methods

Semiempirical quantum mechanical (SQM) methods offer a balanced compromise between accuracy and efficiency. Usually built upon the HF or DFT formalism, these methods incorporate numerous approximations, such as neglecting or approximating two-electron integrals, and compensate for the resulting loss of information by introducing empirically derived parameters. Consequently, SQM methods have become invaluable in computational chemistry, especially for treating large molecular systems where full HF or DFT calculations would be computationally prohibitive [110]. Furthermore, SQM methods find widespread applications in molecular structure optimization and the elucidation of reaction mechanisms in both organic and inorganic chemistry [108, 111, 112], as well as in high-throughput screening [113–115] or multilevel modeling [116].
Over the last century, multiple SQM method families have been developed. The Hückel method [117–119], introduced in the 1930s, was pivotal in simplifying the treatment of π-electron systems using a SQM approach, approximating the Hamiltonian by employing a minimal basis set with only on-site and nearest-neighbor interactions. Despite its simplicity, the method proved highly effective in capturing the essential features of conjugated molecules. To improve the description of more complex systems, such as non-planar molecules or transition metal complexes [120], the extended Hückel theory was developed [121]. This method expanded the original framework by including all valence electrons and was successfully applied in many qualitative studies of inorganic and organometallic compounds [122]. Historically, the Hückel approaches were pivotal as they introduced key approximations that laid the groundwork for more advanced SQM methods [122].
Building on the conceptual foundation of Hückel theory, the tight binding (TB) approach was introduced in 1954 [123]. Closely related to the LCAO ansatz explained above [20], TB assumes that electrons are tightly bound to their parent atoms [124, 125]. Initially developed for the description of electron movements in solids, electrons are assumed to remain localized at their respective atomic centers, with limited interactions with states of surrounding atoms, leading to wavefunctions that closely resemble the atomic orbitals of isolated atoms [126].
Moreover, SQM methods can broadly be divided into those based on the HF formalism and those derived from DFT. Among the HF-based methods, early models such as NDDO (Neglect of Differential Diatomic Overlap) [127] form the basis, where differential overlap between orbitals is assumed to be small and hence the overlap matrix is replaced by a unit matrix. Thereby, under the assumption of zero differential overlap, two-electron integrals containing differential overlap between orbitals

on two different atoms are neglected, but all two-electron integrals with differential overlap between orbitals on the same atom are retained [128]. The following methods are essentially modified versions of the NDDO model, differing primarily in their treatment of core-core repulsion and distinct parameterizations. In MNDO (Modified Neglect of Diatomic Overlap) [129, 130] empirical parameters are adjusted to reproduce experimental properties from ground-state stable molecules. For this reason, MNDO does not capture correlation effects in transition and excited states well [20]. Building upon MNDO, AM1 (Austin Model 1) was developed to address some of its shortcomings by incorporating additional Gaussian functions into the core-core repulsion term [131]. This modification helps to better capture short-range interactions. PM3 (Parametric Method 3) [132, 133] represents another reparameterization of the AM1 framework, retaining the same underlying NDDO formalism. In PM3, exactly two Gaussian functions per element are used in the repulsion term, and all parameters are determined through a fully automated fit to an extensive data set. Later developments such as PM6 [134] and PM7 [135] have significantly advanced the HF-based semiempirical family. PM6 uses a much larger reference data set for parameter optimization and refines the treatment of core potentials by employing pairwise parameters for each atomic pair, allowing to reduce the number of Gaussian repulsion terms to only one for each atom [20]. Furthermore, including parametrization for third-period elements and transition metals. This improvement allows PM6 to extend its applicability to a wide range of systems, from organic molecules to inorganic complexes and biological macromolecules, though it still does not fully account for all non-covalent interactions [136]. PM7 builds on PM6 by incorporating explicit corrections for dispersion and hydrogen bonding, removing the Gaussian core-core terms for all but the H, C, N, O atoms [20]. These enhancements result in improved accuracy for typical organic molecules and a more robust overall description without the need for separate a posteriori corrections. Still, a persistent shortcoming of all NDDO-based methods is their tendency to significantly underestimate rotational barriers in bonds with partial double-bond character.
In contrast to the HF-based methods, SQM approximations to DFT include Density Functional Tight Binding (DFTB) [137–141]. DFTB takes a fundamentally different approach by expanding the total electronic energy around a reference electron density ρ_0, usually derived from a superposition of neutral atomic densities, and retaining only the leading terms in the expansion:

$$E[\rho] = E^{(0)}[\rho_0] + E^{(1)}[\rho_0, \delta\rho] + E^{(2)}[\rho_0, (\delta\rho)^2] + E^{(3)}[\rho_0, (\delta\rho)^3] + \cdots \tag{2.45}$$

with $\delta\rho = \rho - \rho_0$. Typically, this series expansion is truncated after the third-order term [142, 143], as higher order terms contribute only marginally to the total energy while substantially increasing computational complexity and the number of parameters needed. In the regime of small density fluctuations, the first three terms capture the dominant physical effects, providing a good balance between accuracy and efficiency. Similar to NDDO methods, DFTB employs a minimal basis set, neglects three- and four-center integrals, and treats only valence electrons explicitly [20]. The core-core repulsion energy in DFTB corresponds to the zeroth-order term including no electronic contribution and is typically parameterized using spline functions fitted to corresponding all-electron DFT results. In its simplest form, the interactions between atomic orbitals are treated in a TB-like manner using precomputed Slater-Koster tables derived from DFT calculations. This makes DFTB especially suitable for large systems such as biomolecules or periodic solids where full DFT calculations would be computationally prohibitive [144, 145]. However, the standard DFTB formulation struggles with highly polar or charged systems. To address this limitation, the Self-Consistent Charge extension (SCC-DFTB) [141] introduces an iterative scheme to adjust Mulliken atomic partial charges. Furthermore, additional

polarization terms can be added to better describe proton affinities and hydrogen binding [142, 143]. Like standard DFT, DFTB does not inherently account for dispersion interactions, but these can be added a posteriori [146]. Additional empirical corrections can also be applied to better describe specific interactions that are poorly captured by DFTB alone, such as halogen bonding [147].
Another prominent DFT-based SQM method is the extended Tight Binding approach. As it is of particular relevance for this thesis, the following section will present the method developed by Bannwarth et al. (2019) [17, 18] in detail.

Extended Tight Binding

Inspired by the successes of DFT-based tight binding approaches, the so-called "extended Tight Binding" (xTB) methods [17, 18, 148, 149] were developed, in which additional corrections and interactions are incorporated to further refine the electronic structure description in complex systems. The GFN-xTB method family represents a modern class of semiempirical quantum mechanical approaches, specifically tailored for efficient and accurate prediction of molecular geometries, vibrational frequencies, and non-covalent interactions. It is implemented in two parametrizations: GFN1-xTB [17] and its successor GFN2-xTB [18]. Both methods are based on a minimal valence basis of atom-centered, contracted Gaussian functions (STO-mG), employing polarization functions for most main group elements to better capture hypervalent bonding situations. The GFN-xTB Hamiltonian closely resembles that of DFTB, with GFN2-xTB incorporating electrostatic and exchange-correlation effects up to second order in a multipole expansion. Contrary to DFTB's element pair-specific parameters, GFN-xTB requires only a global and element-specific parameter set. Both methods GFN1-xTB and GFN2-xTB are parametrized for elements up to radon ($Z = 86$), thereby including lanthanoids via an interpolation scheme. For lanthanoids, where the $4f$ shell is very compact and has thus little influence on chemical bonding, the "f-in-core" approximation [18, 150] is applied, effectively treating all lanthanoids as lanthanum while implicitly accounting for the $4f$ electrons through parametrization. Notably, GFN-xTB being spin-restricted, favoring low-spin configurations, is generally advantageous for electronically complex systems, as it results in more stable calculations compared to those involving high-spin states.
Similar to DFTB, the Taylor expansion of the total electronic energy in Equation 2.45 forms the basis for the derivation for the xTB energy. In the GFN1-xTB framework the total energy is decomposed as

$$E_{\mathrm{GFN1-xTB}} = E^{(0)}_{\mathrm{rep}} + E^{(0)}_{\mathrm{D3}} + E^{(0)}_{\mathrm{XB}} + E^{(1)}_{\mathrm{EHT}} + E^{(2)}_{\mathrm{ies}} + E^{(3)}_{\mathrm{ies}} + T_{\mathrm{el}} S_{\mathrm{el}} \tag{2.46}$$

where each term accounts for distinct physical interactions. The zeroth-order energy terms do not depend on the charge density but account for geometric factors. Density independent terms include the Coulomb repulsion energy $E^{(0)}_{\mathrm{rep}}$. It is given by the classical pairwise repulsion term, modified by screening effects:

$$E_{\mathrm{rep}} = \frac{1}{2} \sum_{i,j} \frac{Z^{\mathrm{eff}}_i Z^{\mathrm{eff}}_j}{r_{ij}} \exp\left(-\sqrt{\alpha_i \alpha_j}\, r_{ij}^{k_f}\right) \tag{2.47}$$

with $Z^{\mathrm{eff}}_{\{i,j\}}$ the effective nuclear charge, $\alpha_{\{i,j\}}$ element-specific parameters, and a global scaling factor k_f. All these parameters are fitted, leading to e.g. optimized values of Z^{eff} that deviate from the initial nuclear charge Z by up to 30 % [17]. Another density independent contribution is the D3

dispersion correction $E_{\mathrm{D3}}^{(0)}$ [146]. Since GFN1-xTB accounts only for isotropic electrostatics, an additional correction $E_{\mathrm{XB}}^{(0)}$ is required to accurately describe halogen bonding:

$$E_{\mathrm{XB}}^{(0)} = \sum_{\mathrm{XB}} f_{\mathrm{damp}} k_X \left(\zeta^{12} - k_{X2}\zeta^6\right) / \left(1 + \zeta^{12}\right), \quad \text{with} \quad \zeta = \left(\frac{r_{\mathrm{cov},AX}}{r_{AX}}\right) \tag{2.48}$$

Thereby, the correction takes the form of a modified Lennard-Jones potential with the effective covalent distance $r_{\mathrm{cov},AX} = k_{XR}(r_{\mathrm{cov},A} + r_{\mathrm{cov},X})$ and global, halogen-specific parameters k_X, k_{X2} and k_{XR}. The damping depends on the angle $\theta = \sphericalangle AXB$ between the acceptor atom A closest to the halogen, the halogen X, and the donor atom B (Nitrogen or Oxygen) and is designed such that it vanishes for non-planar configurations:

$$f_{\mathrm{damp}} = \left(\frac{1}{2} - \frac{1}{4}\cos\theta\right)^6 \tag{2.49}$$

The first-order energy terms depend linearly on charge density fluctuations, with covalent bond formation described by Extended Hückel Theory (EHT) providing the dominant contribution:

$$E_{\mathrm{EHT}}^{(1)} = \sum_{\mu,\nu} P_{\mu\nu} H_{\mu\nu}^{\mathrm{EHT}} \tag{2.50}$$

where $P_{\mu\nu}$ is the density matrix in a non-orthogonal basis, and $H_{\mu\nu}^{\mathrm{EHT}}$ describes the interaction of neutral atoms accounting for atomic-environment effects via fractional coordination numbers. The density matrix is obtained by solving the Roothaan-Hall equations (c.f. Equation 2.29). To simplify the description of covalent bonding, only valence electrons are considered. Effective one-electron interactions are computed by scaling the averaged on-site level energies $H_{\mu\mu/\nu\nu}^{\mathrm{EHT}}$ with the overlap matrix $S_{\mu\nu}$ and a shell-pair- and distance-dependent polynomial $\Pi_{\mu\nu}$:

$$H_{\mu\nu}^{\mathrm{EHT}} = \frac{H_{\mu\mu}^{\mathrm{EHT}} + H_{\nu\nu}^{\mathrm{EHT}}}{2} \cdot S_{\mu\nu} \cdot \Pi_{\mu\nu} \tag{2.51}$$

The function $\Pi_{\mu\nu}$, which depends on distance and orbital shape, is defined as:

$$\Pi_{\mu\nu} = \left(1 + k_{A,l}\zeta\right)\left(1 + k_{B,l}\zeta\right), \quad \text{with} \quad \zeta = \left(\frac{r_{AB}}{r_{\mathrm{cov},A} + r_{\mathrm{cov},B}}\right)^{1/2} \tag{2.52}$$

where l is the azimuthal quantum number, r_{cov} denotes the covalent radius, and k are element-specific parameters. Additionally, the initial on-site energies h_μ are scaled based on the local atomic environment, incorporating D3 coordination number CN and empirical shell-specific parameters k:

$$H_{\mu\mu}^{\mathrm{EHT}} = h_\mu(1 + k_\mu \mathrm{CN}_\mu) \tag{2.53}$$

The second-order energy terms describe interatomic electrostatic interactions in an isotropic manner:

$$E_{\mathrm{ies}}^{(2)} = \frac{1}{2}\sum_{\mu,\nu} q_\mu J_{\mu\nu} q_\nu \tag{2.54}$$

where q_μ and q_ν are the atomic charges, and the interaction is mediated by the Coulomb tensor J. In GFN1-xTB, the charge-charge interaction kernel is defined as:

$$J_{\mu\nu} = \sqrt{\frac{1}{r_{AB}^2 + f_{\text{avg}}(U_\mu, U_\nu)^{-2}}} \tag{2.55}$$

Here, U represents the Hubbard parameters of the respective orbitals, characterizing their chemical hardness. The function f_{avg} provides an averaged value of these parameters, typically using arithmetic or harmonic means. Since the atomic reference is assumed to be spherical, the chemical hardness U is uniquely defined for each angular momentum quantum number l and chemical element. Thereby, atom-resolved partial charges q_A for atom A can be obtained from the sum of orbital charges $q_A = \sum_{\mu \in A} q_\mu$. The third-order energy terms capture on-site exchange-correlation effects using the shell-resolved partial charges

$$E_{\text{ies}}^{(3)} = \frac{1}{3} \sum_A \Gamma_A \, q_A^3 \tag{2.56}$$

with Γ_A being an atom-specific Hubbard parameter and q_A the Mulliken charge on atom A.
To incorporate Fermi smearing, which accounts for fractional orbital occupations, the electronic free energy is introduced. This approach considers different electronic temperatures T_{el} and an electronic entropy S_{el}, following a Fermi distribution for varying occupation numbers. The finite electronic temperature framework facilitates covalent bond dissociation and enhances the stability of GFN-xTB methods, particularly for describing transition states and transition metal complexes.

GFN2-xTB refines the GFN1-xTB framework by incorporating improved long-range and anisotropic interactions. Its total energy expression is given by

$$E_{\text{GFN2-xTB}} = E_{\text{rep}}^{(0)} + E_{\text{EHT}}^{(1)} + E_{\text{ies}}^{(2)} + E_{\text{aes}}^{(2)} + E_{\text{ies,l}}^{(3)} + E_{\text{D4}}^{(\infty)} + T_{\text{el}} S_{\text{el}} \tag{2.57}$$

where the D3 dispersion correction is replaced by the D4 dispersion model, along with anisotropic electrostatic terms, eliminating the need for explicit halogen bonding corrections and using shell-resolved partial charges in the third-order onsite electrostatics. The anisotropic electrostatic terms are realized through a multipole expansion, extending the interaction tensors beyond simple point-charge interactions to include higher-order dipole-dipole and charge-quadrupole terms.

Further implementations of TB include the PTB model [151], which focuses on obtaining the electronic density matrix ρ at DFT-level accuracy. By employing a deeply contracted, polarized valence double-zeta basis set, PTB eliminates the major accuracy bottleneck of traditional SQM approaches and conducts only two self-consistent field steps after which an interpolation scheme is employed. Although no explicit energy expression is provided, PTB – constructed in the spirit of GFN2-xTB and enhanced with additional corrective terms – delivers excellent electronic properties while reducing computational cost by orders of magnitude compared to conventional DFT methods.

2.2 Machine Learning

Machine Learning (ML) is a subfield of Artificial Intelligence (AI) focused on algorithms that learn patterns from data [152]. ML algorithms are computational methods which are typically statistical and data-driven and characterized by their ability to identifying patterns and relationships within data. Instead of relying on explicit, hard-coded instructions, ML algorithms build internal models using statistical techniques and iterative optimization during the training phase [153]. In contrast, AI encompasses any general pursuit of intelligent behavior in machines to mimic human intelligence, which includes rule-based systems, logic programming, and other methods not necessarily learning from data (c.f. [154]).
In recent years, ML has been increasingly employed in QC to efficiently predict molecular properties. Conventional first-principles methods, such as DFT, become computationally prohibitive as system size grows (c.f. section 2.1). ML models can overcome this limitation by directly mapping molecular inputs to target properties using data obtained from QC calculations. For this purpose, ML models require sufficiently diverse and high-quality training data as predictions may become unreliable for molecules substantially outside the training distribution. Although the initial generation of a representative dataset and the training process itself can be computationally costly, trained ML models provide fast predictions at often negligible computational cost, typically scaling linearly with system size $O(N)$. As a result of providing rapid predictions and the capability to compute large structures at reasonable cost, ML models significantly accelerate exploration of chemical space, enabling high-throughput screening tasks in drug discovery and large-scale applications in molecular simulations, where systems comprising thousands of atoms at near-DFT accuracy are modeled [155–157].
Primary focus of ML in the context of this thesis lies on supervised learning, where the task is to approximate a mapping function from an input (molecular representation) to a target property (energy, force, dipole moment, ionization potential, etc.) based on given input-output pairs as training reference. However, for a holistic view on ML, it is worth mentioning other paradigms of ML in the following: Unsupervised learning deals with finding patterns or structure in unlabeled data. In chemistry, unsupervised methods can be used, for example, to cluster molecules according to structural similarity or to learn low-dimensional representations of molecular conformations [158, 159]. Generative unsupervised models such as autoencoders [152, 160], generative adversarial networks [161], or diffusion models [162] can learn distributions of molecular structures and have been used to propose new molecules or materials with desired attributes by sampling the learned distribution [163]. Reinforcement learning involves an agent learning a policy for making sequential decisions that maximize a defined reward. In QC and drug design, it has been employed for tasks such as optimizing multi-step reaction routes and design chemical libraries with a bias towards compounds within a desired range of physical properties [164, 165].

Parallels between traditional QC approaches and ML
As described in section 2.1, DFT research aims at finding a functional $E_{\mathrm{DFT}}[\rho]$ that maps electron density $\rho(\mathbf{r})$ to the energy of a system (cf. [11, 65]). Since the exact functional remains unknown, researchers typically construct approximate functionals (e.g. LDA, GGA, hybrid functionals) by enforcing known physical constraints and fitting to empirical data or higher-level calculations. Similarly, SQM methods determine optimal parameters by fitting to chemical reference data, effectively approximating physical interactions through a parameterized model. Both DFT's search for an appropriate functional form and SQM's parameter optimization are reflected in the central idea of

supervised ML: approximating an unknown function by learning from labeled training data. This analogy between traditional QC approaches and ML methods naturally suggests the question: Can the advantages of physics-based QC methods and data-driven ML techniques be effectively combined? Note that, unlike many common ML applications (e.g. image recognition), where governing equations are unknown, the ground-truth in QC is given by the Schrödinger equation, which is yet computationally too costly to compute accurately. As shown in the following section, this scenario has motivated the development of physics-informed ML methods, which explicitly encode fundamental principles, such as symmetry invariance or interaction properties, directly into their network architectures. Indeed, recent work increasingly integrates these approaches, leveraging ML algorithms for automated functional development, while explicitly incorporating physical constraints to ensure correct asymptotic behavior and consistency with fundamental principles [166–168]. The integration of physics-based knowledge and statistical ML approaches will be a central theme throughout this thesis. To establish the theoretical foundation, this section briefly outlines essential ML concepts relevant to QC applications.

2.2.1 Fundamentals and Key Concepts

In the following key ML principles are introduced, including model training, parameter optimization via backpropagation, and regularization techniques to reduce overfitting and improve generalization. Generally, regression predicts continuous values, while classification assigns discrete labels to samples. In chemistry, regression is widely used to estimate molecular properties like energies, reaction barriers, or solubilities, whereas classification helps identify toxic compounds, functional groups or charge transfer nature for instance. For brevity and alignment with projects covered in the thesis presented, this theoretical introduction focuses solely on regression tasks. It is worth noting that most classification tasks can essentially be viewed as a form of regression, where continuous predictions are converted into discrete outcomes using a predefined threshold. The fundamental concepts detailed here will be used for the optimization of various supervised methods as explained in subsection 2.2.3.

Quantum Chemical Datasets

Before diving into ML concepts, one needs to take a look at the generation and processing of data. QC datasets form the backbone of ML applications in chemistry and typically consist of collections of molecules, which can represent a sampling of chemical space [169, 170], sets of conformations [171, 172], or reaction data [173, 174]. Typically, besides molecular geometries and element types, QC datasets contain detailed molecular information in the form of physical properties (energies, dipole moments, atomic charges, etc.) or non-physical properties (SMILES identifiers [175–177], stereochemistry information, etc.), which can serve as labels for supervised learning tasks. These properties are either computed using QC calculations (e.g. DFT or SQM methods) or obtained from experimental measurements.

ML models can be tailored to a specific target (e.g. a distinct reaction or a particular class of chemical systems), or designed as general-purpose models capable of handling multiple elements and diverse chemical classes (e.g. ML potentials). Depending on the intended application, the dataset must either accurately represent a homogeneous chemical domain or be sufficiently diverse to capture broader chemical complexity. In both cases, robust ML models typically require substantial amounts of data. While for simpler tasks just a few dozen or hundreds of data points might suffice, more complex tasks generally require beyond thousands of samples to ensure robust and accurate predictions [178].

It is common ML practice, that the available data is partitioned into separate subsets to facilitate

objective model development and evaluation [152]. A common approach is to allocate a large portion of the dataset, often around 80 %, for model training and reserve the remaining 20 % as a hold-out test set, ensuring that performance after training is assessed on unseen data for an unbiased estimate of generalization ability. The training portion is frequently further split to create a validation set, which is used for hyperparameter tuning and to monitor the learning process. It is critical that the test set be completely disjunct from the training set and remain unseen by the model during training. Any inclusion of test data in the training or tuning process results in information leakage, potentially leading to artificially inflated model performance [179]. Moreover, the repeated use of a single test set over the years can lead to overly optimistic evaluations, causing benchmarks to become less reflective of a model's true performance in real-world scenarios [152].

Overfitting occurs when a model captures not only the underlying data patterns but also the noise in the training data, resulting in high performance on the training set but poor generalization to unseen data (i.e. poor performance on the test set). To counteract overfitting, techniques such as regularization can be employed (see below). Furthermore, when the dataset is small, dividing it into a fixed training set and (a small) test set can introduce significant statistical uncertainty in the generalization estimation. In such cases, k-fold cross-validation provides a more robust evaluation method by partitioning the data into k non-overlapping subsets. In each fold, one subset serves as the test set while the remaining data is used for training, and the final error is averaged over all folds [152]. Another statistical model validation method is bootstrap aggregating, where multiple samples are drawn with replacement from the original dataset to form new training sets. For each bootstrap sample, a model is trained and then evaluated on the remaining out-of-bag data. The resulting ensemble of trained models can then be evaluated on the test set. This approach not only provides an estimate of the model's performance but also allows for the computation of confidence intervals. Bootstrapping is particularly useful when dealing with small datasets, as it maximizes the use of available data while mitigating overfitting [152].

Prominent examples of organic datasets include GDB-17 [169], QM9 [170], GEOM [171], and SPICE [180, 181], which contain millions of molecular structures capturing a wide range of chemical properties. In addition, specialized datasets such as tmQM [182] for transition metal complexes, LnQM for lanthanoids [183], and AcQM [184] for actinoids provide targeted insights into specific classes of molecules. Handpicked collections like GMTKN55 [89, 185], although comprising fewer molecules, are highly valued for their chemical relevance and diversity. Other notable datasets include the ANI-1 dataset [157, 186], and the S66x8 dataset [187], which offers detailed insights into intermolecular interactions at a coupled-cluster reference level. Additionally, the MD17 dataset [188], derived from molecular dynamics trajectories, and datasets focusing on conformers and solvent effects in large drug-like molecules [172] further enrich the available resources. This diversity in data not only enables precise analysis but also fosters the development of robust ML models in chemical research.

Loss Functions for Regression Tasks

In supervised learning, the loss function is used to quantify the discrepancy between model predictions and reference values, directly influencing optimization dynamics of gradient-based methods. Its appropriate choice is critical to aligning model performance with scientific objectives. For each input sample x with its associated label $\hat{y}$, a parameterized function f with parameters θ generates a prediction $y = f_\theta(x)$. A general loss function L can then be written as

$$L := L(\mathbf{y}, \hat{\mathbf{y}}) = L(f_\theta(\mathbf{x}), \hat{\mathbf{y}}) \tag{2.58}$$

where θ represents the parameters of the model, $\mathbf{x}$ denotes the input data, $\hat{\mathbf{y}}$ the reference labels, and $\mathbf{y} = f_\theta(\mathbf{x})$ the predicted outputs for all samples in the dataset. In regression tasks common loss functions include mean squared error (MSE) and mean absolute error (MAE), each with distinct properties affecting sensitivity to outliers. The MSE loss corresponds to the L^2 norm of the error vector and is defined as

$$L_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.59}$$

where y_i is the reference value and $\hat{y}_i$ is the predicted value for the i-th sample. MSE measures the average squared error, penalizing larger deviations more strongly due to the quadratic term. However, this makes MSE highly sensitive to outliers, as large errors contribute disproportionately to the loss. MSE is a smooth, differentiable function with respect to the difference between y and $\hat{y}$, which facilitates optimization of the network parameters θ through gradient-based methods (see below). Additionally, MSE corresponds to performing maximum likelihood estimation under the assumption of Gaussian noise [189], further justifying its widespread use in regression. Furthermore, the MAE loss corresponding to the L^1 norm of the error vector is defined as

$$L_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}|\, y_i - \hat{y}_i| \tag{2.60}$$

which represents the average absolute error between predictions and ground truth. Unlike MSE, MAE is less sensitive to large outliers, making MAE more robust in the presence of high-variance or noisy data. Note that minimizing the MAE leads the model to predict the median of the conditional distribution of y given x, in contrast to MSE, which corresponds to the mean [190]. One major drawback of MAE is that it is not differentiable at zero error due to the cusp of the absolute value function, though this is typically handled using subgradient methods or smooth approximations in optimization frameworks [191].
In practice, a variety of loss functions are utilized, each tailored to specific problem characteristics and data distributions [192]. Further examples include the Root Mean Squared Error (RMSE) loss, which provides an error metric in the same units as the target variable and gives an unbiased expectation forecast for asymmetric predictive distributions:

$$L_{\text{RMSE}} = \sqrt{L_{\text{MSE}}} \tag{2.61}$$

The Huber loss function [193] combines the advantages of MSE and MAE, behaving quadratically for small errors and linearly for large ones, making it more robust to outliers. Another example is the Log-Cosh loss [194], which is defined as the logarithm of the hyperbolic cosine of the prediction error. Similar to Huber loss, it smooths the effect of large deviations while remaining differentiable.

Optimization Algorithms and Backpropagation

For the optimization of many parametric models, especially neural networks, the loss function and backpropagation play a central role. It is important to note that non-parametric models are generally trained differently, for example Gaussian processes models typically combine a prior and likelihood function based on the training samples [195]. In the following the optimization of parametric models

will be elaborated, as parametric models play a pronounced role in this thesis. Once a loss function is defined, optimizing a parametric ML model involves finding the model parameters (so-called "weights") that minimize this loss on the training data, a process corresponding to a parameter optimization and in ML referred to as "training". In most cases, an analytical solution to this optimization problem is neither feasible nor possible due to the complexity and high dimensionality of modern ML models. Therefore, numerical optimization techniques are employed to iteratively adjust parameters and minimize the loss. Gradient descent and its variants, such as stochastic gradient descent (SGD) [2] and Adam optimizer [196], are widely used, particularly in neural network training, due to their scalability and efficiency in large parameter spaces. Nevertheless, alternative optimization techniques exist, including Newton's method [197–199], quasi-Newton methods (e.g. L-BFGS [200–204]), and evolutionary algorithms [205, 206], which can be advantageous in specific settings where gradient information is unavailable.
Gradient descent in its simplest form updates the model parameters θ iteratively in the direction of the negative gradient of the loss:

$$\theta \leftarrow \theta - \eta \, \nabla_\theta L(\theta, \mathbf{x}) \tag{2.62}$$

where η is the learning rate controlling the step size, and $L(\theta, \mathbf{x})$ is the loss function evaluated on the training data $\mathbf{x}$. In practice, computing the gradient on the entire training set $\mathbf{x}$ for each update can be very slow when the dataset is large. Therefore, most implementations use SGD or mini-batch gradient descent: at each step, the gradient is estimated from a random subset ("mini-batch") $\mathbf{x}_i$ of the training points [2, 152]. This introduces some noise in the gradient but allows much faster iterations and often helps escape local minima due to the stochastic nature. For SGD the size of the mini-batch amounts to one, i.e. a single sample, whereas for mini-batch gradient descent the batch size is a variable hyperparameter.
Modern optimization algorithms build on basic gradient descent with various enhancements to improve convergence speed, robustness, and adaptability. One such technique is "momentum", which introduces a velocity term that accumulates past gradients as an exponential moving average. This smooths oscillations in the gradient descent trajectory, particularly in narrow, steep valleys of the loss landscape, and often accelerates convergence along consistent gradient directions [152]. The momentum update follows

$$v \leftarrow \beta v + (1 - \beta) \nabla_\theta L(\theta, \mathbf{x}) \, , \quad \theta \leftarrow \theta - \eta \, v \tag{2.63}$$

where β is the momentum coefficient, η is the learning rate, and $\nabla_\theta L(\theta, \mathbf{x})$ is the gradient of the loss function with respect to the parameters θ. An important class of optimizers employs adaptive learning rates, i.e. adjusting the step size for each parameter based on the history of gradient magnitudes. A widely used method in this category is the Adam (adaptive moment estimation) optimizer [196], which combines momentum with adaptive learning rates. Adam maintains exponentially decaying averages of past gradients m_t and squared gradients v_t at a given training iteration t via

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \, g_t \tag{2.64}$$

$$v_t = \beta_2 v_{t-1} \; + (1 - \beta_2) \, g_t^2 \tag{2.65}$$

where $g_t = \nabla_\theta L(\theta_t, \mathbf{x})$ is the current gradient, and β_1, β_2 are decay rates. To correct for initialization

bias, Adam applies bias correction before updating the parameters:

$$\theta \leftarrow \theta - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon} \tag{2.66}$$

where $\epsilon << 1$ prevents division by zero and the weighted moments $\hat{m}_w = m_t/(1-\beta_1^t)$ and $\hat{v}_w = v_t/(1-\beta_2^t)$ compute the bias-corrected moment estimate. These weighted moments are used to counterbalance the initial bias introduced by setting the moment estimates to zero. As the time step t increases, the correction factors converge to $\hat{m}_w \approx m_t$ and $\hat{v}_w \approx v_t$. Since v_t estimates the uncentered variance of the gradient at each step, it can be interpreted as providing an adaptive learning rate $\sim 1/\sqrt{\hat{v}_t}$, resulting in larger(smaller) steps when the gradient fluctuations are small(large). Although Adam does not guarantee convergence for all convex objectives [207], it has become a standard optimizer in deep learning due to its robustness, ability to handle sparse gradients, and adaptability across different parameter scales [208].
Second-order optimization methods, such as Newton's method [197–199] and quasi-Newton techniques like Limited-memory BFGS (L-BFGS) [200–204], leverage curvature information to accelerate convergence. L-BFGS approximates the Hessian using a compact set of vectors, making it feasible for high-dimensional problems without storing the full matrix [204]. While it requires fewer iterations than first-order methods, each step is computationally expensive due to the high computational cost of inverting the Hessian, limiting its use to smaller-scale tasks like fine-tuning small networks. Large-scale deep learning instead favors first-order stochastic optimizers like Adam, which scale much more efficiently. In contrast to gradient-based optimization, evolutionary algorithms (EAs) take a population-based approach, inspired by natural selection [205, 206]. These methods iteratively evolve candidate solutions using genetic operators such as mutation, crossover, and selection. Evolutionary strategies and genetic algorithms are particularly useful in non-convex, high-dimensional, or black-box optimization problems where gradients are unavailable or impractical to compute. While EAs are often less sample-efficient than gradient-based methods, they excel in optimization landscapes with multiple local optima and are applied in hyperparameter tuning, reinforcement learning, and neural architecture search [209].

The basis for gradient-based optimization in neural networks is the backpropagation algorithm [210], which efficiently applies the chain rule of calculus to systematically compute gradients of the loss function with respect to each weight within the network. By propagating prediction errors backward through the network layers, backpropagation enables efficient parameter adjustments for arbitrary depth of the network, progressively minimizing the loss [152]. One notable advantage of backpropagation lies in the reuse of all computed partial derivatives, eliminating the need to recalculate the entire chain of derivatives from scratch for each gradient. Despite its effectiveness, particularly in supervised learning tasks, gradient-based optimization encounters challenges such as the vanishing and exploding gradient problems in deep neural networks [211–213]. To mitigate these issues, specialized architectures and optimization techniques, including residual networks [214, 215] and normalization methods (e.g. batch and layer normalization) [211, 216], have been developed. Importantly, backpropagation itself is not an optimization algorithm but a procedure to compute gradients, which are subsequently used by optimizers such as SGD or Adam to update network weights. Without backpropagation or a similar automatic differentiation technique, training deep neural networks with millions of parameters would be impractical [152].

Regularization Techniques

A constant concern in ML is overfitting, whereby a model captures random noise or specific idiosyncrasies in the training data instead of the underlying general patterns. Overfitting reduces the model's ability to generalize to unseen data, thus necessitating regularization techniques, which impose constraints or penalties on model complexity. Regularization often involves adding penalty terms to the loss function L, resulting in a modified cost function L_{reg} that is minimized during model training. Thereby, the hyperparameter λ controls the strength of the regularization penalty. Two common approaches are lasso and ridge regularization. In lasso regression (L1), the absolute values of the weights are penalized [217, 218]:

$$L_{\text{reg}} = L + \lambda \sum_j |w_j| \tag{2.67}$$

The L1 penalty induces sparsity, potentially driving individual weights to zero and effectively selecting relevant features, beneficial for identifying critical molecular descriptors in chemical applications. In ridge regression (L2), the squared Euclidean norm of the weight vector $\mathbf{w}$ is penalized [219–221]:

$$L_{\text{reg}} = L + \lambda \sum_j w_j^2 \tag{2.68}$$

This penalty also mitigates overfitting by controlling model complexity. In comparison to L1 regularization, L2 regularization is not sparse, driving weights only close to zero.

Further regularization techniques include early stopping, dropout and data augmentation: In iterative training methods, it can be observed that both training and validation error initially decrease together, but after a certain number of iterations, the validation error begins to rise again while the training error continues to decline. In those cases, early stopping terminates the training process once the performance on a validation set begins to deteriorate, hence reducing overfitting [152]. Dropout randomly sets the weight of a fraction of neurons to zero during each training iteration, preventing individual neurons from relying excessively on specific noise patterns [222–224]. At inference time, all neurons are utilized again. To compensate for dropout during training, the outgoing weights of a unit that was kept with probability p are scaled by p, ensuring that the expected activation remains consistent [223]. Data augmentation increases dataset diversity by applying transformations or perturbations, such as small variations in molecular geometries or alternative molecular representations, thus improving the model's invariance and generalization capability. Examples include augmentation of SMILES string representation [175–177] to enhance ML model performance [225, 226]. Each of these aforementioned regularization techniques can be used individually or in combination. In practice, the choice of regularization and its strength (λ or dropout rate, etc.) is often tuned via cross-validation to find the best trade-off between training and validation error [152].

2.2.2 Molecular Representations and Descriptors

An important step in applying ML to QC is choosing how to represent a molecule in a computer-interpretable way. This representation, often referred to as a "molecular descriptor" or "feature vector", must capture the salient information about the molecule's structure and composition that determines the property of interest. At the same time, it should satisfy symmetries with respect to permutations, rotations and translations of atomic coordinates. Note that most scalar properties (e.g.

energy) are invariant to the aforementioned transformations, while vector properties (e.g. dipole moments) generally transform covariant under these transformations. Furthermore, for extensive properties, descriptors must exhibit size extensivity, i.e. combined non-interacting subsystems should yield descriptors and predictions reflecting additive contributions. Molecular representations that do not fulfill these criteria might lead to unphysical behavior. For an overview over different approaches, in this section common traditional handcrafted descriptors and graph representations are introduced.

Handcrafted Molecular Representations

Early work of ML in QC relied on explicit, fixed-length descriptors of molecules. These descriptors are mathematical quantities that are derived from the molecular structure including atoms and their positions and are often designed to be invariant to certain transformations such as rotation of the molecule.

One of the first successful representations of molecules is the Coulomb matrix [155]. For a molecule with N atoms, the Coulomb matrix C is an $N \times N$ matrix defined by

$$C_{ij} = \begin{cases} \frac{Z_i Z_j}{\|R_i - R_j\|} & i \neq j \\ \frac{1}{2} Z_i^{2.4} & i = j \end{cases} \tag{2.69}$$

where Z_i is the atomic number of atom i and R_i its Cartesian coordinates. Off-diagonal elements represent the Coulomb repulsion between atoms i and j, while diagonal elements are an arbitrary, fixed formula related to the atomic number – originally $\frac{1}{2} Z_i^{2.4}$ was proposed as a heuristic to encode a notion of atom self-energy. The Coulomb matrix is symmetric and, in principle, contains enough information to reconstruct the full set of interatomic distances and atom types. However, one problem is that the matrix depends on the ordering of atoms via the indices i, j. Permuting atom labels will thus lead to a different matrix and thereby complicating a unique representation for a given system. To counter this, different sorting strategies can be applied to the entries in the Coulomb matrix (e.g. by atomic charge). However, this introduces a degree of discontinuity, whereby small changes in geometry might change the sorting order of atoms.

To improve upon the Coulomb matrix, the Bag-of-Bonds (BoB) representation was developed [227] by eliminating the need to sort whole matrices. The idea is to treat each type of pairwise interaction separately. Thereby, all off-diagonal Coulomb matrix elements are grouped into "bags" according to the pair of atom types involved (e.g. C-H, H-H, C-N, ...). Within each bag, the Coulomb terms are sorted and padded with zeros to a fixed bag size. Finally, the sorted values from all bags are concatenated into one long vector. The BoB representation is invariant to atom index permutations but typically yields a higher-dimensional representation than a Coulomb matrix for the same molecule, due to being effectively a sparse expansion of the Coulomb matrix entries.

Instead of a global descriptor for the whole molecule, Atom-Centered Symmetry Functions (ACSF) [228] are a set of local descriptors computed around each atom. ACSFs encode the local chemical environment of each atom in a way that is invariant to permutations of neighboring atoms and to overall translation and rotation of the molecule. Typically, a set of radial and angular functions are defined. For example Behler (2011) suggests, a radial symmetry function for atom i

$$G_i^{\text{rad}} = \sum_j e^{-\eta (R_{ij} - R_s)^2} f_c(R_{ij}) \,, \tag{2.70}$$

where R_{ij} is the distance between atom i and j, η and R_s are parameters , and $f_c(R)$ is a cutoff function that smoothly goes to 0 at some cutoff radius R_c, so that only neighbors within R_c contribute, ensuring locality. This captures a smoothed histogram of neighbor distances around atom i. To enhance local resolution, multiple radial symmetry function can be constructed for different choices of η, R_s. Angular symmetry functions consider triplets of atoms [228]

$$G_i^{\text{ang}} = 2^{1-\zeta} \sum_j \sum_{k \neq i} (1 + \lambda \cos\theta_{ijk})^{\zeta} \, e^{-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)} f_c(R_{ij}) f_c(R_{ik}) f_c(R_{jk}) \tag{2.71}$$

with parameters ζ, λ, η. Here θ_{ijk} is the angle at atom i subtended by neighbors j and k. This function accumulates contributions from each pair of neighbors (j, i) and (k, i), encoding both their distances to i and the angle between them. By using various parameter sets $(\eta, R_s, \zeta, \lambda)$, one generates a vector of symmetry function values for each atom that uniquely describes the environment of that atom up to the cutoff radius. The ACSFs are invariant to swapping neighbors j and k due to summation and to rotation/translation due to their sole dependence on interatomic distances and angles. Despite their advantages in accurately capturing the local atomic environment, ACSFs become computationally expensive as chemical complexity increases due to the large number of symmetry functions required. Additionally, while they explicitly encode local geometry up to three-body correlations, ACSFs necessitate careful function selection and rely on a fixed cutoff, meaning that long-range interactions beyond this threshold are neglected unless addressed separately.
Further molecular descriptors include: The many-body tensor representation which generalizes the Coulomb matrix to include higher-body terms [229]. The smooth overlap of atomic positions (SOAP) uses a rotationally invariant power spectrum of atomic neighbor density expansions [230]. Classical cheminformatics fingerprints such as Morgan fingerprints [231, 232] encode substructure patterns as a binary vector in which each bit indicates the presence of a particular chemical fragment up to a certain radius. These fingerprints are invariant to atom reordering and are extremely efficient to compute, yet they lose geometric information because they typically consider only connectivity.

Graph-Based Molecular Representations

An efficient way to represent molecules is as graphs [233]. In a molecular graph [234], each node corresponds to an atom, and each edge to a chemical bond connecting the corresponding atoms. Optionally, graph edges can also connect non-bonded atoms, e.g. by using a distance cutoff to connect atoms within a certain radius, thereby incorporating geometric proximity beyond formal bonds. Furthermore, features can be assigned to nodes, edges or even the entire graph. In that case, each node holds a feature vector containing information such as an encoding of the element type as well as other atomic properties (e.g. atomic partial charge, hybridization state, electronegativity). Edges can also store information, such as bond order or distance-based metrics.

An advantage of graph-based molecular representations lies in their intrinsic ability to handle inputs of variable size. Molecules with arbitrary numbers of atoms directly map to graphs of corresponding topology. Furthermore, graphs explicitly encode locality via adjacency, naturally capturing the molecular environment for each atom. Additionally, graph representations facilitate interpretability by aligning with chemical intuition, offering a direct correspondence between nodes and atoms, as well as functional groups and subgraphs. In comparison to handcrafted descriptors, little domain knowledge needs to be embedded explicitly in graph-based approaches as features are “learned” by the model itself during training [235, 236].

Another advantage of graph-based molecular representations lies in their inherent adherence to symmetry. Thereby, representing molecules as graphs naturally respects key physical invariances. Permuting node indices does not alter the abstract graph structure, since a graph is formally defined by its connectivity pattern rather than by specific node labels. Consequently, algorithms, which operate on connectivity and node features, inherently treat isomorphic graphs equivalently, independent of atom labeling [237]. Furthermore, pure graph representations based solely on connectivity are inherently invariant to translations and rotations, as they exclude absolute spatial information. However, this invariance introduces a notable limitation: connectivity-only graphs cannot distinguish between different spatial arrangements, such as stereoisomers. Therefore, explicit inclusion of geometric information, typically via distance-based node or edge features, is necessary to enable graph-based methods to perform tasks like conformational searches effectively. In such cases, modern models predominantly utilize relative spatial information (e.g. interatomic distances or angles) or employ specialized equivariant architectures to maintain symmetries [238]. Examples include equivariant networks such as NequIP [239] and MACE [240], which in-detail go beyond the scope of this theoretical overview but represent powerful tools for incorporating physical symmetries into learned representations.

It is important to note that representation quality often directly correlates with model performance: A powerful learning algorithm cannot fully compensate for a poor choice of representation that does not reflect important symmetries or loses crucial information. Conversely, a well-chosen representation can in-theory simplify the task so much that even a simple model would achieve excellent results [152, 241]. The development of representations and models has gone hand-in-hand, leading to the next section on how supervised learning models are built on top of these representations.

2.2.3 Supervised Learning Approaches in Quantum Chemistry

Having established how to represent molecular data for ML, this section turns to the various supervised learning models that can be used to map those representations to desired properties. Supervised models in QC span from simple regression techniques to complex deep learning architectures. The following section briefly surveys the landscape of these approaches, starting with classical regression methods and then focusing on neural networks, which are of key relevance to this work.

Traditional Regression Models

Early applications of ML in chemistry often utilized relatively simple regression algorithms on top of handcrafted molecular descriptors. These methods are grounded in statistical learning theory [242] and usually offer advantages in terms of interpretability and training on smaller datasets, at the cost of potentially lower asymptotic performance compared to large neural networks trained on huge datasets.

The arguably most simple regression model is linear regression [189]. This assumes the property y is a linear combination of the features x_j: $\hat{y} = w_0 + \sum_j w_j x_j$. Training involves solving for weights w_j that minimize loss on the training data. While a linear model is too simplistic for most QC properties across diverse chemistry, it can be useful in limited contexts such as correlating a property for a series of similar molecules. Often linear regression in combination with lasso and ridge regularization (c.f. subsection 2.2.1) is applied in the context of sparse data availability or for feature selection (i.e. selecting most relevant descriptors from a larger pool). Despite being limited to linear correlations, its interpretability makes linear regression a great supplement to many statistical analyses.

Classical linear models such as ridge regression often struggle to capture nonlinear relationships effectively. A common strategy to address this is transforming input data x into a higher-dimensional feature space via $x \mapsto \phi(x)$, where complex nonlinear patterns become linearly representable. However, explicitly performing this transformation $\phi(x)$ is typically expensive or impractical.

Kernel ridge regression (KRR) [243, 244] is a nonlinear extension of ridge regression using the "kernel trick". Instead of solving the regression in the input space, one implicitly maps x to a high-dimensional feature space via a kernel function $k(x, x') = \phi(x)^\top \phi(x')$ that computes scalar products in the feature space without the need of explicit information on transformation ϕ. Solving the the regression problem in the feature space using the kernel, enables KRR to model complex nonlinear relationships while retaining computational efficiency. Kernel functions are designed such that they quantify the similarity between data points based on their descriptors. Common kernels in chemistry include Gaussian kernels [155]:

$$k(x, x') = \exp\Big(- \frac{\|x - x'\|^2}{2\sigma^2} \Big) \tag{2.72}$$

KRR features uncertainty estimates and a closed form solution which is advantageous for small to medium-sized datasets regarding computing time. However, KRR scales poorly with the number of training points N, due to solving the linear system requiring $O(N^3)$ operations. Thus, for large datasets other regression methods (e.g. neural networks) become more feasible.

Gaussian process regression (GPR) [245] is closely related to KRR. Both KRR and GPR leverage the kernel trick to enhance their expressiveness, allowing them to better fit the training data. While KRR identifies a single target function by minimizing a loss function, GPR adopts a probabilistic framework. In GPR, Bayes' theorem [246] is used to construct a Gaussian posterior distribution over target functions, effectively merging prior beliefs with a likelihood function derived from the observed data to estimate the posterior distribution. Indeed, given appropriate hyperparameters, the mean of this posterior with observation noise is the KRR predictor. GPR intrinsically provides a prediction uncertainty, making it useful for guiding experiments or active learning approaches [247, 248]. Similar to KRR, the scaling of $O(N^3)$ limits GPR to a small datasets unless sparse GPs or approximations are used [249, 250].

Generally, the presented regression models require choosing a suitable kernel or assuming linearity, which means they rely heavily on the quality of the chosen descriptors. If the descriptors lead to the target quantity being relatively smooth or linear in some transformed space, these regression methods can produce excellent results even with limited data. However, for more complex tasks where the choice of descriptors is not straight-forward, neural networks as presented in the next section pose a viable alternative.

Neural Networks

Neural networks (NNs) are a class of regression models constructed in a layer-wise fashion, which enables them to capture complex, non-linear relationships within data. Supported by the Universal Approximation Theorem, even a single hidden layer neural network can approximate any continuous function on a compact domain to an arbitrary degree of accuracy, given enough hidden units [251, 252]. As a result, NNs have emerged as the method of choice for addressing a wide range of challenges, from standard pattern recognition [253] to advanced applications in QC [254–256]. Their scalability

and flexibility allow them to effectively model intricate relationships in data that are often intractable by traditional methods. In the following, ML architectures most relevant to this thesis are elucidated, starting with fully-connected feedforward neural networks and progressing to graph neural networks in the next section, which are particularly pertinent for applications in QC.

Multi-Layer Perceptrons (MLPs) [257] are fully-connected feedforward networks that transform a fixed-length input vector (i.e. a molecular descriptor) through successive layers to capture complex nonlinear relationships. Each layer performs a linear transformation followed by a nonlinear activation function σ. A single layer is generally defined by

$$\mathbf{h}_i = \sigma(W_i \mathbf{h}_{i-1} + \mathbf{b}_i) \tag{2.73}$$

with W_i being the weight matrix and $\mathbf{b}_i$ the bias vector in the given layer. By sequentially propagating vector $\mathbf{h}_i$ through each layer, information is processed. Layers other than the input and output layers are commonly referred to as "hidden layer". Specifically, the input vector $\mathbf{x}$ is fed into the network such that $\mathbf{h}_0 = \mathbf{x}$, and the final layer outputs the predicted quantity $\mathbf{h}_{\text{last}} = \hat{\mathbf{y}}$. As detailed in subsection 2.2.2 the input vector $\mathbf{x}$ can constitute molecular descriptors such as Coulomb matrix eigenvalues. Common activation function include rectified linear unit (ReLU) and sigmoidal functions such as logistic and hyperbolic tangent:

$$\sigma_{\text{ReLU}}(x) = \max(0, x) \tag{2.74}$$

$$\sigma_{\text{logistic}}(x) = \frac{1}{1 + e^{-x}} \tag{2.75}$$

$$\sigma_{\tanh}(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.76}$$

Typically, the same activation function is used throughout all hidden layers. Depending on the research question, the activation function for the output layer is adjusted, e.g. when predicting a continuous scalar variable $y \in [-\infty, +\infty]$ the activation function in the last layer is usually omitted. Optimization of the network parameters (W_i and $\mathbf{b}_i$) is usually performed using backpropagation in combination with adaptive methods such as Adam (c.f. subsection 2.2.1). Figure 2.3 illustrates a fully connected feedforward NN with an input vector of dimension 3, a single hidden layer, and a scalar output.

The architecture of a MLP is highly flexible, with no predefined constraints on the number of hidden layers or the size of each individual layer, i.e. the dimensionality of W_i, $\mathbf{b}_i$, and consequently $\mathbf{h}_{i+1}$. The predictive accuracy of NN models often depends not only on the availability of data but also on the chosen architecture. Deep models have gained prominence for their ability to automatically learn hierarchical feature representations from raw data [152]. Unlike shallow architectures, deep NNs stack multiple hidden layers, allowing them to extract increasingly abstract features. This automated feature learning reduces the reliance on manual feature engineering and often leads to improved predictive performance [257]. However, the training of deep models can be challenging due to issues like vanishing or exploding gradients [213] (see also subsection 2.2.1). For example, when the outputs of multiple sigmoid activations are very small, the gradient associated with these parameters can become nearly zero. To mitigate these problems, modern techniques* such as batch normalization [211], residual connections [214] or gradient clipping [258] are employed. These innovations have not only enhanced training stability but have also propelled deep learning to achieve state-of-the-art

* Probably, Jürgen Schmidhuber had invented them all already back in 10 000 BC.

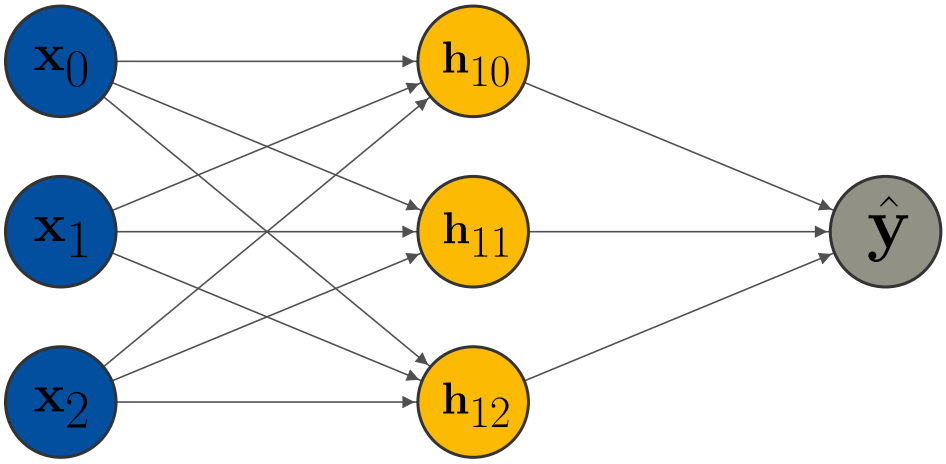

Figure 2.3: A fully connected feedforward neural network with an exemplary input vector $\mathbf{x}$ (blue), one hidden layer (yellow), and a scalar output (gray). The arrows represent the flow of information, where weights W_{ij} and corresponding bias terms b_{ij} are applied, followed by an activation function σ. For visual clarity, these terms are not explicitly shown on every arrow.

results across a wide range of applications, for instance in computational chemistry [259].

Further examples of the application of NNs in QC include the development of NN potentials by Behler et al. (2007), which employ MLPs to represent high-dimensional potential energy surfaces of complex molecular systems [6]. In this approach, the total energy of a system is partitioned into atomic contributions, applying a separate MLP associated with each element type, that computes an atom-specific embedding. This is based on the locality assumption, as an atom's contribution to the total energy is primarily determined by its local environment. Hence, many´ NN-based models compute the total energy as a sum of atomic contributions. Moreover, ACSF-based descriptor models have been proposed [228], taking the ACSF vector of an atom as input for these NNs (c.f. subsection 2.2.2). Another notable example is ANI-1 [157], a NN potential designed to predict molecular energies with near-DFT accuracy at the computational cost of classical force fields. It employs a modified version of Behler-Parrinello symmetry functions to construct atomic environment vectors, which serve as molecular representations that capture both configurational and conformational degrees of freedom. For the generation of training data, ANI-1 utilizes Normal Mode Sampling to produce a diverse set of molecular conformations, thereby enhancing its transferability. Additionally, PhysNet [260] is a high-dimensional NN designed to predict properties such as energies, forces, dipole moments, and partial charges. It applies MLPs that leverage learnable distance-based attention masks to model atomic interactions in a physically meaningful way. By integrating residual blocks and explicit electrostatic terms, PhysNet effectively captures both short- and long-range interactions, making it suitable for a wide range of chemical systems.

2.2.4 Graph Neural Networks

Graph Neural Networks (GNNs) [261–263] represent a significant advancement in molecular ML by providing a native representation of molecular structure (c.f. subsection 2.2.2). Unlike traditional NNs, like MLPs, which require fixed-size input vectors, GNNs can process molecular graphs of

arbitrary size, enabling predictions at the node, edge, or global graph level. In a message passing scheme each atom is initially assigned a feature vector, which is then refined through multiple graph layers with iterative message passing. This native integration of molecular descriptors renders GNNs one of the most universally applicable and high-performing architectures, thus playing a central role in state-of-the-art models for QC [4, 264, 265]. Thereby, a GNN can be trained end-to-end using techniques such as gradient descent to minimize the loss on the target property (c.f. subsection 2.2.1). In the following a detailed look at GNNs in the context of QC is given, including their mechanisms, advantages, and some specific architectures and challenges.

Commonly, a molecule is represented as a graph $G = (V, E)$, where each node $i \in V$ corresponds to an atom and each edge $(i, j) \in E$ signifies a bond or interaction between atoms i and j [263]. The graph is further augmented with node features $h_i^{(0)}$ for each atom, which encode the atomic number – typically represented as a one-hot vector over element types or as an integer embedding – and may also include additional properties such as atom type, formal charge or hybridization. Similarly, edge features e_{ij} capture characteristics of each bond, such as bond type (e.g. single, double, triple, aromatic), ring participation, or, in the context of 3D graph networks, spatial information like the vector or distance between atoms [235]. In fully connected graph representations every pair of atoms is linked by an edge defined by the interatomic distance r_{ij}, with those beyond a certain cutoff typically being disregarded to manage complexity. In molecular graph representations, rather than constructing a fully connected graph where every pair of atoms is linked by an edge defined by the interatomic distance r_{ij}, a cutoff is introduced to retain only local connections. This approach yields a graph that is not fully connected overall, although the resulting subgraphs within the cutoff radius are densely connected. The ultimate objective is to utilize these inputs and topology to generate predictions of physical quantities. For molecular properties that apply to the entire molecule, information from all atoms is aggregated, whereas for atom-specific outputs, such as atomic charges or forces, the network directly provides per-node predictions.

The core mechanism of many GNNs is iterative message passing. At iteration t, each node i has a state vector (embedding) $h_i^{(t)}$. Initially $h_i^{(0)}$ is the feature embedding of atom i, usually a learned vector associated with its element type. At each iteration a "message" is sent from node j to neighbor i:

$$m_{ij}^{(t)} = M^{(t)}\left(h_i^{(t)}, h_j^{(t)}, e_{ij}\right) \tag{2.77}$$

The function $M^{(t)}$ for generating messages could be the same at every layer or different. Sometimes, messages are computed only using the neighbor's state $h_j^{(t)}$ and edge features e_{ij} – hence not using $h_i^{(t)}$ explicitly as this information is already processed using the update function $U^{(t)}$ (see below) [261, 263]. After message generation node i aggregates all incoming messages:

$$m_i^{(t)} = \sum_{j \in \mathcal{N}(i)} m_{ij}^{(t)} \tag{2.78}$$

where $\mathcal{N}(i)$ denotes neighbors of i. Note that the sum (or average) is invariant with respect to permutations of neighbors. Subsequent to message aggregation the node state is updated using an update function $U^{(t)}$

$$h_i^{(t+1)} = U^{(t)}\left(h_i^{(t)}, m_i^{(t)}\right) \tag{2.79}$$

where $U^{(t)}$ is typically an MLP that produces the new embedding $h_i^{(t+1)}$ from the previous embedding $h_i^{(t)}$ and the aggregated message $m_i^{(t)}$. Since embeddings and aggregated messages have the same fixed size for all nodes, fixed-size MLPs can be utilized. The size for these internal representations are hyperparameters which are part of the architectural design of the model. Hence, the input/output dimensionality for the employed MLPs is independent of the graph size and thus the same model can be applied to molecules of different sizes.
Repeating this messing passing for several iterations allows information to propagate through the graph. After T iterations, each node's embedding $h_i^{(T)}$ contains information about the node and its environment up to T edges away. This naturally defines a receptive field for each node, i.e. the part of the graph, whose information influences the given node. At that point, for a graph-level target, a readout function R is applied, "pooling" the information of the graph

$$\hat{y} = R(\{h_i^{(T)} \mid i \in V\}) \tag{2.80}$$

which must be invariant to node ordering. A simple and common R is summation: $\hat{y} = \sum_i f_{\text{out}}(h_i^{(T)})$, where f_{out} might be a small network to map node embeddings to contributions. This works especially well for extensive properties like energy, where the total energy is expected to be the sum of atomic contributions [235, 240, 266–268]. Indeed, GNNs can be seen as an extension of the idea behind Behler-Parrinello networks but with learned message functions instead of fixed symmetry functions (c.f. [6]). Nevertheless, even for intensive properties, if the network can learn to distribute the intensive property appropriately among atoms, simple node summation could yield reasonable results. Though generally for intensive properties, more complex pooling is used [235, 269–271]. A schematic illustration of a GNN as molecular representation is given in Figure 2.4.

By stacking layers, GNNs can approximate many-body interactions, such as three-body and four-body interactions, which can be associated with bond and torsion angles, respectively. Thus, increasing the number of layers expands the receptive field and facilitates the integration of many-body effects. Some architectures, like DimeNet, even explicitly incorporate angular terms in the message function rather than relying solely on multiple layers [266, 267].
One limitation of message passing is its inherent locality, which does not explicitly address long-range interactions. Many molecular properties, however, are affected by interactions beyond nearest neighbors. For example, in conjugated systems, electron delocalization extends over multiple bonds, and in proteins or large molecules, non-bonded interactions such as electrostatics and dispersion operate over long distances. To overcome this limitation, several strategies have been proposed. One approach is to increase the number of message passing steps T, so that, in theory, every node can influence every other within T hops. However, a large T can complicate training due to issues like oversmoothing, where repeated averaging causes node embeddings to become too similar [272], and oversquashing, where the influence between nodes decreases exponentially with the number of hops between them [273, 274]. Another strategy is to incorporate distance-based edges beyond covalent bonds, i.e. connecting all atom pairs within a specified cutoff radius, thereby directly capturing long-range non-bonded interactions, such as intramolecular hydrogen bonds or interactions between spatially adjacent regions [238, 266, 267]. Additionally, specialized architectural elements, such as attention mechanisms, gating functions or Ewald-based message passing, can facilitate the propagation of indirect influences from far-away nodes by enabling gradients to flow globally [275, 276]. Finally, multi-scale or hierarchical GNNs can coarsen the graph by merging groups of atoms into super-nodes [277].

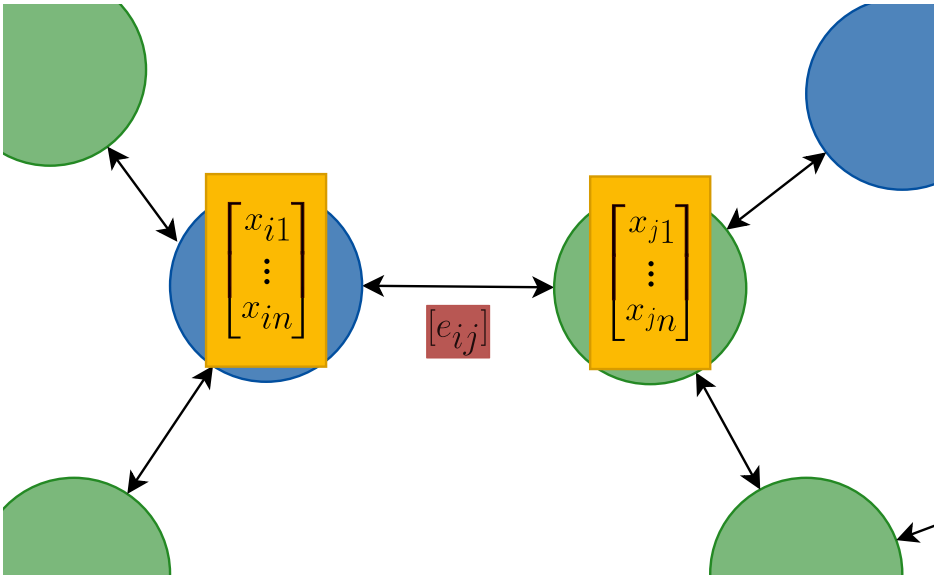

Figure 2.4: Illustration of a graph-based molecular representation using a GNN. The molecular topology is represented by a graph whose nodes (blue and green) correspond to atoms, and whose arrows denote covalent bonds. The node embeddings (yellow) and edge attributes (red) encode the local chemical environment. Messages are generated from these features and passed between the nodes to propagate information. After multiple iterations of message passing, a global output quantity can be computed via pooling over all node vectors.

Multiple GNN architectures for applications in QC have emerged as powerful tools to predict molecular properties with high accuracy. For example, SchNet [238] employs continuous filter convolution, where each pair of atoms within a specified cutoff contributes to the message via a radial basis expansion of the interatomic distance multiplied by a learned filter. Building upon SchNet, PhysNet [260] incorporates explicit terms for long-range electrostatics and extends the framework to predict atomic charges and dipoles. It introduces mechanisms to model physical atomic charge interactions and to ensure energy conservation in MD simulations. Another notable architecture is DimeNet/DimeNet++ [266, 267], which explicitly incorporates directional (angular) information during message passing by using a triplet of atoms to compute an angular basis function that modulates the messages. This design enables DimeNet to achieve very high accuracy particularly for properties sensitive to angular variations. More advanced recent architectures, such as EGNN [278], GemNet [268], PaiNN [279], NequIP [239], and MACE [240], have focused on equivariance with respect to geometric transformations. For instance, PaiNN [279] and similar equivariant GNNs ensure that rotations of the input coordinates lead to corresponding rotations in the predicted vectorial properties (e.g. forces or dipoles). NequIP [239] and MACE [240] further utilize $SO(3)$ group representations, such as spherical harmonics, to systematically capture directional geometric information, achieving state-of-the-art performance on force field predictions with substantially reduced data requirements.

As outlined above, GNNs offer several significant advantages for tasks in QC, i.e. a flexible yet structured approach that aligns well with molecular structure. They effectively overcome many earlier limitations, like fixed-size input, and thus allow the model complexity to be focused on learning chemistry rather than symmetries. GNNs naturally allow for constructing predictions that are

invariant with respect to permutations of atoms, eliminating the need for explicit sorting or additional preprocessing to fulfill symmetry requirements. Furthermore, GNNs allow for efficient encoding of local chemical information by directly leveraging the bonded connectivity and functional group structures inherent in molecular graphs. Unlike traditional methods that rely on MLPs or kernel approaches to indirectly infer connectivity from descriptors, GNNs operate directly on the graph structure. In addition, many GNN architectures are inherently extensible, as they capture how a property scales with molecular size. For instance, adding another CH_2 group to a chain can typically be handled without retraining the model, with the network predicting an incremental change in the property. Empirically, GNNs have achieved state-of-the-art accuracy on numerous benchmarks, often outperforming fixed-descriptor methods as data abundance increases [7, 8, 240, 267].

Despite these advantages, several challenges and limitations remain. GNNs can require large amounts of data to avoid overfitting, especially when deep architectures with many parameters are used [152, 263]. To mitigate this, techniques such as data augmentation [280], e.g. through rotations of moieties or small perturbations to atomic positions, and transfer learning via pretraining on large datasets have been employed [281]. While message passing scales linearly with the number of edges $\mathcal{O}(|E|)$, the inclusion of dense graphs (e.g. by connecting all atom pairs within a large cutoff) can lead to quadratic scaling $\sim \mathcal{O}(|V|^2)$, which becomes problematic for large molecules, such as proteins, unless partitioning or sparse representations are utilized. Furthermore, although attention-based GNNs can provide insights by highlighting influential bonds or substructures, exact interpretability remains daring [275]. Notably, even with extensive training datasets, GNNs remain stochastic methods that primarily interpolate within the chemical space defined by the training data. In general, guaranteeing or even assessing their extrapolation capabilities is challenging. For instance, a model trained on small organic molecules might not generalize well to larger, more complex drug-like molecules if it encounters novel patterns or scales beyond those represented in its training set. Additionally, GNNs typically operate on static graphs, which limits their ability to model dynamic chemical processes such as reactions or bond-breaking events. Although some approaches have been developed to handle dynamic graphs or predict bond rearrangements [282, 283], capturing continuous bond transformations remains difficult. Furthermore, integrating GNNs with established physical principles is an ongoing research area. Hybrid physics-ML models [284, 285], the prediction of Hamiltonian matrix elements [286, 287], or corrections to physics-based simulations [288, 289] are under active development. Eventually ensuring that the outputs of GNNs (and other ML methods) adhere to physical constraints, such as energy conservation and symmetry requirements (e.g. through equivariant architectures), is crucial for advancing their applications in QC.

CHAPTER 3

Hybrid DFT Geometries and Properties for 17k Lanthanoid Complexes – The LnQM Dataset

Christian Hölzer,[*] Igor Gordiy,[†] Stefan Grimme,[*] and Markus Bursch[‡]

Received 14 November 2023, Published online 18 January 2024.

Reprinted in Appendix A (adapted) with permission[§] from
C. Hölzer, I. Gordiy, S. Grimme and M. Bursch, *Hybrid DFT Geometries and Properties for 17k Lanthanoid Complexes – The LnQM Data Set*, J. Chem. Inf. Model. **64** (2024) 825, DOI: 10.1021/acs.jcim.3c01832.

Own manuscript contributions

- sample generation based on CSD database
- performing calculations
- interpretation of the results
- writing of the manuscript

[*] Mulliken Center for Theoretical Chemistry, University of Bonn, Beringstr. 4, 53115 Bonn, Germany
[†] Department of Chemistry and Applied Biosciences, ETH Zürich, Vladimir-Prelog-Weg 2, Zürich 8093, Switzerland
[‡] Max-Planck-Institut für Kohlenforschung, Kaiser-Wilhelm-Platz 1, 45470 Mülheim an der Ruhr, Germany

Lanthanoids play a central role in modern technologies and applications, from display technology [290–292] and medical imaging [293, 294] to catalysis [295] and energy production [296]. However, despite their importance, there is a need of comprehensive lanthanoids datasets that can support both theoretical quantum chemical investigations as well as data-driven approaches such as machine learning.

The LnQM dataset, introduced in this study, addresses this gap by providing detailed information on 17 269 mononuclear lanthanoid complexes, including their geometrical, electronic, and energetic properties. A key feature of the LnQM dataset is the systematic permutation of the central lanthanoid for each ligand motif, ensuring comparability of different lanthanoid complexes. Due to its large effective ionic radius, lanthanum is taken as a starting point in the +3 oxidation state. The permutation of other lanthanoids allowed the isolation of specific chemical influences and trends within the lanthanoid series. Thereby, the dataset focuses entirely on lanthanoid complexes in the +3 oxidation state combined with a variety of ligands. In total, 31 distinct ligands were included, showcasing a wide range of chemical variability. These ligands range from simple molecules like water to more complex structures such as bipiperidine. The molecular distribution covers molecular charges from -1 to +3 and molecules comprising 10 to 87 atoms. Whereby the majority of samples in the dataset are either neutral or carry a charge of ±1. The overall dataset composition is well balanced between all lanthanoids. To ensure the validity of each complex created, various criteria are applied. These constraints range from convergence of quantum chemical calculations to confirming structural integrity. The latter include limiting heavy-atom RMSD deviations from the initial lanthanum structures, discarding strongly distorted complexes or enforcing a minimum coordination number, among others. Generally, a high convergence rate could be observed, out of 18 075 investigated structures 96% passed the selection process and could be used for the final dataset. Geometrical optimizations were performed at the PBE0-D4/def2-SVP level [45, 68, 82, 91, 92, 297–299], while single-point calculations at the ωB97M-V/def2-SVPD level [45, 99, 298–300] level determined the energetic properties. To correct for London-dispersion, the D4 dispersion correction [301, 302] is applied.

The dataset further includes chemical data such as bond lengths, atomic partial charges, dipole moments, coordination numbers, and the HOMO-LUMO gap. The aforementioned diversity enables the dataset to facilitate a detailed comparison of lanthanoid properties independent of ligand effects, which is crucial for analyzing chemical and physical trends. For instance, the influence of lanthanoid contraction becomes apparent in bond lengths and HOMO-LUMO energy differences, with the HOMO-LUMO gap varying by up to 2 eV across the lanthanoid series, peaking, as expected, for europium and gadolinium. Another key focus of the dataset is the analysis and validation of different quantum chemical methods for calculating atomic partial charges. In comparison to the orbital-independent Hirshfeld [303] partition scheme, the Charge Extended Hückel [304] model proved particularly reliable, whereas orbital-dependent models, such as Mulliken [305] and Löwdin [306], showed larger deviations, partially due to the diffuse basis set. Additionally, the performance of force field and semiempirical methods like GFN-FF [16] and GFN2-xTB [18, 148] on lanthanoid complexes were evaluated. While these methods provide overall reasonable geometrical structures, they exhibit weaknesses in predicting bond lengths. These limitations can be addressed by specific parameterizations for lanthanoids, which are now more feasible with the LnQM dataset.

Overall, the LnQM dataset provides a robust foundation for the advancement of quantum chemical methods and data-driven analyses on lanthanoids. Its open accessibility allows researchers to explore new research avenues, such as extending the dataset with additional ligands or incorporating aforementioned spin-orbit couplings. Future studies could include multi-nuclear lanthanoid complexes

or focus on experimentally relevant parameters. By enabling a detailed exploration of the versatile properties of lanthanoids, the LnQM dataset opens new perspectives for scientific research in this future-oriented domain of chemistry.

CHAPTER 4

ConfRank: Improving GFN-FF Conformer Ranking with Pairwise Training

Christian Hölzer,[*] Rick Oerder,[† ‡] Stefan Grimme,[*] and Jan Hamaekers[‡]

Received 27 August 2024, Published online 20 November 2024.

Reprinted in Appendix B (adapted) with permission[§] from
C. Hölzer, R. Oerder, S. Grimme and J. Hamaekers, *ConfRank: Improving GFN-FF Conformer Ranking with Pairwise Training*, J. Chem. Inf. Model. **64** (2024) 8909, DOI: 10.1021/acs.jcim.4c01524.

Own manuscript contributions

- conceptualization of research question
- development of methodology and ansatz
- performing data curation and related calculations
- interpretation of the results
- writing of the manuscript

[*] Mulliken Center for Theoretical Chemistry, University of Bonn, Beringstr. 4, 53115 Bonn, Germany
[†] Institute for Numerical Simulation, Friedrich-Hirzebruch-Allee 7, 53115 Bonn, Germany
[‡] Fraunhofer Institute for Algorithms and Scientific Computing SCAI, Schloss Birlinghoven 1, 53757 Sankt Augustin, Germany
[§] Permission requests to reuse material from this chapter should be directed to American Chemical Society.

The ConfRank research project centers on advancing the ranking of molecular conformers, a challenge that remains central in drug discovery and related fields of computational chemistry. Leveraging machine learning (ML) and focusing on pairwise training techniques, this approach refines the prediction of energy differences between conformers, ultimately achieving better performance than common semiempirical quantum chemical methods in terms of both accuracy and computational speed.

In pharmaceutical research, the three-dimensional structure of a molecule is critical for its interactions with biological targets [308–310]. Subtle conformational changes can affect binding affinity and thus influence biological activity. However, the complexity of conformational space poses a significant computational challenge. Many molecules adopt multiple conformations with minimal energy differences, making it difficult to identify the most stable structure. Traditional methods to tackle conformer search are often based on molecular-dynamics simulations, which can incur high computational costs [311]. To overcome this, programs such as the Conformer-Rotamer Ensemble Sampling Tool (CREST) employ meta-dynamics for enhanced conformer sampling [312, 313]. Thereby, force fields or semiempirical quantum mechanical methods can be used to explore the potential energy surface. For this purpose, both the GFN-FF force field [16] and the semiempirical GFN-xTB methods [17, 18] are integrated into CREST.

Building upon the advancement of recent ML models predicting single-point energies on individual molecules [240, 266–268, 314], the project presented extends this research by training pointwise ML models on relative energy differences between conformers. Instead of predicting absolute conformer energies for individual samples (pointwise prediction), ConfRank employs a pairwise training strategy by learning energy differences between conformer pairs from an ensemble (pairwise prediction), focusing on learning relative differences rather than absolute energy values, facilitated by a pairwise loss function during training. On a theoretical note, it is argued that taking the difference between two pointwise model predicitions for a pair of conformers is the most reasonable ansatz for predicting relative conformer energies in the light of symmetry requirements. Moreover, this approach can be motivated by the consideration that by focusing on relative rather than absolute energies, the model should become more sensitive to subtle geometric variations, such as rotations of moieties, and might benefit from some degree of error cancellation, potentially reducing biased offsets caused by shifts or scaling issues in the prediction of the absolute energy.

Since in principle any pointwise ML model suitable for the prediction of structure-property relationships could be utilized for the pairwise approach, the performance of different ML models based on molecular graph representation is examined. The training set for all models includes a comprehensive collection of 159 760 organic molecules taken from the GEOM dataset [171], with reference data computed at the r^2SCAN-3c [315] level of theory. Thereby, ensembles were generated using CREST at the GFN-FF level. Subsequently, the conformer ranking accuracy of the ConfRank ansatz was evaluated compared to GFN-FF and GFN2-xTB, being the default option in CREST. Thereby, the ML models trained in a pairwise fashion did not only outperform the GFN-FF and GFN2-xTB methods but also their pointwise trained counterparts. The DimeNet++ [267] demonstrates best overall performance in terms of computational cost to accuracy ratio and is therefore used as baseline hereinafter. On the GEOM test set it reduces the GFN-FF mean absolute deviation (MAD) of 4.08 kcal mol^{-1} down to 0.49 kcal mol^{-1}. Accordingly, the root mean square deviation (RMSD) is improved from 5.65 kcal mol^{-1} down to 0.71 kcal mol^{-1}, achieving an accuracy below the conformer-ranking-relevant energy window of 1.0 kcal mol^{-1}.

ConfRank's enhancements in statistical accuracy translate directly into better conformer identification. For this purpose, besides comparison of basic statistical measures such as MAD or RMSD, correlation and ranking coefficients are taken into account as well as self-developed metrics centered on conformer ranking. The latter include sign flip probability and the probability for containing the lowest conformer within a given energy window. The pairwise trained DimeNet++ successfully identifies 81 % of the lowest-energy conformers, a considerable increase compared to the 10 % success rate achieved by GFN-FF and even outperforming GFN2-xTB (47 %). Such high accuracy in identifying the correct conformer is critically important CREST / CENSO pipelines [116, 316], where incorrectly ranked conformers lead to high computational costs at later stages of the optimization funnel. Furthermore, using a pairwise trained ML model yields an almost 100-fold computational speedup for energy calculation compared to GFN2-xTB, leveraging GPU-parallelization and batching.

To assess the robustness and generalizability, ConfRank was also tested against external datasets, including QM9 [170] and GMTKN55 [89]. These datasets contain a wide variety of molecular structures, differing in size, complexity, and chemical composition. On the QM9 dataset, large conformer ensembles on GFN-FF level were evaluated, whereby on the conformational subsets of the GMTKN55 chemical relevant ensembles were investigated. In both studies the pairwise DimeNet++ excels over the GFN methods. Moreover, the model was also successfully tested on an ensemble of a biomolecular complex of 176 atoms, a size larger than all structures used in training. Despite slightly worse results compared to previous studies, the ConfRank ansatz shows promising evidence to generalize to larger structures.

CHAPTER 5

dxtb – An Efficient And Fully Differentiable Framework For Extended Tight-Binding

Marvin Friede,* Christian Hölzer,* Sebastian Ehlert,† and Stefan Grimme*

Received 30 April 2024, Published online 09 August 2024.

Reprinted in Appendix C (adapted), with the permission of AIP Publishing‡ from
M. Friede, C. Hölzer, S. Ehlert and S. Grimme, *dxtb – An efficient and fully differentiable framework for extended tight-binding*, J. Chem. Phys. **161** (2024) 062501, DOI: 10.1063/5.0216715.

Own manuscript contributions

- conceptualization of research question
- development of methodology and ansatz
- design of software framework
- supporting in visualization and writing

* Mulliken Center for Theoretical Chemistry, University of Bonn, Beringstr. 4, 53115 Bonn, Germany
† AI4Science, Microsoft Research, Evert van de Beekstraat 354, 1118 CZ Schiphol, Netherlands
‡ Permission requests to reuse material from this chapter should be directed to AIP Publishing.

The integration of quantum mechanical methods into machine learning (ML) models has become increasingly fundamental in advancing computational quantum chemistry (QC). Numerous approaches have emerged that utilize ML to predict chemical properties based on features calculated by existing QC methods [318]. Among those, the extended tight-binding methods GFN1-xTB [17] and GFN2-xTB [18] have emerged as effective semiempirical quantum mechanical methods for fast and robust optimization of geometries and calculation of molecular properties. In this project, the groundwork is laid to enhance these GFN-xTB and enable their further integration into ML frameworks.

The developed method dxTB is an efficient and fully differentiable framework for extended tight-binding, currently supporting the GFN1-xTB Hamiltonian. Thereby, dxtb provides a re-implementation of the GFN1-xTB method in PyTorch, which allows for automatic differentiation (AD) and seamless integration into ML frameworks. Generally, fast computation of gradients is paramount for QC methods as it is not only required for geometry optimization but also for the calculation of higher-order properties. Although analytical gradients and Hessians are favorable in terms of accuracy and speed, their derivation demands considerable effort and expertise, often resulting in long development times and complicated expressions that can be difficult to adapt for new methods. Alternatively, numerical differentiation for nuclear gradients, whereby derivatives are approximated by evaluating finite perturbations of atomic positions, is prone to numerical inaccuracies, and scales poorly with system size, making it less feasible for investigations of larger systems. In contrast, AD offers a compelling solution by automatically deriving partial derivatives through the application of the chain rule to the sequence of arithmetic operations conducted in a computer program. By those means, AD combines the accuracy of analytical methods with the flexibility and ease of implementation typically associated with numerical approaches, while remaining highly scalable and adaptable to new models. Moreover, AD inherently facilitates efficient computation of derivatives of the output quantity (typically energy) with respect to arbitrary input parameters, not just nuclear coordinates. This capability within dxTB is particularly beneficial for exploring the influence of individual parameters in highly-parametrized methods such as GFN1-xTB. Such flexibility is generally not achievable using analytic closed-form solutions, due to the enormous required effort in manual deduction and the large number of involved parameters.
However, dynamic computational graphs, as used in frameworks like PyTorch, present challenges for iterative procedures such as the self-consistent field (SCF) procedure in GFN-xTB methods. These can be addressed either by explicitly differentiating through all iterations (unrolling) [319, 320] or by applying implicit differentiation [320]. Explicit differentiation for the SCF procedure is supported by dxTB, but naive unrolling in dynamic graphs leads to high memory usage and computational overhead, as the memory consumption scales linearly with each iteration, making it unfeasible for the calculation of large systems. Inspired by experiences from TBMaLT[321], a “perfect guess” shortcut was tested, where the SCF runs outside the graph and is reconnected via a final tracked iteration. However, this yields inaccurate gradients, as intermediate charge dependencies are lost. Implicit differentiation, based on the implicit function theorem, on the other hand requires differentiation only at the SCF’s converged solution, not across all iterations [320]. This removes guess-dependency, yields well-defined errors tied to the SCF convergence threshold, and allows for the flexible choice of iterative solvers. Additionally, it reduces memory usage to a constant footprint.

Regarding software design, dxTB is written in Python and leverages PyTorch [322, 323], a widely used ML framework. As mentioned above, PyTorch’s automatic differentiation and high-performance tensor operations enable efficient computation of derivatives, crucial for optimizing molecular geometries and exploring potential energy surfaces. This eliminates the need for manual derivative

calculations and avoids numerical differentiation errors. Also, PyTorch facilitates GPU computation, making dxTB particularly suited for large-scale ML applications where modern GPU infrastructure becomes rather indispensable. Additionally, PyTorch's widespread infrastructure facilitates seamless integration into research workflows and allows easy extension by other scientists. Also for this purpose, the dxTB framework embraces a highly modular architecture. This modularity simplifies the process of adding new features, integrating alternative Hamiltonians, or coupling the framework with ML models for property prediction. Core functionalities within the software include routines for computing classical energy terms, such as those related to repulsion and dispersion, as well as the management of self-consistent interactions that are fundamental in describing electronic structure. A particularly noteworthy enhancement is the integration of the "libcint" library [324], a state-of-the-art computational backend built to handle integral evaluations with high computational performance. By incorporating libcint via the DQC package [325, 326], dxTB achieves a remarkable speed boost relative to purely Python-based approaches, ensuring that the framework runs efficiently even on larger and more complex molecular systems. For the implementation of implicit differentiation the xitorch library[327] was utilized.

Compared to the compiled, Fortran-based tblite implementation of GFN1-xTB [328], dxTB achieves comparable runtimes for energy and gradient evaluations, managing to limit the overall slowdown to a factor of about two to five. Note that this difference is expected when contrasting an interpreted language like Python with a compiled language such as Fortran, yet the aforementioned optimizations substantially close the performance gap. For example, evaluating energies and gradients for a large LNCI16 system with 538 atoms [329] takes approximately 180 s with dxTB, versus 41 s with tblite. When parallelized over four cores, these runtimes reduce to 50 s and 14 s, respectively. Notably, dxTB with libcint and implicit differentiation significantly outperforms its naive Python AD version, which requires around 4 000 s for the same system. For the QM9 dataset [170], which comprises around 134 000 molecules, total energy and gradient evaluations using dxTB with libcint take 214 min. This is substantially faster than both the pure AD implementation of dxTB (397 min) and dxTB using analytical derivatives for overlap integrals (366 min). In comparison, tblite is still roughly twice as fast, completing the full set in 99 min. A similar trend holds for the smaller yet chemically more diverse GMTKN55 dataset [89], where the optimized dxTB outperforms pure AD implementation and completes in 4.1 min, while tblite again runs approximately a factor of two faster (1.7 min).

These results highlight the significant speed-up enabled by integrating libcint, and demonstrate that dxTB supporting the GFN1-xTB Hamiltonian, delivers robust and scalable performance across a wide range of molecular sizes and chemistry. Notably, batching significantly improves runtime. Batched calculations are consistently faster than sequential processing, both in single-core and parallel execution. This performance gain is largely due to the elimination of repeated setup steps and overhead inherent in processing molecules one at a time. For instance, computing energies and gradients for 1 000 molecules from the QM9 dataset takes only 22.2 s when batched on a single core, compared to 55.4 s when processed sequentially. Parallelizing the batch over four cores further reduces the runtime to just 15.9 s. To fully exploit the capability of efficiently computing arbitrary higher-order derivatives using dxTB, vibrational modes and spectroscopic properties of planar ammonia could be reproduced employing non-numerical (exact) Hessians. Furthermore, the IR spectrum of capsaicin could be computed without the need of numerical differentiation. The calculation of these higher-order derivatives could not be done beforehand with the Fortran implementation and is novel to dxTB. Furthermore, even arbitrary-order derivatives could theoretically be computed using dxTB, opening up avenues to further investigate chemical space.

CHAPTER 6

Summary and Outlook

In the field of computational quantum chemistry (QC) the upcoming usage of machine learning (ML) leads to transformative advancements, enabling breakthroughs with respect to both accuracy and computational speed. This allows previously unattainable system sizes and quantities to be calculated and opens up new research possibilities for investigations in the field of biology and medicine. Traditional approaches, such as semiempirical quantum mechanical (SQM) methods and density functional theory (DFT), have long formed the backbone of computational chemistry, allowing accurate predictions of molecular properties and guiding experimental designs. However, these methods face challenges in scalability and computational cost, particularly for high-throughput studies and large or chemically diverse systems. In recent years, advances in computing facilities and QC methods have enabled the creation of large amounts of high-quality reference data, driving the development of ML models that achieve high accuracy at a fraction of the traditional computational cost. Notably, ML not only complements established methods like SQM and DFT but also extends their applicability to problems previously deemed computationally prohibitive. Thereby, ML techniques accelerate molecular property prediction, automate retrosynthetic analysis or aid reaction discovery. In this context, ML is not merely a tool for speeding up calculations but a catalyst for innovation in QC. ML methods facilitate the exploration of uncharted chemical spaces, enabling large-scale compound screening, and allowing for iterative improvement through hybrid workflows that combine ML predictions with physics-based concepts. The development of physics-inspired ML architectures further enhances the ability to capture the underlying equations of quantum systems, paving the way for breakthroughs in drug design and materials development.

This thesis focuses on a) the generation of high-quality data and b) the integration of ML methods into QC algorithms. Data serves as the foundation for most computational methods, making it a critical resource for research. Hence, the availability of datasets with not only ample quantity but also sufficient variance and quality is indispensable. However, especially many areas of inorganic chemistry lack comprehensive datasets to train and test models effectively. To address this gap, the introduced LnQM dataset provides a foundation benchmark in the context of mono-lanthanoid complexes. The dataset features a diverse range of 31 organic ligands, including both neutral and anionic types, and uses permutation of the central lanthanoids to achieve high comparability across the lanthanoid series. In this, 17 269 structures are optimized at the PBE0-D4/def2-SVP level and the featured content comprises geometric, energetic, molecular and electronic properties for each structure at ωB97M-V/def2-SVPD level. By enabling systematic comparisons between lanthanoids

within the LnQM, expected trends in the lanthanoid series such as the SOMO- and HOMO-LUMO gaps can be verified. Moreover, the influence of lanthanoid contraction and discernible trends in bond lengths for different central lanthanoids are apparent. It could also be shown that neither the GFN-FF nor the GFN2-xTB method are fully suitable for treating lanthanoid complexes. Even though geometry optimization through these methods leads to surprisingly similar overall structures compared to PBE0-D4/def2-SVP level, bond lengths around the central lanthanoid are heavily distorted. Both methods are therefore inadequate for extensive usage in the lanthanoid regime and should only be used carefully when treating complexes containing lanthanoids. As noted this is partially due to an insufficient lanthanoid parametrization of the GFN methods. This topic has been further explored by Rose et al. (2024) in [330], showing promising results in the description of lanthanoid complexes using an adopted GFN-FF parametrization. Subsequently, the LnQM study revealed shortcoming in several charge models. Compared to the density-based Hirshfeld charges, the ab initio charge models Mulliken and Löwdin exhibit a large spread and no discernible trend along the lanthanoid series. Even though this is particularly favored due to the diffuse basis set which was employed for the feature calculation, it highlights the fragile nature of these basis set dependent charge models. At the time of the study, the atomic partial charges on the central lanthanoid could best be described by the charge extended Hückel model. Nevertheless, this highlights the importance of evaluating different atomic partial charge theories and the ongoing need for consistent and reliable charge models, particularly in the field of inorganic chemistry. To overcome aforementioned charge partition issues, new methods are constantly being developed. In the course of this research direction, the LnQM dataset has been utilized for parametrizing new charge models, such as the Charge Extended Hückel model, as demonstrated by Müller et al. (2024) in [331]. Beyond these insights for QC methods, the systematic construction of the included lanthanoid complexes allows for the methodical evaluation of QC and ML model performance across different elements. The controlled ligand motif provides a unique framework for evaluating the performance of new QC and ML models in a manner that isolates the influence of the central lanthanoid. By maintaining structural consistency, the methodology enables a direct comparison of model performance across the entire lanthanoid series, thereby facilitating the development and benchmarking of models tailored to these elements. As a result, the LnQM dataset not only supports research in the realm of lanthanoids, but might also inspire the construction of datasets in adjacent chemical domains.

Another central topic of this thesis is the integration of ML into QC workflows, with the ConfRank ansatz addressing a core challenge in computational chemistry: the energetic ranking of molecular conformers. Common computational techniques for conformer generation and ranking often rely on fast force field methods, which can predict geometries but often lack the accuracy in energy required to confidently rank near-degenerate conformers. By employing a pairwise training approach, the prediction accuracy of relative conformer energies can be substantially improved. For a pair of two conformers from an ensemble, pseudoenergies are inferred using a pointwise ML model. The conformational energy for the given pair is then calculated using the difference between these pseudoenergies. In the study it is even shown that the plain difference between these pseudoenergies is not only the most-straightforward function to combine pseudoenergies for a given pair, but in fact the only viable choice when symmetry and transitivity constraints should hold. This novel finding, albeit rather trivial to formulate, might help future researchers to design even more efficient algorithms for conformer ranking. The employed pairwise loss function focuses on energy differences between conformers, thus honing sensitivity to geometric variations and benefiting from intrinsic error cancellation. Especially the latter is of paramount importance when comparing relative rather than

absolute energies. For the model choice, various state-of-the-art ML architectures in computational QC are deployable, ultimately selecting the DimeNet++ as a representative model, based on criteria such as inference speed and the number of trainable parameters. Using a large organic dataset based on small- to medium-sized drug molecules for training and testing, it can be shown that the prediction accuracy for conformer energies improves dramatically compared to the widely used GFN-FF and GFN2-xTB methods, reducing the root mean square deviation (RMSD) from 5.65 kcal mol^{-1} to 0.71 kcal mol^{-1}. Given that these methods are used as default in CREST, one of the de facto standard tools for generating conformer ensembles, the results highlight the significant possible advancements in conformer search achievable with ML-based approaches. Further tests on datasets covering a vast organic space, such as QM9 and GMTKN55, confirm the robust performance of the ConfRank approach. In a significant part of the ensembles, the lowest-energy conformer could be accurately identified, surpassing the aforementioned traditional ranking methods by a substantial margin. For example, the RMSD on the GMTKN55 subsets improves by 29 % on average. Furthermore, the performance of the ML model even closely resembles the performance of the reference r^{2}SCAN-3c composite meta-GGA DFT-functional on some subsets. Beyond its notable predictive accuracy, the new method is up to orders of magnitude faster than even GFN-FF, particularly when utilizing GPU infrastructure - which will probably become indispensable for computational QC laboratories in the future. This speedup in ranking makes it feasible to scan through vast libraries of conformer geometries and lowers the barrier for large-scale studies, such as those often required in drug development. Also, the development of new metrics for the evaluation of conformer ranking accuracy during this work will help future developers to more efficiently assess and compare new model performances. While future developments can expand on capabilities to account for non-covalent interactions and incorporate molecular charges, thereby expanding the range of analyzable systems, ConfRank already demonstrates highly promising performance in improving the energetic ranking of conformers using ML techniques.

The concluding work in this thesis transitions from merely using ML models as additive corrections to existing QC algorithms to investigating how the two fields QC and ML can be holistically integrated. To achieve this, the well-established semiempirical extended tight-binding (xTB) program has been reimplemented in PyTorch, a leading ML framework written in Python. The differential xTB (dxTB) model allows for full differentiability by leveraging PyTorch's native support for automatic differentiation. Using backpropagation - a technique widely employed in the domain of ML neural networks - this optimization technique can now be applied to the original method xTB. One challenging aspect of the implementation lies in the self-consistent field (SCF) procedure, a recursive part of the calculation that can lead to a computational graph blow-up when using standard explicit differentiation. To circumvent this and maintain computational efficiency, implicit differentiation is implemented, utilizing the implicit function theorem. This allows differentiation with respect to the optimality conditions rather than the entire iterative process. The dxTB implementation is almost fully vectorized, enabling GPU acceleration for enhanced computational performance. Additionally, all integral-related computations are delegated to the high-performance C library libcint, offloading the most time-consuming tasks and allowing overall efficiency close to the Fortran reference implementation. The dxTB implementation supports batch-wise processing, allowing efficient computations for multiple molecular systems simultaneously. Furthermore, it enables higher-order nuclear derivatives than previously possible with the default xTB implementation, in fact even up to arbitrary orders. Even derivatives with respect to any input variable, such as parameters from the GFN parameterization, are now feasible. These parameters can also be optimized directly through gradient descent and backpropagation, offering a new way to optimize the model for specific applications. In the future,

dxTB will support additional Hamiltonians beyond GFN1-xTB, such as GFN2-xTB with its multipole expansion. Moreover, dxTB enables the native integration of xTB into PyTorch ML models, allowing it to become an integral part of the ML pipeline rather than serving solely as a feature provider. Thereby, through backpropagation, all parameters of the GFN1-xTB implementation can be individually trained, enabling the native adjustment of the GFN parametrization to less commonly studied chemical groups, such as lanthanoids. The LnQM dataset [183], for instance, could serve as a starting point for such efforts. Additionally, KAN networks [332] could be applied to analyse the functional dependencies of physical quantities. Their learnable activation functions make them particularly apt for the design and optimization of internal functions, such as the calculus of coordination numbers. Thereby their inclusion might provide a flexible attempt for fine-tuning and extending the GFN-xTB model further. The development of dxTB illustrates the paradigm shift from applying ML to QC towards fully integrating QC into ML frameworks. Thereby, dxTB serves as a proof of concept on how QC can evolve into an integral component of ML frameworks, combining the strengths of both disciplines.

In summary, the work of this thesis provides advancements in improving computational QC with ML along multiple dimensions. Starting from the generation of datasets, over using ML models within existing QC workflows, up to the integration of QC and ML methods. The combination of the QC and ML domains is expected to contribute significantly to future research in material science, medicine and biology. For this purpose, this thesis provides a toolbox and foundation for future developments.

Appendix

APPENDIX A

Hybrid DFT Geometries and Properties for 17k Lanthanoid Complexes – The LnQM Dataset

Christian Hölzer,[*] Igor Gordiy,[†] Stefan Grimme,[*] and Markus Bursch[‡]

Received 14 November 2023, Published online 18 January 2024.

Reprinted in Appendix A (adapted) with permission[§] from
C. Hölzer, I. Gordiy, S. Grimme and M. Bursch, *Hybrid DFT Geometries and Properties for 17k Lanthanoid Complexes – The LnQM Data Set*, J. Chem. Inf. Model. **64** (2024) 825, DOI: 10.1021/acs.jcim.3c01832.

[*] Mulliken Center for Theoretical Chemistry, University of Bonn, Beringstr. 4, 53115 Bonn, Germany
[†] Department of Chemistry and Applied Biosciences, ETH Zürich, Vladimir-Prelog-Weg 2, Zürich 8093, Switzerland
[‡] Max-Planck-Institut für Kohlenforschung, Kaiser-Wilhelm-Platz 1, 45470 Mülheim an der Ruhr, Germany

Abstract The unique properties of lanthanoids and their diverse applications make them an indispensable part of modern research and industry. While the field has garnered attention, there remains a gap in available molecule datasets that facilitate both classical quantum chemistry calculations and the burgeoning field of machine learning in data science applications. This research addresses the need for a comprehensive dataset that allows for a comparative analysis of various lanthanoids. The herein presented, curated dataset includes 17269 mono-lanthanoid structures derived from 1205 distinct ligand motifs. Structures encompass all 15 lanthanoids in the +3 oxidation state and exhibit molecular charges ranging from −1 to +3, including structures with high spin multiplicity up to 8. Starting from lanthanum complexes, samples were processed with a permutation of the central lanthanoid atom, resulting in highly comparable subsets, facilitating comparative studies in which the influence of the lanthanoid can be investigated independently of ligand effects. The dataset provides a broad range of features such as PBE0-D4/def2-SVP optimized geometries and optimization trajectories, while also covering ωB97M-V/def2-SVPD energies, rotational constants, dipole moments, HOMO−LUMO energies, and Mulliken, Löwdin, and Hirshfeld population analyses. Additionally, coordination numbers, polarizabilities, and partial charges from D4, EEQ, GFN2-xTB, and CEH calculations are included. The dataset is openly accessible and may serve as a basis for further investigations into the properties of lanthanoids.

A.1 Introduction

Lanthanoid complexes play an important role in various scientific and technological applications, such as light-emitting materials used in display technologies and optical sensors [290–292, 333, 334], bio-imaging [335–337], and are particularly important in magnetic resonance and fluorescence imaging [293, 294, 338–340]. Lanthanoid complexes are also relevant for homogeneous catalysis [295, 341–344]. Furthermore, lanthanoids are employed as neutron absorbers in nuclear reactors, for instance, up to less commonplace applications as energy sources in satellites [296, 345].

However, there is currently a significant gap in the available data sets for conducting theoretical studies on lanthanoid complexes. This lack of available data sets for theoretical studies on lanthanoid complexes impacts both quantum chemical (QC) methods, such as density-functional theory (DFT), and emerging machine learning (ML) approaches [69, 346, 347]. Although ML methods are considered promising, they are data-intensive and require sufficient amounts of reference data for model building and validation. Their effectiveness is therefore highly dependent on the availability and quality of data. Diverse quantum chemical data sets for more common chemical elements exist for offering validation opportunities for QC methods. They typically feature relatively few but high-quality data points, usually at coupled-cluster wave function theory level. For instance, the S66x8 dataset [187] provides valuable insights into intermolecular interactions at a highly accurate coupled-cluster reference level (CCSD(T)/CBS) [348]. Moreover, the GMTKN55 database offers an extensive repository of reaction data widely employed in QC research to validate and enhance computational methods for thermochemical and kinetic properties [89, 185].

On the other hand, datasets generated with relevant features for ML applications often contain abundant data points of potentially lower data quality (often below DFT level). The majority of existing ML datasets focus on organic or main group elements. A typical example is the widely utilized QM9 dataset [169, 170], which contains DFT calculations for a variety of small organic molecules, and the ANI-1 dataset [157, 186], which was developed specifically for neural network applications to predict

molecular properties. In addition, datasets such as QM7, QM7b, and QM8 allow for a more in-depth study of the electronic properties of small organic molecules [155, 169, 349–351]. In this context, it is pertinent to accord special recognition to the "transition metal Quantum Mechanics" (tmQM) dataset, a compilation that encompasses an extensive set of transition metal geometries along with corresponding properties of organometallic structures [182]. The tmQM was recently extended by the tmQMg and the tmQMg-L dataset [352, 353]. Although the tmQM dataset considerably extends the spectrum of available sets beyond organic chemistry to transition metals, there is currently no extensive dataset for ML applications in the field of lanthanoids.

In the present study, we introduce a comprehensive dataset for all 15 lanthanoids, focusing on small to medium-sized mono-lanthanoid complexes, optimized for compatibility with conventional QC methodologies. This **La**n**n**thanoid **Q**uantum **M**echanics dataset (LnQM) is designed to provide features and geometries for ML applications and to facilitate the development of semiempirical methods with focus on lanthanoids. Our objective is to extend the prevailing momentum of constructing specialized datasets targeting distinct regions of the periodic table, akin to the tmQM dataset. By doing so, we aim to facilitate scientific investigations in the domain of lanthanoids. Furthermore, manual generation of thousands of lanthanoid structures is both time-consuming and susceptible to errors. Data sources such as the Cambridge Structural Database (CSD) [354] present limitations, as the quality of the data cannot automatically be assured. Anomalies such as incorrectly assigned hydrogen atoms could emerge, which cannot be accurately identified and filtered out through automated means. For this reason, synthetically generated data was used for the study. To facilitate this, we employed the recently published Architector program, which is specifically designed for the robust generation of organometallic complexes [355]. A key advantage of generating synthetic data is that it offers control over created structures, parameters like molecular charge, oxidation state, and ligand properties. Geometries with various ligand motifs are generated on the GFN2-xTB level [18, 148], subsequently optimized on the PBE0-D4/def2-SVP level, and singlepoint features evaluated on the ωB97M-V/def2-SVPD level [68, 82, 91, 92, 99, 297, 300]. Details of the software and methods used are detailed in section A.2. Section A.3.1 outlines the workflow for generating the samples and the dataset. Following this, section C.4 examines the dataset's properties. A concluding discussion is provided in section B.6.

A.2 Computational Details

Quantum mechanical calculations were performed using the ORCA computational chemistry software suite (version 5.0.4) [40, 356]. Using the PBE0 functional [68, 82, 91, 92, 297], geometry optimizations employed the def2-SVP basis set [45, 298, 299] and were carried out under default convergence settings. Dispersion corrections were applied with the D4 scheme [302], integral approximations were enabled with the RIJCOSX setting [357], and the DEFGRID2 option was used to set up the DFT integration grid. In accordance with general recommendations on the selection of DFT functionals [86], the single-point feature calculations have been conducted using the range-separated hybrid ωB97M-V functional [99, 300] utilizing the def2-SVPD basis [45, 298, 299]. This functional integrates the post-SCF evaluated VV10 non-local dispersion correction [358]. For feature calculation TightSCF convergence criteria were used. The basis set was chosen to be efficient due to the large number of calculations required for the dataset, while also accommodating for anions via the diffuse functions. The def2-ECP Effective Core Potentials (ECPs) were employed for the lanthanoids [359–361]. Atomic

partial charge calculations are conducted with GFN2-xTB [18, 148] using the tblite framework [328]. Furthermore, for force field calculations, the GFN-FF was applied [16].

The structures of the complexes were generated using the Architector tool (version 0.0.10), the source code of which is available on GitHub [355, 362]. To facilitate utility tasks and enable efficient scaling in the generation of structural complexes, the ArchitectorWrapper package was employed [363]. Data is stored in the hierarchical data format (HDF5) [364]. All scripts were written in Python version 3.11.4 [322, 365].

A.3 Results and Discussion

A.3.1 Creation of the LnQM dataset

A comprehensive overview of the steps involved in generating the LnQM dataset is given in Figure A.1. The workflow for the dataset creation is divided: (1) Sample creation, (2) 3D structure generation, (3) permutation, and (4) feature calculation. The following subsections provide a detailed description of each step.

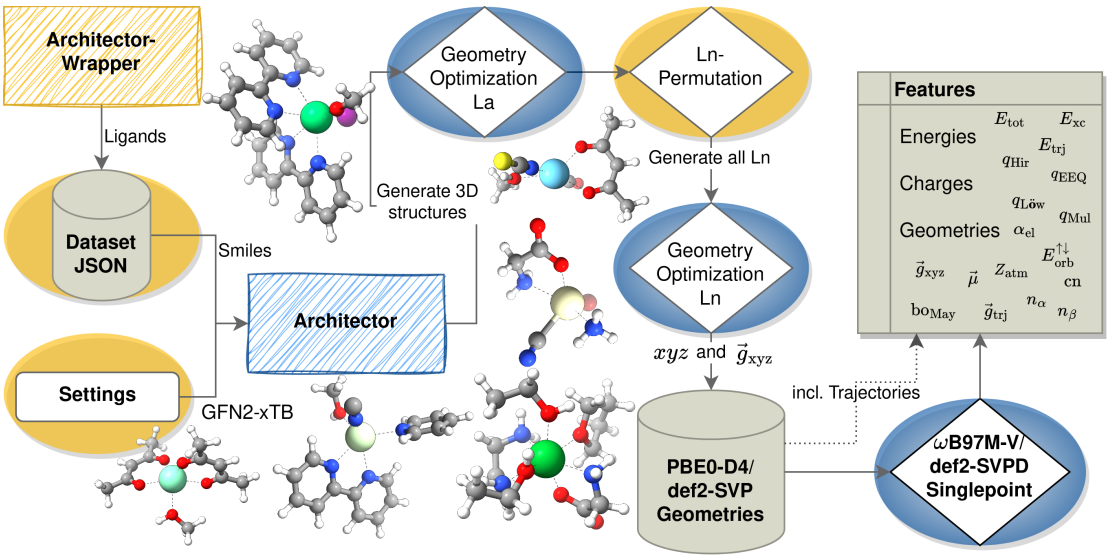

Figure A.1: Workflow for dataset creation. Components using the ORCA (ArchitectorWrapper) framework are indicated by blue (yellow) highlighting. For sample creation, ligands and settings are prepared and fed into the Architector package for generation of 3D structures. Subsequently, these structures are optimized and permuted. Finally, singlepoint properties are calculated for each individual sample and stored in a database.

1. Sample Creation

The initial step involves the selection of ligands that are utilized in constructing the samples. Also, molecular charge and oxidation state of the ligand-lanthanoid complex are determined. The ArchitectorWrapper package is employed for automated generation of input parameters.

Lanthanum serves as the initial central atom and starting point for generating samples. As elaborated in the subsequent "Permutation" section, samples for the other 14 lanthanoids are derived based on these initial lanthanum-based structures. As detailed in Table A.1, lanthanum was selected as the

Element	EC^{atm}	OS	R^{+3} / Å
La	[Xe] $5d^1$ $6s^2$	+3	1.05
Ce	[Xe] $4f^1$ $5d^1$ $6s^2$	+3, +4	1.00
Pr	[Xe] $4f^3$ $6s^2$	+3, +4	1.00
Nd	[Xe] $4f^4$ $6s^2$	+3, +4	0.98
Pm	[Xe] $4f^5$ $6s^2$	+3	0.97
Sm	[Xe] $4f^6$ $6s^2$	+2, +3	0.96
Eu	[Xe] $4f^7$ $6s^2$	+2, +3	0.95
Gd	[Xe] $4f^7$ $5d^1$ $6s^2$	+3	0.94
Tb	[Xe] $4f^9$ $6s^2$	+3, +4	0.92
Dy	[Xe] $4f^{10}$ $6s^2$	+3, +4	0.91
Ho	[Xe] $4f^{11}$ $6s^2$	+3	0.90
Er	[Xe] $4f^{12}$ $6s^2$	+3	0.89
Tm	[Xe] $4f^{13}$ $6s^2$	+2, +3	0.88
Yb	[Xe] $4f^{14}$ $6s^2$	+2, +3	0.87
Lu	[Xe] $4f^{14}$ $5d^1$ $6s^2$	+3	0.86

Table A.1: Overview over lanthanoids and their properties. Denoted are the atomic electron configuration (EC^{atm}), the lanthanoid's formal oxidation states (OS) and effective ionic radii (R^{+3}) [367–370].

starting point for generating lanthanoid samples for two primary reasons. First, it exhibits the largest atomic radius among lanthanoids, making it an ideal candidate for initial structure generation, i.e. allowing for adaptability during lanthanoid permutation without compromising coordination patterns. Second, its natural oxidation state is +3, which is also the only oxidation state universally available across all lanthanoids. In accordance with these considerations, the parameters for metal oxidation state and total spin were set to +3 and 0, respectively. Spin-orbit coupling effects are not considered in this study, as they are expected to be small for the considered properties [366]. We further constrained the coordination numbers (CN) for all complexes to fall within a range of 5 to 9, aiming to encompass a representative set of realistic geometries. Additionally, only samples with molecular charges ranging from -1 and +3 are considered. This charge constraint was implemented to avoid the generation of chemically irrelevant or unrealistic structures, particularly as highly negatively charged systems are prone to yielding implausible (gas phase) geometries not relevant to actual chemical contexts.

A diverse range of ligands was employed to probe the structural and chemical space of lanthanoid complexes. Table A.2 presents a detailed account of the 31 ligands utilized for constructing the dataset as well as their frequency in the lanthanum subset. The selection spans from simple, small uncharged ligands like water, to larger, more complex molecules such as bipiperidine. Mono-atomic anions like fluoride, chloride, bromide, and iodide were also included to study the influence of halogens. The inclusion of ligands like thiocyanate allows to investigate varying degrees of polarity and charge distribution. Additionally, a range of nitrogen-based ligands like pyridine, pyrazine, and ammonia were employed. Also the role of sulfur and phosphorus containing ligands, such as trimethylphosphite, were examined. Moreover, neutral and charged ligands have been taken into account. This variability enables a study of the interactions between lanthanoids and multiple charge carriers. The choice of ligands includes a diverse array of binding characteristics, featuring mono- and multidentate ligands,

including bidentate and tridentate configurations. The inclusion of these varied ligands in the sample permits the investigation and categorization of various bond types and spatial arrangements.

The dataset focuses exclusively on mono-nuclear lanthanoid structures. Aside from the central lanthanoid atom, no heavy metals or similar entities are included within the ligands or the overall complexes. This decision was made to isolate the effects of the lanthanoid center and to avoid additional complexities arising from the presence of other heavy atoms. Systems are inspired by experimentally synthesizable compounds although explicit verification or restriction to exclusively experimentally available structures has not been performed.

2. 3D Structure Generation

Subsequent to ligand selection, 3D structures are generated by attaching ligands to the central lanthanoid using the Architector framework, which involves determining metal center symmetry, ligand geometry, complex conformers, and identifying the lowest conformers. The Architector setup was configured using the following settings: GFN2-xTB [18, 148] was selected for both conformer assembly and final conformer evaluation, multiple conformers were generated and relaxed, with up to 10 different metal-centered symmetries investigated. Initial lanthanoid structures featured lanthanum in its +3 oxidation state. Final geometric relaxation of the complexes with GFN2-xTB was conducted, improving the quality of the generated structure. Lanthanum was chosen as initial central lanthanoid ion for this preliminary structure generation step. Finally, the GFN2-xTB geometries were optimized at the PBE0-D4/def2-SVP level in the gas-phase.

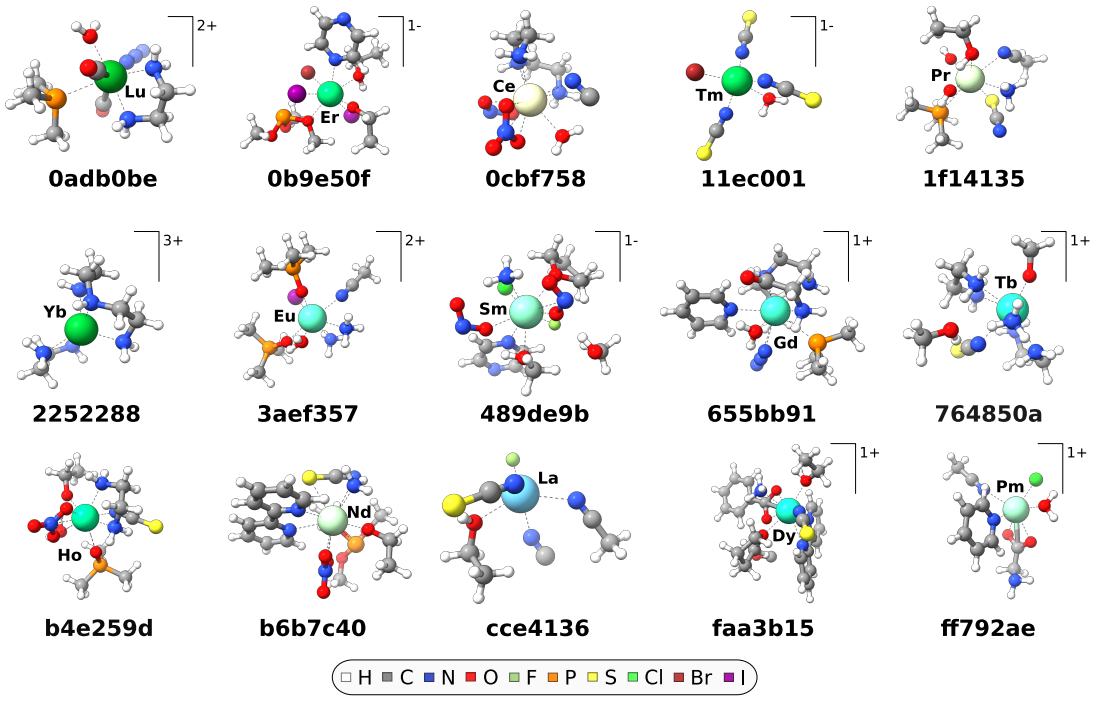

Figure A.2: Overview over various optimized geometries for different ligand motifs and lanthanoids. Given is a shortened version of the unique identifier used in the dataset.

Name	SMILES	Frequency	Charge / *e*
	Neutral systems		
Acetonitrile	CC#N	252	0
Ammonia	N	463	0
Bipiperidine	N1CCCCC1-C2CCCCN2	205	0
Carbon monoxide	[C-]#[O+]	252	0
Diethylenetriamine	NCCNCCN	164	0
Ethanol	CCO	225	0
Ethylendiamin	NCCN	197	0
Methanol	CO	272	0
Pyridine	C1CCNCC1	242	0
Pyrazine	C1CNCCN1	267	0
Tetrahydrofuran	C1CCCO1	217	0
Trimethylphosphine	CP(C)C	228	0
Trimethylphosphine oxide	CP(=O)(C)C	245	0
Trimethylphosphite	COP(OC)OC	216	0
Water	O	240	0
	Ionic systems		
Azide	[N-]=[N+]=[N-]	196	-1
Acetylacetonate	CC(=CC(=O)C)[O-]	177	-1
Benzoate	C1=CC=C(C=C1)C(=O)[O-]	155	-1
Bromide	[Br-]	179	-1
Chloride	[Cl-]	182	-1
Cyanide	[C-]#N	395	-1
Ethanolate	CC[O-]	193	-1
Fluoride	[F-]	202	-1
Glycinate	C(C(=O)[O-])N	167	-1
Hydroxide	[OH-]	174	-1
Iodide	[I-]	186	-1
Isothiocyanate	S=C=[N-]	187	-1
Methanolate	C[O-]	204	-1
Nitrate	[N+](=O)([O-])[O-]	178	-1
Nitrite	N(=O)[O-]	197	-1
Thiocyanate	[S-]C#N	203	-1

Table A.2: The 31 ligands used for the dataset creation, their SMILES representation, and their abundance in the lanthanum subset.

3. Permutation

To create a dataset with comparable lanthanoid configurations, the central lanthanum ion of the initially generated and optimized complexes was interchanged with the other lanthanoids to create corresponding subsets for each of them. At this point lanthanum complexes are a chemically reasonable starting point, due the typically similar chemical compositions throughout the lanthanoid series. Moreover, the ionic radii of the lanthanoids vary systematically as highlighted in Table A.1 with lanthanum incorporating the largest ion radius, thus preventing structural clashes in the subsequent permutation steps. By this permutation strategy, the volume of data that is essential for data science applications is systematically increased. Further, direct comparability across various lanthanoids is guaranteed, enabling comparative benchmark studies evaluating performance for individual lanthanoids.

During permutation, the ligands and thus the molecular charge remained unchanged; only the spin multiplicity was affected by the substitutions. As the attached ligands are exclusively closed-shell, the spin multiplicity was set equivalent to the multiplicity of the isolated lanthanoid ion in the +3 oxidation state according to the NIST database [371].

Furthermore, an examination of spin multiplicities of actinoid compounds as reported by *Andreadi et. al* aligned with this adopted methodology, prompting its extrapolation for lanthanoids [372]. In total, 2406 closed-shell systems are included in the dataset, for higher spin multiplicities up to +8 over 2200 samples are available per multiplicity.

The permutated structures were subsequently optimized at the PBE0-D4/def2-SVP level. To ensure consistency in data management, samples in the dataset are tagged with a unique identifier (UID). This UID is based on a hash value of ligands and configuration. Hence, for each lanthanoid, the LnQM holds a distinct subset with unique structures, but shared ligand motifs. The substitution of lanthanoids generally exerts a negligible impact on the overall molecular geometry, as illustrated in Figure S1 in the Supporting Information. For a comparison of the geometric structures for different lanthanoids, see Figure A.2, which displays a selection of structures from the dataset.

Dataset Composition

Initially, 1500 structures were processed using Architector, of which 1205 (80%) successfully underwent 3D generation. During the DFT optimization phase, 17661 samples (98%) converged, and subsequently, 17385 single points (98%) were successfully completed. To ensure the quality of the dataset, different criteria for the inclusion of samples are scrutinized. Exclusion criteria are based on several constraints:

1. Heavy atom root-mean-square deviation (RMSD), excluding hydrogen, for each sample was ensured to not exceed 3 Å from the initial structure generated by Architector framework on the GFN2-xTB level.

2. Permuted samples that diverged from the initial PBE0-D4/def2-SVP optimized lanthanum complex by an RMSD of more than 4Å were discarded.

3. Samples with an artificially low coordination number $CN_{Ln} < 2.5$ were excluded.

4. All samples with a Ln-{C,N,O,S} distance surpassing $15\,a_0$ ($\sim$ 8Å) were removed.

5. Intermediary non-convergence during the geometry optimization process, i.e. observed in ytterbium samples, led to sample exclusion.

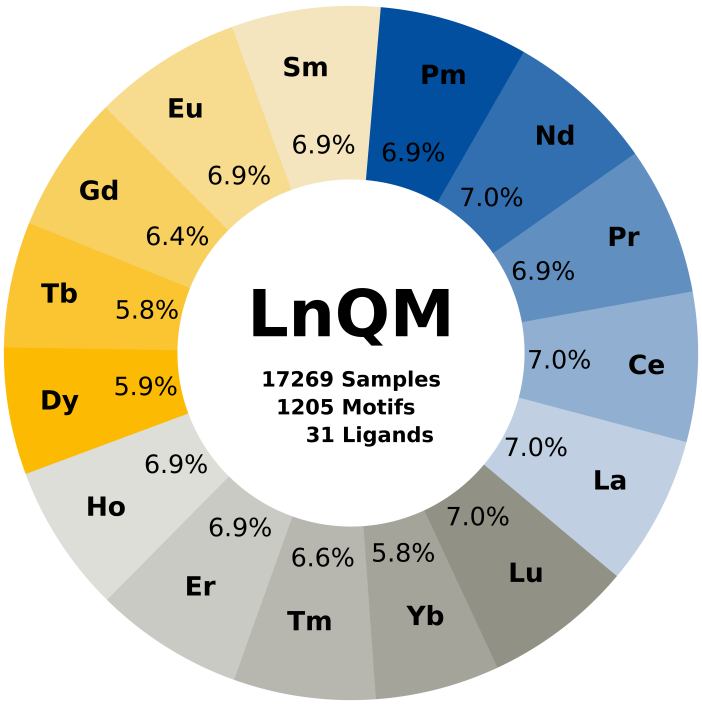

Figure A.3: Overview over the LnQM dataset composition and relative sizes of individual lanthanoid subsets.

The refined dataset comprises a total of 17269 data points containing 1205 different motifs in total. Within this, 657 motifs are presented in all possible permutation combinations, accounting for 9855 directly comparable structures. The dataset is balanced in terms of lanthanoid distribution, the size of each lanthanoid subset is given in Figure A.3.

Figure A.4 depicts the elemental and molecular size distribution within the LnQM. As expected for mostly organic ligands, hydrogen and carbon atoms dominate the dataset with relative frequencies of approximately $\nu_{\mathrm{H}} = 0.5$ and $\nu_{\mathrm{C}} = 0.4$, respectively. The relative frequencies for nitrogen and oxygen range up to $\nu_{\mathrm{N,O}} = 0.1$. Other elements included in the data, such as fluorine, phosphorus, sulfur, chlorine, bromine, and iodine, as well as lanthanoids, exhibit relative frequencies significantly below $\nu_{\mathrm{x}} = 0.1$. Furthermore, molecules carry a molecular charge $\in [-1, +3]$, with 30% of the molecules being neutral, while 50% carrying a charge of ± 1. The dataset predominantly features medium-sized structures, with the average sample size being $n_{\mathrm{atm}}^{\mathrm{avg}} = 42$ atoms. The sample sizes range from $n_{\mathrm{atm}}^{\mathrm{min}} = 10$ to $n_{\mathrm{atm}}^{\mathrm{max}} = 87$ atoms. Structures with $n_{\mathrm{atm}} > 90$ atoms are deliberately excluded during dataset creation, as choice of coordination number and ligands inherently limit the overall size of the samples.

4. Feature Calculation

All DFT-based features were computed at the ωB97M-V/def2-SVPD level. Geometric properties include optimized geometries, trajectory points of atomic positions during optimization, as well as energies and gradients at these points. These points are chosen from the total trajectory using a logarithmic sampling scheme. The trajectory points provide a dynamic view of the structure, shedding light on the the local structure of the PES. Such insights prove valuable for studies on non-equilibrium states and temporal analyses such as molecular dynamics. Additionally, coordination numbers from a D4 calculation are available, allowing investigations of coordination patterns. Energetic properties

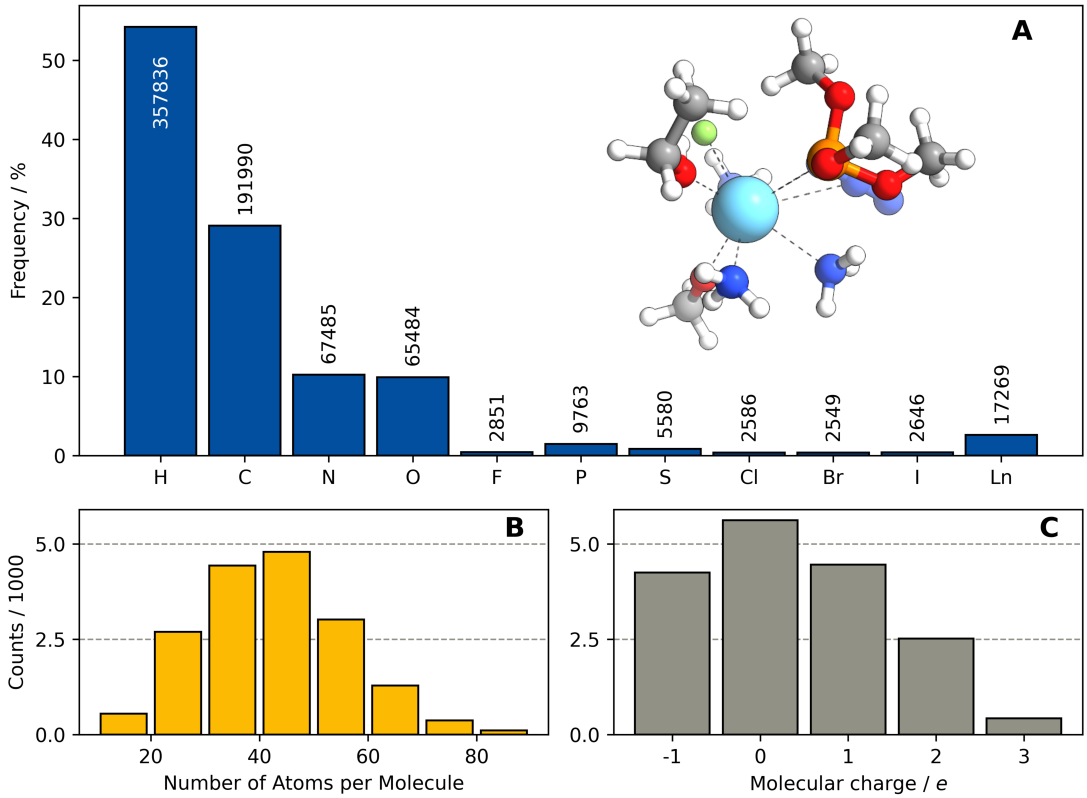

Figure A.4: The figure provides a comprehensive overview of the LnQM dataset composition. The upper section (A) depicts the relative distribution of included elements, with lanthanoids consolidated under the label Ln. Each column is provided with absolute frequency values. The lower left section (B) visualizes the distribution of molecular sizes. The lower right section (C) displays the distribution of molecular charges across all structures. The entire dataset comprises 17269 structures.

include the total single-point energy, dispersion correction, exchange-correlation energies. Orbital energies are available (for open-shell systems in both spin channels) are provided. Molecular properties include rotational constants, components of the dipole moment along rotational axes, its overall magnitude, and separate contributions from electronic and nuclear sources. Electronic properties include the molecular charge, number of unpaired electrons, and overall electron count. Additionally, the dataset features polarizabilities from D4 calculations using default parameters and diverse sets of atomic charges from electronegativity equilibration (EEQ) model [149, 373], charged extended Hückel (CEH) [304], and GFN2-xTB [18, 148] methodologies. Population analyses using Mulliken [305], Löwdin [306], and Hirshfeld [303] partition schemes were conducted including spin populations. Mayer bond orders [374], along with free and bonded valences, are included.
Note that only a selection of features can be discussed in detail. A comprehensive overview over all properties featured in the LnQM is given in Table S1 in the Supporting Information.

A.3.2 Analysis of Selected Properties

This chapter offers an analysis of select properties included in the LnQM dataset. It further examines structural aspects of the LnQM and several properties are investigated for possible trends within the lanthanoid series.

Frontier Orbital Gaps

Many chemical and spectroscopic characteristics of organometallic complexes correlate with the energetics of the corresponding frontier orbitals. For closed-shell compounds typically the HOMO-LUMO gap ΔE_{H-L} – the difference between the highest doubly occupied and the lowest unoccupied molecular orbitals – determines many properties and has a significant impact on the chemical behavior. With many of the included lanthanoid complexes being open-shell compounds, also the SOMO-LUMO gap ΔE_{S-L} is of key importance. As illustrated in Figure A.5, clear trends across lanthanoids are evident in both the average SOMO−LUMO and HOMO−LUMO gaps. Due to the electronic configuration of the lanthanoid series, unpaired electrons are added to the f-shell from lanthanum to gadolinium. From gadolinium to lutetium, these unpaired electrons are progressively paired correspondingly. The SOMO−LUMO gap depicted for the higher populated spin channel is thus sensitive to systems with many unpaired electrons, like europium (half-filled f-shell, deviation of 0.8 eV), and remains nearly constant from terbium onward as the unpaired electrons are progressively paired up. The electronic configuration also results in a continuous increase in the average HOMO−LUMO gap up to gadolinium (highest number of unpaired electrons), followed by a decrease towards lutetium. This leads to an overall difference in the average HOMO−LUMO gaps of 2.0 eV.

Atomic partial charges

Partial charges play a crucial role across various chemical domains, such as EEQ within D4 dispersion corrections [302], the newly developed q-vSZP basis set [304] or calculation of the hydrophilic/lipophilic index [375]. Atomic partial charges of the central lanthanoid are explored based on various charge models. Figure A.6 contrasts (basis set-based) ab initio and semiempirical charge models to density-based Hirshfeld charges. In comparison to AIM charges [376, 377], Hirshfeld charges tend to be numerically smaller and thus in better agreement with common chemical sense. Hirshfeld and other ab initio partial charges have been calculated on the ωB97M-V/def2-SVPD level.

The ab initio charge models, Mulliken and Löwdin, are basis set dependent [20]. Particularly, Mulliken partial charges are spread out due to the diffuse functions in the def2-SVPD basis [20] and do not feature a strong correlation with Hirshfeld partial charges (Pearson correlation coefficient $r_{Mul} = 0.44$, $r_{Löw} = 0.38$). A charge model comparison across varying basis set sizes is given in Supporting Information section 4. This study reveals that Hirshfeld charges exhibit minimal basis set dependence, establishing them as a robust reference choice. Additionally, basis set dependence for other charge models is relatively small, affirming the continued suitability of the def2-SVPD basis set in this context. Although the semiempirical methods EEQ, CEH and GFN2-xTB include parameters for lanthanoids, these are not explicitly fitted for each individual lanthanoid element but interpolated through the series. In the EEQ method, the partial charges across different lanthanoids remain consistent ($r_{EEQ} = 0.27$, $MD_{Ln}^{EEQ} \approx 0.6\,e$) due to uniform parameters across all lanthanoids. The CEH model, optimized to reproduce Hirshfeld partial charges on ωB97X-D4/def2-TZVP level, excellently reproduces the Hirshfeld charges for lanthanoids ($r_{CEH} = 0.79$, $MD_{Ln}^{CEH} < 0.1\,e$). The trend towards heavier lanthanoids can possibly be attributed to the interpolation of CEH parameters between lanthanum and lutetium.

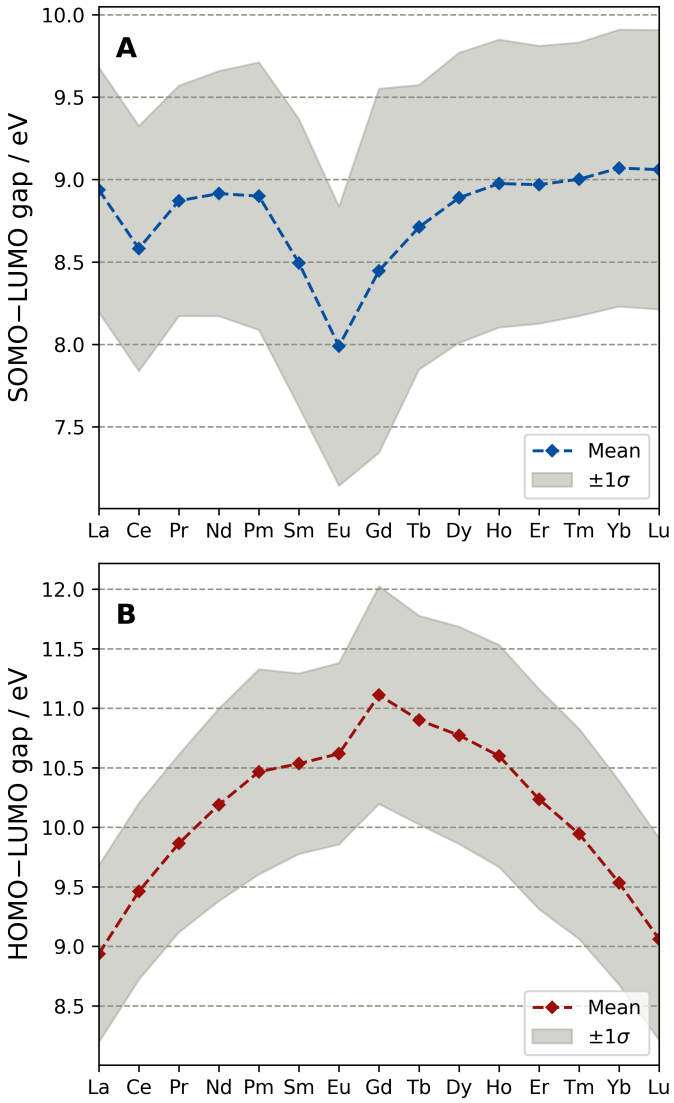

Figure A.5: Average SOMO−LUMO gap (A) and HOMO−LUMO gap (B) for each subset, calculated on the ωB97M-V/def2-SVPD level. The 1σ standard deviation is shaded in gray.

These results are remarkable, particularly when considering the CEH method's low computational cost, as currently, no other semiempirical charge model can achieve such accuracy. Accounting for the f-in-core approximation [18], partial charges calculated with GFN2-xTB lead to mean deviations similar to the CEH deviations. However, as can be derived from Figure A.6 and Figure S2 and S3 in the Supporting Information, correlation of GFN2-xTB partial charges with Hirshfeld partial charges is poor ($r_{xTB} \approx 0.00$). GFN2-xTB features rather constant partial charges, similar to the EEQ model, as there is no specific parameterization for lanthanoids. In general, the data suggests that only the CEH charge model is currently capable of adequately reproducing Hirshfeld partial charge distribution on lanthanoid structures. More detailed information is provided in the Supporting Information section 3.

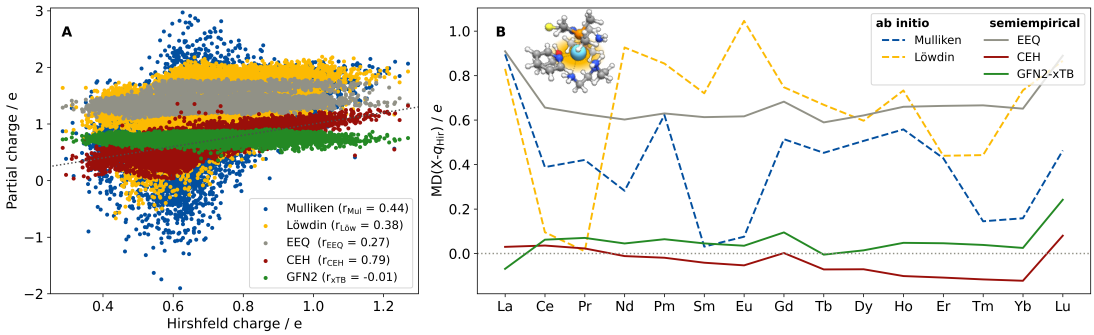

Figure A.6: Overview of atomic partial charges on the central lanthanoid for each subset of the LnQM for different charge models (A). Comparison of the mean deviation (MD) of partial charges on the central lanthanoid atom across various charge models relative to Hirshfeld partial charges (B).

Structural Features

Lanthanoids have gained increasing interest in studies on bioinorganic systems [343, 378], metalloproteins [379, 380] and chelating markers for biomolecular NMR [381–383], among other applications. Therefore, an accurate characterization of their geometrical structures is of key importance. However, for the study of large structures, computationally efficient methods are essential. In this regard, we evaluate the performance of GFN-FF and GFN2-xTB compared to hybrid DFT for the assessment of geometric features.

Nitrogen and oxygen are among the predominant bonding partners of lanthanoids in typical structures since, as small and hard ions in the +3 oxidation state, they tend to form ionic bonds, with small and hard donor atoms particularly favored. Therefore, a comparison of nitrogen and oxygen bond lengths on PBE0-D4/def2-SVP level in different subsets of the LnQM is shown in Figure A.7. For a coherent comparison, only samples with identical ligand motifs are considered. Bonds are determined using a Mayer bond order larger than 0.3. Average bond lengths $d_{\text{PBE0-D4}}$(Ln-X) are reasonably close to expected covalent radii, highlighting the lanthanoid contraction leading to shorter bond lengths. The covalent radii of cerium and ytterbium exhibit distinctive behavior, which might stem from the initial calculation of diatomic covalent radii for cerium and ytterbium [384]. In general, the bonds in LnQM are longer than the theoretically determined covalent radii [384], a result of Pauli repulsion and steric effects. On average the bond lengths are longer by 6% (3%) for lanthanoid - nitrogen (oxygen) bonds. These systematically longer bonds are most likely attributed to inter-ligand steric repulsion, such as touching ligand spheres or imperfectly aligned bidentate ligands.

The potential energy surface (PES) vicinity of the PBE0-D4/def2-SVP optimized structures for different methods is investigated by re-optimizing these structures using a semiempirical (GFN2-xTB) and a force field (GFN-FF) method. The histogram in Figure A.8 provides a comparative analysis of the heavy atom RMSD between PBE0-D4/def2-SVP and GFN2-xTB (GFN-FF) optimized structures over the whole dataset. Both GFN methods cluster around an RMSD between 0.5 and 1 Å. Two effects arising from using the semiempirical and force field methods seemingly cancel each other out leading to similar RMSD distributions – on the one hand, GFN2 incorporates the electronic structure and therefore tends to be closer to the DFT PES. On the other hand, GFN-FF establishes more rigid bonds based on the original topology, making bond changes less likely. Generally, the

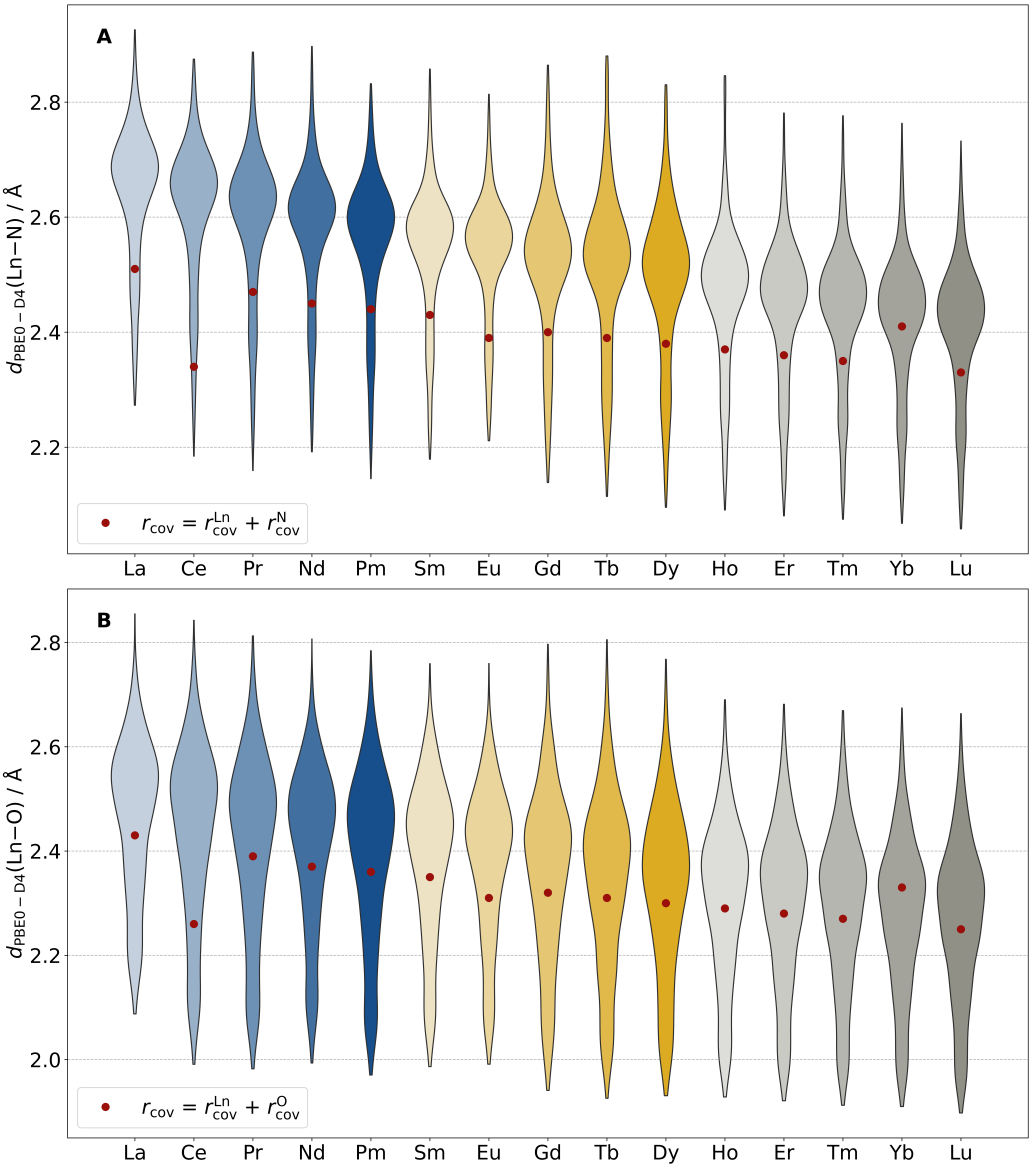

Figure A.7: Bond lengths between central lanthanoid atom and nitrogen (A) and oxygen (B) considered in structures with identical motifs for all 15 lanthanoid subsets. Bonds are identified using Mayer bond orders and respective distance estimates. The red dot indicates the sum of covalent radii of bonded partners $r_{cov} = r^{Ln}_{cov} + r^{\{N,\,O\}}_{cov}$.

RMSD displays only little differences from the PBE0-D4/def2-SVP optimized structures. However, investigating individual bond lengths reveals major discrepancies between the GFN methods and PBE0-D4/def2-SVP. A comparison of bond lengths d_X(Ln-N) and d_X(Ln-O) between DFT and semiempirical or force field level is shown in Figure A.9. The GFN2-xTB bond lengths feature a reasonable correlation $r_{xTB} > 0.70$, though displaying a large spread. The force field shows only weak correlation with the PBE0-D4/def2-SVP bond lengths ($r_{FF} < 0.60$). Indicating reasonable geometries,

although offering room for improvement for geometry optimization on lanthanoids below the DFT level. Better geometries could be achieved by lanthanoid-specific parameterizations for the GFN methods.

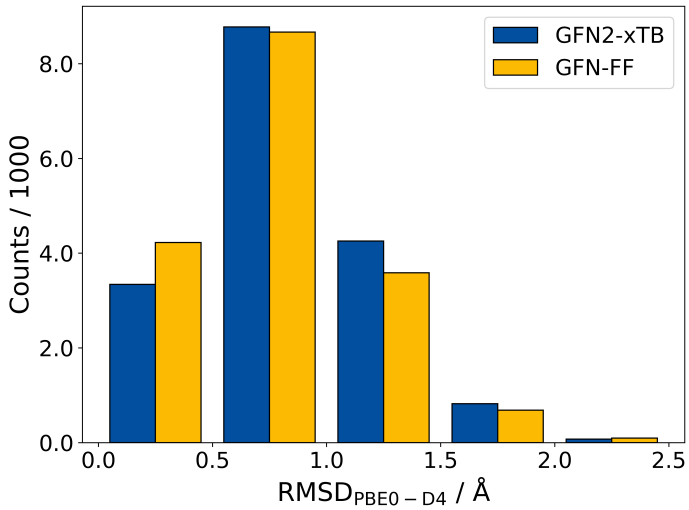

Figure A.8: Comparison of GFN2-xTB and GFN-FF geometry-optimized structures to PBE0-D4/def2-SVP level using the heavy atom RMSD.

A.4 Conclusions

Despite the widespread utility of lanthanoids in various research applications, few publicly available standardized datasets for big data applications are available. To close this gap, the LnQM dataset is introduced, a comprehensive compilation of more than 17000 mono-lanthanoid structures in the +3 oxidation state optimized at the PBE0-D4/def2-SVP level. The dataset's extensive array of DFT features offers a starting point for data-intensive explorations in the realm of lanthanoids. Beyond geometries, the dataset encapsulates gradients, optimization trajectories, coordination numbers, energy-related features like single-point energies and the HOMO−LUMO gap, and partial charges from various charge models (Hirshfeld, Mulliken, Löwdin, EEQ, among others), and electronic attributes such as dipole moments, computed at the ωB97M-V/def2-SVPD level. Due to the variation of the central lanthanoid, several lanthanoid-specific subsets are available for all 15 lanthanoids.

The analysis of the dataset demonstrates notable variations in the HOMO-LUMO gap. Furthermore, LnQM reveals shortcomings in some ab initio and semiempirical charge models, where only the newly developed CEH method provides satisfactory results. Generally, GFN methods yield reasonably accurate geometries for lanthanoid complexes, although there is potential for improvement through dedicated parametrizations for lanthanoids, a capability now made feasible by the LnQM.

The dataset enables comparative analyses across different lanthanoids and serves as a platform for evaluating the performance of QC methods on distinct lanthanoid complexes and set up machine learning studies in the future. Upcoming studies might explore different avenues such as extending the dataset, adjusting initial oxidation states, or modifying molecular charges. Also the incorporation of additional ligands into the generation setup could yield further insights, particularly for metallic

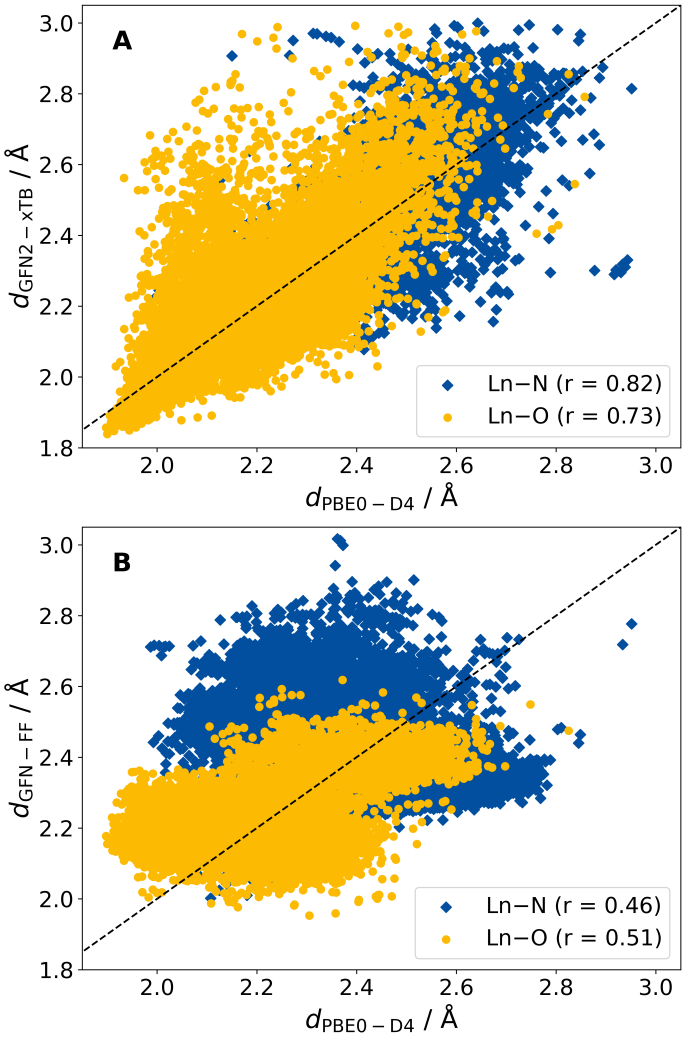

Figure A.9: Bond lengths d_X(Ln-{N, O}) using geometry optimization on GFN2-xTB (A) and GFN-FF (B) level compared to PBE0-D4/def2-SVP level.

complexes or experiment-specific setups. Moreover, the inclusion of multi-lanthanoid structures could shed light on lanthanoid-lanthanoid interactions. In more detailed analyses, the inclusion of spin-orbit coupling might be considered. Introducing spin state screening could be implemented to reliably identify ground states. This might currently pose an issue if the ground state is not identified correctly. Further limitations arise from the semi-automated Architector generation of structures. While we strive for a balanced representation in ligand structures, subtle biases, such as more frequent occurrences of ligands like ammonia or cyanide, are challenging to circumference. Potential biases may also stem from the limited diversity of ligand motifs, which cannot fully represent "all" real-world chemical environments. The LnQM is designed to serve as a starting point for continued exploration in the field of lanthanoid research.

Data availability The dataset is available at https://github.com/grimme-lab/lnqm. Please refer to the information available there for instructions on setting up and loading the data. The dataset is stored in HDF5 format. To facilitate access, we provide a Python module for reading the data. Furthermore, molecular geometries are accessible in xyz and coord file formats.

Acknowledgement

The authors thank Dr. Jan-Michael Mewes for fruitful discussions. AI-based methods were used to assist in the development of the body text. This work was supported by the DFG in the framework of the SPP 2363 on “Utilization and Development of Machine Learning for Molecular Applications - Molecular Machine Learning”. S.G. and M.B. gratefully acknowledge financial support by the Max Planck Society through the Max Planck fellow program.

APPENDIX B

ConfRank: Improving GFN-FF Conformer Ranking with Pairwise Training

Christian Hölzer,[*] Rick Oerder,[† ‡] Stefan Grimme,[*] and Jan Hamaekers[‡]

Received 27 August 2024, Published online 20 November 2024.

Reprinted in Appendix B (adapted) with permission[§] from
C. Hölzer, R. Oerder, S. Grimme and J. Hamaekers, *ConfRank: Improving GFN-FF Conformer Ranking with Pairwise Training*, J. Chem. Inf. Model. **64** (2024) 8909, DOI: 10.1021/acs.jcim.4c01524.

[*] Mulliken Center for Theoretical Chemistry, University of Bonn, Beringstr. 4, 53115 Bonn, Germany
[†] Institute for Numerical Simulation, Friedrich-Hirzebruch-Allee 7, 53115 Bonn, Germany
[‡] Fraunhofer Institute for Algorithms and Scientific Computing SCAI, Schloss Birlinghoven 1, 53757 Sankt Augustin, Germany

Abstract Conformer ranking is a crucial task for drug discovery, with methods for generating conformers often based on molecular (meta)dynamics or sophisticated sampling techniques. These methods are constrained by the underlying force computation regarding runtime and energy ranking accuracy, limiting their effectiveness for large-scale screening applications. To address these ranking limitations, we introduce ConfRank , a machine learning-based approach that enhances conformer ranking using pairwise training. We demonstrate its performance using GFN-FF-generated conformer ensembles, leveraging the DimeNet++ architecture trained on pairs of 159 760 uncharged organic compounds from the GEOM dataset with r^2SCAN-3c reference level. Instead of predicting only on single molecules, this approach captures relative energy differences between conformers, leading to a significant improvement of the overall conformational ranking, outperforming GFN-FF and GFN2-xTB. Thereby, the pairwise RMSD of the relative energy difference of two conformers can be reduced from 5.65 kcal mol^{-1} to 0.71 kcal mol^{-1} on the test dataset, allowing to correctly identify up to 81 % of all lowest lying conformers correctly (GFN-FF: 10 %, GFN2-xTB: 47 %). The ConfRank approach is cost-effective, allowing for scalable deployment on both CPU and GPU, achieving runtime accelerations by up to two orders of magnitude compared to GFN2-xTB. Out-of-sample investigations on CREST-generated conformer ensembles from the QM9 dataset and conformers taken from an extended GMTKN55 dataset show promising results for the robustness of this approach. Thereby, ranking correlation coefficient such as Spearman can be improved to 0.90 (GFN-FF: 0.39, GFN2-xTB: 0.84) reducing the probability of an incorrect sign flip in pairwise energy comparison from 32 % to 7 %. On the extended GMTKN55 subsets the pairwise MAD (RMSD) could be reduced on almost all subsets by up to 62 % (58 %) with an average improvement of 30 % (29 %). Moreover, an exemplary case study on vancomycin shows similar performance, indicating applicability to larger (bio-)molecular structures.

Furthermore, we motivate the usage of the pairwise training approach from a theoretical perspective, highlighting that while pairwise training can lead to a decline in single sample prediction of absolute energies for ML models, it significantly enhances conformer ranking performance.

The data and models used in this study are available at https://github.com/grimme-lab/confrank.

B.1 Introduction

In drug research, understanding and identifying conformational isomerism is crucial as the biological activity of a molecule significantly depends on its precise three-dimensional geometry when interacting with biological targets, such as enzymes or receptors [308–310, 385–387]. Hence, developing methods to identify structural and energy differences between conformers is highly important. Thereby, pinpointing the energetically most favorable conformers is challenging due to the often relatively small energy differences (on the order of thermal energy at room temperature) and a large number of conformations [388–392]. Moreover, identifying the lowest-lying conformer in an ensemble of conformers (”stabilomer”) is particularly relevant, as the lowest energy conformer is generally the most abundant and relevant form of a molecule under physiological conditions [393–395] and the predicted effectiveness and specificity of a drug can heavily depend on the most abundant conformer [396–400]. Therefore, understanding and predicting conformational behavior may lead to the development of drugs with better efficacy, reduced side effects, and improved pharmacokinetic properties. As such, conformational analysis is a key component in the optimization of therapeutic agents and the rational design of new drugs. Furthermore, conformers are relevant in quantitative structure-activity

relationship analyses [401] in the context of biodegradability [402, 403], industrial applications [404–409], and consumer goods [410–413].

Software designed to generate conformational libraries for a large number of compounds needs to be computationally efficient while producing high-quality geometries. These models are required to provide a broad conformational coverage, representing each compound with diverse, non-redundant, and physically plausible conformers. Conformer ensembles can be generated through systematic, stochastic, or machine learning (ML)-based methods [414–416].
In the last years, several methods for generating conformer ensembles based on ML models have been developed, including ones leveraging supervised learning [417, 418], unsupervised learning [419–421], and reinforcement learning [422–425]. Furthermore, recent advancements in diffusion models have led to the creation of multiple implementations for conformer generation [426–428].
Systematic methods employ rule-based conformed generation such as rotations around bonds with fixed angles [429–431] or more elaborated methods i.a. using allowed paths to constrain the search space [432] or other knowledge bases [433–435]. Other physics- and heuristics-based conformer generation methods include the open-source cheminformatics library RDKit, which utilizes distance geometry algorithms for generating small-molecule conformers [436]. Additionally, commercial software such as OpenEye's OMEGA [437, 438] employs a torsion-driving approach for conformer generation.
Various methods explore the conformational space using stochastic methods, often employing physically inspired approaches such as surface collision [439, 440], Monte Carlo-sampling [441] or using (accelerated) molecular-dynamics [311, 442, 443].

Among this plethora, the meta-dynamics based Conformer-Rotamer Ensemble Sampling Tool (CREST) [115, 312, 313, 444, 445] has established itself as one popular method for ensemble generation due to its robustness and effectiveness in finding and generating relevant conformers [446].

For conformational analysis, two steps are essential: generation of conformer structures and energetic ranking of these structures. The latter requires a) accurate conformational energies at b) fast inference timings such that large-scale scanning applications become feasible. Often heuristics or computationally fast methods such as classical force fields are employed for this task [447–449]. In recent years multiple ML-based ab initio force fields were introduced [8, 157, 450]. Albeit yielding sufficient computational speed, classical force fields often lack the accuracy needed for the precise identification of energetically relevant conformers.

Therefore, conformer generators such as CREST are often concluded with tools like a Commandline ENergetic SOrting of Conformer Rotamer Ensembles (CENSO) [116, 316] pipeline to refine ensembles at a higher theoretical level, usually different levels of Density Functional Theory (DFT). Following different optimization and energy calculations steps, the selection of conformers is narrowed down to identify the lowest lying conformer. Thereby, major causes of high computational costs are incorrect geometries or inaccurate energy rankings, and overall computational costs can be significantly reduced if the initial energetic ranking is improved. An improved energetic sorting allows fewer conformer candidates to be evaluated using higher-level DFT functionals, which are computationally demanding.
Research on improving geometries during conformer generation, such as through specialized force fields and ML potentials, is ongoing. In this study, the focus is placed on improvements for the relative energy prediction. Even though various ML models demonstrate promising results in this context [7, 8, 157, 451, 452] and show promising outcomes when inferring ensemble data instead of single data [453], to the best of the authors' knowledge, few research has been conducted on pairwise comparisons of conformers using ML methods so far.

In this research, a ML training ansatz is developed to improve upon energetic ranking using existing ML architectures. For this purpose, the training objective of a ML model is shifted towards predicting relative energies for a given pair of conformers. We exemplarily show that this approach can be applied to various ML architectures to obtain DFT-like energy rankings at force field cost. In the following section C.2, the theoretical motivation for the specifics of our pairwise training workflow is presented. Subsequently, in section B.3 a short overview of the technical setup is given. Section B.4 describes the training data and curation steps. In the first part of section C.4, the ranking performance of ML methods is compared to conventional quantum chemistry (QC) methods, in the later part a comparison of pairwise and pointwise training procedures is conducted. For this purpose, different datasets are curated, spanning from GEOM and QM9 to GMTKN55. For all samples, references at r^2SCAN-3c level are provided and a trained DimeNet++ model is made available. Section B.6 concludes with a discussion on potential future research directions such as targeting broader areas of chemical space.

B.2 Theory and Methods

The following section features a formal mathematical problem statement for conformer ranking, focusing on pairwise training. A brief discussion of the machine learning models evaluated in this study is provided, followed by an overview of the GFN-FF theory and background, which serves as the baseline method.

B.2.1 Conformer Ranking with Machine Learning

Problem statement Our goal is the ranking of conformers in an ensemble with respect to their conformation energy. For a specific molecule, we are given an ensemble of N molecular conformers $\left\{\mathbf{X}^{(i)}\right\}_{i=1}^{N}$. The i-th molecular conformer of an uncharged molecule with n atoms is given as

$$\mathbf{X}^{(i)} = \left\{\left(\mathbf{r}_j^{(i)}, Z_j\right)\right\}_{j=1}^{n}, \tag{B.1}$$

where $\mathbf{r}_j^{(i)}$ denotes the Cartesian coordinates of the j-th atomic nuclei and Z_j is the respective atomic number.

Essentially, one can view the total energy as a function $E_{tot}^{(i)} = f\left(\mathbf{X}^{(i)}\right)$ and approximate it, for example, by a statistical model and subsequently rank the conformers with respect to the prediction. The rise of Machine Learning Interatomic Potentials (MLIPs) over the past years together with advances of hardware and in particular accelerators such as GPUs has lowered the bar for the development of highly accurate, yet fast surrogate models. Inspired by the growing availability of powerful machine learning architectures that have shown their ability to model large datasets, we decide to tackle the problem of high-throughput conformer ranking by tailoring such a model to our specific application.

Pairwise approach A key challenge in conformer ranking is that we are mainly interested in potentially very small ($\lesssim 1\,\mathrm{kcal\,mol^{-1}}$) relative energy differences between conformers of the same chemical composition and covalent bond topology, whereas chemically different ensembles might differ in their total energies by orders of magnitude. A natural solution to overcome this difficulty is to consider the relative energy of two conformers i and j, rather than their respective total energies:

$$E_{rel}^{(i,j)} = E_{tot}^{(i)} - E_{tot}^{(j)} \tag{B.2}$$

In this work, we approximate $E_{rel}^{(i,j)}$ by a neural network based model f_θ

$$E_{rel}^{(i,j)} \approx f_\theta\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}\right), \tag{B.3}$$

where θ denotes the set of trainable parameters. In particular, we investigate the process of minimizing a loss function $\mathcal{L}$ over a dataset $\mathcal{D}$ containing relative energies $E_{ref}^{(i,j)}$ for pairs of conformers as computed by a reference method, i.e. finding an optimal set of model parameters θ^* via

$$\begin{aligned}\theta^* = \arg\min_\theta \quad & \mathop{\mathbb{E}}_{\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}, E_{ref}^{(i,j)}\right)\sim\mathcal{D}} \\ & \mathcal{L}\left(E_{ref}^{(i,j)}, f_\theta\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}\right)\right).\end{aligned} \tag{B.4}$$

In contrast to the *pointwise* training process that is common for most machine learned interatomic potentials, equation (B.4) minimizes a loss function that depends on pairs of data points, hence we are in the setting of *pairwise learning* [454–456]. As a result of learning differences, special importance is put on (geometric) changes that lead to energetic changes within an ensemble. On the other hand, contributions that are more or less constant within the same ensemble cancel out in equation (B.2). Note that machine learning models for ranking have been investigated in various contexts before [457–462]. Conceptually, our work is closest to the idea of *pairwise difference regression* [463] and *twinned regression* [464, 465].

Symmetry requirements The prediction of relative energies needs to satisfy certain consistency requirements in order to be meaningful from both a physical perspective and a technical perspective. That is, for any two pairs i and j of data points or respectively for any triplet i, j and k our predictive model f_θ needs to fulfill

$$\texttt{I} \;:\; f_\theta\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}\right) + f_\theta\left(\mathbf{X}^{(j)}, \mathbf{X}^{(i)}\right) = 0 \tag{B.5}$$

$$\begin{aligned}\texttt{II} \;:\; f_\theta\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}\right) + f_\theta\left(\mathbf{X}^{(j)}, \mathbf{X}^{(k)}\right) \\ + f_\theta\left(\mathbf{X}^{(k)}, \mathbf{X}^{(i)}\right) = 0.\end{aligned} \tag{B.6}$$

The first condition simply implies that the prediction should change its sign when commuting the arguments (antisymmetry). The second condition guarantees that the prediction for the relative energy of (k, i) is consistent with the prediction for (i, j) and (j, k) (transitivity). More general, the predictive model f_θ should yield consistent predictions also for arbitrary n-tuples of data points which is referred to as higher order loop consistency. As outlined by Wetzel et al. [464], conditions (B.5) and (B.6) are sufficient for higher order loop consistency. Wetzel et al. [464] suggest using a pairwise neural network architecture and enforcing the consistency requirements by additional regularization terms in the loss function. At inference, they consider violations of loop consistency as a measure for uncertainty of the model prediction. However, in the spirit of Informed Machine Learning [466], we include these constraints directly into the model architecture, allowing for exact fulfillment of loop consistency. To that end, we will achieve 2- and 3-loop consistency by decomposing the pairwise model as

$$f_\theta\left(\mathbf{X}^{(i)}, \mathbf{X}^{(j)}\right) = g_\theta\left(\mathbf{X}^{(i)}\right) - g_\theta\left(\mathbf{X}^{(j)}\right). \tag{B.7}$$

This choice seems very natural, and indeed we argue in the Supporting Information that this is the only way to build a pairwise architecture that satisfies the loop consistency relations.

Additionally, we assume f_θ to be invariant with respect to translations and rotations of the molecule and invariant with respect to permutations of atom indexing. Hence, it is a natural choice to employ an architecture from the vast pool of ML models for atomistic systems that have been developed over the past years [157, 239, 240, 260, 266–268, 278, 314, 450, 467–470] as g_θ. Another nice feature of decomposing f_θ according to equation (B.7) is that, despite being trained in a pairwise fashion, at inference, new structures can be evaluated in a pointwise manner by simply applying g_θ. Therefore, in practice, we still deal with model architectures that operate on single samples but train them in a pairwise fashion. However, the output of g_θ does not necessarily equate to the actual potential energy but rather to some *pseudo-energy* that can be used for computing pairwise differences during training and for direct ranking at inference time. Intuitively, we expect that during the pairwise training process, g_θ is optimized to account for differences within ensembles, while being less sensitive to dependencies that are more or less constant among conformers of the same ensemble. If g_θ was a classical force field with bond, angle, torsion energy terms etc. we can imagine locally constant contributions to cancel out in equation (B.7), potentially leading to a less complex loss function landscape. This becomes especially prevalent in the context of conformer search, where isolated atom energies, covalent bond energies etc. cancel out (almost) entirely. In the Supporting Information, we describe how pseudo-energies can be related in part to the actual potential energies by performing a linear fit of constant per-atom energy contributions.

B.2.2 Machine Learning Interatomic Potentials

Machine Learning Interatomic Potentials are ML models that predict the potential energy surface (PES) of atoms in a system. Traditional methods such as QC can be highly accurate but computationally expensive. MLIPs aim to provide a balance by offering accuracy close to that of ab initio computations at a fraction of the computational cost. [471] They are trained on data generated from high-level QC calculations or (less often) experimental data and can be used to simulate large systems over longer timescales. Common applications of MLIPs include material science or drug design [472].

Most MLIPs are designed to achieve a linear scaling behavior in terms of the system size by representing the predicted value as a sum over atom-wise contributions [6]. Of particular importance for this work are Graph Neural Networks (GNNs), which often construct these atomic contributions from learned node features that result from the training process. Many GNNs used for the prediction of atomistic properties employ a variant of *message passing* [473] and thus belong to the group of Message Passing Neural Networks (MPNNs). The concept of message passing allows for propagating information between nodes, edges, etc. along the molecular graph, that is usually specified based on inter-atomic Euclidean distances and a cutoff radius. As a result, MPNNs can be used to model both short-range dependencies and long-range dependencies up to the boundary of their *receptive field* while retaining their linear scaling behavior. The size of the receptive field can usually be adjusted by choosing the hyperparameters of a specific MPNN architecture appropriately. An advantage of modern GNN architectures is that they are usually agnostic to the specific dataset they are trained on and are often ready to use, requiring almost no feature engineering. During the training process

functional dependencies with respect to the molecular geometry, such as dependencies of the energy on bond stretching or bending, are learned by the network. Note that only atomic coordinates and atom identifiers are taken as an input.

We want to stress the vast amount of different model architectures available through publications and open-source software. Although there are publications [471, 474–477] providing an overview of different architectures, it is usually not obvious which architecture is the best for a specific problem *a priori*. For practical reasons, we restrict ourselves to only a few different MPNN architectures. In the following, we give a short overview of architectures used in this work.

SchNet [314] employs a continuous-filter convolutional neural network architecture to model atomic interactions, thus transferring the *convolution* concept from images to molecular graphs. Its architecture enables the efficient capture of both local chemical environments and long-range interactions by propagating information across the molecular graph.

DimeNet++ [267] is a MPNN architecture designed to predict molecular properties by leveraging directional message passing. It is an enhancement of the original DimeNet [266], improving both performance and computational efficiency. Internally, a radial Bessel basis and a 2D spherical Fourier-Bessel basis expansion are employed to encode information on interatomic distances and angular dependencies of nodes. DimeNet++ was one of the first GNNs to incorporate directional information in its message passing process, allowing for improved accuracy.

GemNet [268] is another architecture based on DimeNet++. In contrast to DimeNet++ it allows for the explicit description of 4-body interactions (GemNet-Q), although using 3-body information exclusively is possible as well (GemNet-T). Building on spherical representations the authors show that GemNet is a universal GNN, hence being able to approximate any continuous function of a point cloud that is invariant to (global) rotations, translations and permutations. This theoretical result in combination with diverse training data makes GemNet a promising candidate for accurate modeling of molecular properties in a wide area of applications.

MACE [240] [478] has its theoretical foundation in the atomic cluster expansion [479] and builds on $SE(3)$- and $E(3)$-equivariant modeling [480] in combination with higher order message passing. In particular, the message passing process builds on a hierarchical body order expansion. The largest body order that is taken into account is treated as a hyperparameter providing the possibility – in addition to other hyperparameters – to balance between computational cost and performance.

B.2.3 GFN-FF

The GFN-FF is a multi-purpose force field in the GFN method family that enables accurate and efficient predictions of molecular **g**eometries, vibrational **f**requencies, and **n**on-covalent interactions [16]. The GFN-FF method handles π-conjugated systems by an iterative Hückel scheme. Covalent-bonding information, atomic charges, and bond orders are calculated from the input molecular structure. The covalent topology is based on inter-atomic distances R_{AB} compared to a reference value R^0_{AB}, calculated as:

$$R^0_{AB} = (R^0_A + R^0_B + R_{sft})(1 - \sum_i c_i |\Delta EN|^i) \tag{B.8}$$

where coordination number dependent R^0_A and R^0_B are derived from D3 damping radii [481], R_{sft} is

an element-specific shift, c_i are fitted parameters and Pauling atomic electronegativities EN are taken. A covalent bond is assigned if $R_{AB} < f_R(q)R^0_{AB}$, with $f_R(q)$ a charge-dependent scaling function. Atoms are assigned hybridization states based on neighbors, identifying π-conjugated fragments and assigning π-electrons. Initial topology-based electronegativity-equilibrium (EEQ)[373, 482] charges q_{eeq} are derived, and the procedure is iterated to obtain an improved topology.

The total energy is given by the sum of covalent interactions and non-covalent interactions $E_{\text{GFN-FF}} = E_{\text{CI}} + E_{\text{NCI}}$, with the covalent part featuring Gaussian-type potentials and three-body bonding corrections. The non-covalent part employs the EEQ atomic charge model for electrostatics, a topology-based D4 scheme for dispersion [302], and charge-dependent corrections for hydrogen and halogen bonds. Energy contributions are composed of the sum of two- and three-particle energies covalent bonding E_{bond}, angular bending E_{bend}, repulsion $E^{\text{CI}}_{\text{rep}}$, rotational torsion E_{tors} and three-body repulsion correction E_{abc} for the covalently bonded part. The intra- and inter-molecular non-covalent contribution ("non-bonded") is given by the sum of isotrophic electrostatics E_{IES}, dispersion E_{disp}, Pauli repulsion $E^{\text{NCI}}_{\text{rep}}$, as well as hydrogen bridge E_{HB} and halogen bridge bond E_{XB} correction terms [16]:

$$E_{\text{CI}} = E_{\text{bond}} + E_{\text{bend}} + E^{\text{CI}}_{\text{rep}} + E_{\text{tors}} + E_{\text{abc}} \tag{B.9}$$

$$E_{\text{NCI}} = E_{\text{IES}} + E_{\text{disp}} + E^{\text{NCI}}_{\text{rep}} + E_{\text{HB}} + E_{\text{XB}}\,. \tag{B.10}$$

Although there are a plethora of other force fields (UFF, AMBER, etc.) [447–449], GFN-FF is highly suitable for the generation of conformer ensembles due to its focus on geometries and non-covalent interactions. Moreover, GFN-FF includes parametrization for the full periodic table for elements up to radon ($Z \leq 86$), allowing for a broad applicability in chemistry. Furthermore, GFN-FF is available in CREST [312]. Note that, as the name "GFN" implies, GFN-FF is particularly suitable for producing reasonable geometries at force field level. However, it was not designed to yield very accurate conformational energies and performs in this respect similar to other "traditional" FFs like Amber or MM3 [449, 483]. In this study, we improve upon this aspect.

B.3 Technical Setup

All quantum mechanical DFT calculations were performed using the ORCA computational chemistry software suite (version 5.0.4) [40, 356]. Following general recommendations for selecting DFT functionals [86], reference calculations were conducted using the r^{2}SCAN-3c functional [315]. This composite electronic-structure functional includes a modified version of the semi-classical D4 dispersion model [301, 302, 484, 485] and a geometrical counter-poise correction [51]. The default convergence criteria were applied, with the DEFGRID2 option for the DFT integration grid. To compare with semi-empirical quantum methods, the GFN2-xTB method [18, 148] and for force field calculations, the GFN-FF method was used [16], utilizing xtb version 6.6.1 [486]. Conformer generation was conducted using CREST [115, 312, 313, 444, 445] version 2.12 and 3.0.2 [487], employing the default configuration and GFN-FF for the underlying PES. All scripts are written in Python 3.11.5 [322, 488] using PyTorch [323] version 2.1.0 [489]. Dimenet++ and SchNet model architectures were directly imported from PyG (PyTorch Geometric) version 2.5.0 [490]. The MACE architecture used in this work was imported from the mace_torch package version 0.3.5 [491]. The

code from the GemNet architecture was copied from the gemnet_pytorch repository [492] and the included code for transforming training data into the required format was slightly modified to allow for integration into our framework. For more details on the software versions used, see the setup description in the GitHub repository of this work. Data is stored on disk in the hierarchical data format (HDF5) [364]. Unless explicitly mentioned otherwise, calculations were performed on Intel Xeon "Sapphire Rapids" 48-core/96-thread 2.10GHz CPUs. Training of machine learning models was performed on a single Nvidia RTX3070 with 8GB VRAM.

B.4 Dataset and Preprocessing

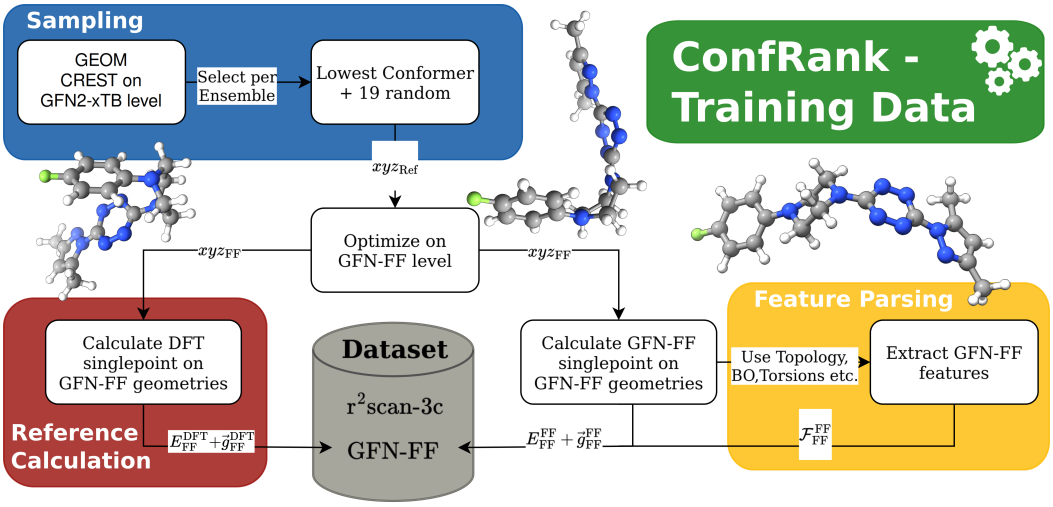

Figure B.1: Generation of ConfRank training data involves three main steps: sampling, reference calculation, and feature parsing. During sampling, 20 conformers are randomly selected from each ensemble, including the lowest energy conformer at the reference energy level. The selected conformers are then optimized on GFN-FF level. Subsequently, the training targets, energies and gradients, are determined using DFT calculations on the GFN-FF geometries. Additionally, a GFN-FF singlepoint calculation is performed on the GFN-FF equilibrium geometries, and further features from the GFN-FF calculation are parsed. Energy and gradient values from both the r^{2}SCAN-3c reference and GFN-FF calculations are incorporated into the dataset, with nomenclature indicating geometry level (subscripts) and singlepoint level (superscripts).

For training the ML model, a subset of the GEOM dataset [171], encompassing 37 million molecular conformations across 450 000 molecules was employed in this study. The published GEOM dataset is characterized by its CREST-generated equilibrium structures and its focus on organic compounds, making it well-suited for the target purpose in this study. The dataset includes 317 000 conformers for molecules related to experimental data, e.g. molecules found by in-vitro high-throughput screening approaches [493].

The workflow for data generation and preprocessing is depicted in Figure B.1. To ensure a broad representation of chemical diversity and improve generalization across organic chemical space, the number of samples per conformer ensemble was limited to a fixed amount, which allows the model to encounter a wide variety of molecules while optimizing computational resources and avoid over- or undersampling of molecules for which a large or respectively small number of conformers is available.

From the molecules related to experimental data, we randomly select uncharged ensembles with a number of unique conformers $n_{\text{conf}} > 20$. Subsequently, per ensemble, the lowest DFT-ranked conformer is sampled alongside 19 other conformers at random.

Samples included in the GEOM dataset are CREST structures optimized on GFN2-xTB level. In order to train on geometries while being consistent regarding the level of theory, which is compatible with common CREST output, all samples are optimized on GFN-FF level before further singlepoint calculations are conducted. As reference labels for training, singlepoint total energies and gradients are calculated on r^{2}SCAN-3c level and included in the dataset. The r^{2}SCAN-3c DFT functional was chosen due to its generally good performance on organic conformers and excellent cost-accuracy ratio [86, 315, 494]. Note that explicitly providing GFN-FF features as input into existing ML architectures did not yield improvements during training. This can be attributed to the fact that these aforementioned ML architectures were designed with a focus on geometrical input and feature learning, implying that relevant features are learned internally and not given explicitly as input. For completeness, GFN-FF singlepoint features such as topology, bond orders, angle- and torsion- force-constants and EEQ atomic charges are included in the dataset, allowing for the possibility of stratification and exploratory data analysis. For future research avenues, we provide this data free of charge at `https://zenodo.org/records/13354132`.

Moreover, GFN2-xTB energies and gradients are calculated as well as energies of various ML models. To prevent overlap of training and test sets, particularly in the context of investigating datasets from different sources as in section C.4, duplicates were identified and removed using the MolBar identifier [495]. MolBar hereby identifies constitutional, configurational as well as conformational isomerisms.

Following this procedure, one arrives at a dataset of 8 986 randomly picked ensembles containing 179 720 conformers - in the following signified as "ConfRank" dataset. An overview over the content of the ConfRank dataset is given in Figure B.2. For training 7 349 ensembles are used, keeping 639 ensembles for validation and 998 for testing. It is noteworthy that only 2 % of the ensembles from the initial GEOM dataset are sampled. Nonetheless, we deem the resulting model performances (c.f. subsection B.5.2) to be sufficient for our purposes, while keeping the computational effort for r^{2}SCAN-3c reference calculations minimal, and disregard the option to sample more data. The dataset comprises organic drug-like molecules including biologically active functional groups and medium-sized compounds up to 145 atoms. An overview on the elemental composition and the distribution of the number of atoms is shown in Figure B.2. Most conformation energies are below 10 kcal mol^{-1}, making them particularly challenging to resolve, especially for force field methods [496]. The distribution of relative energies is given in the Supporting Information.

B.5 Results

B.5.1 Energetic Improvement

In order to obtain an overview over the capabilities of existing MLIPs, energetic improvements are compared for existing force field and semiempirical quantum methods (GFN-FF and GFN2-xTB) [16, 18, 148] with a selection of before-mentioned ML architectures: SchNet, DimeNet++, GemNet-T[¶]

[¶] We found the implementation of GemNet-Q to be too slow in the training process and hence, for practical reasons, collected results for GemNet-T.

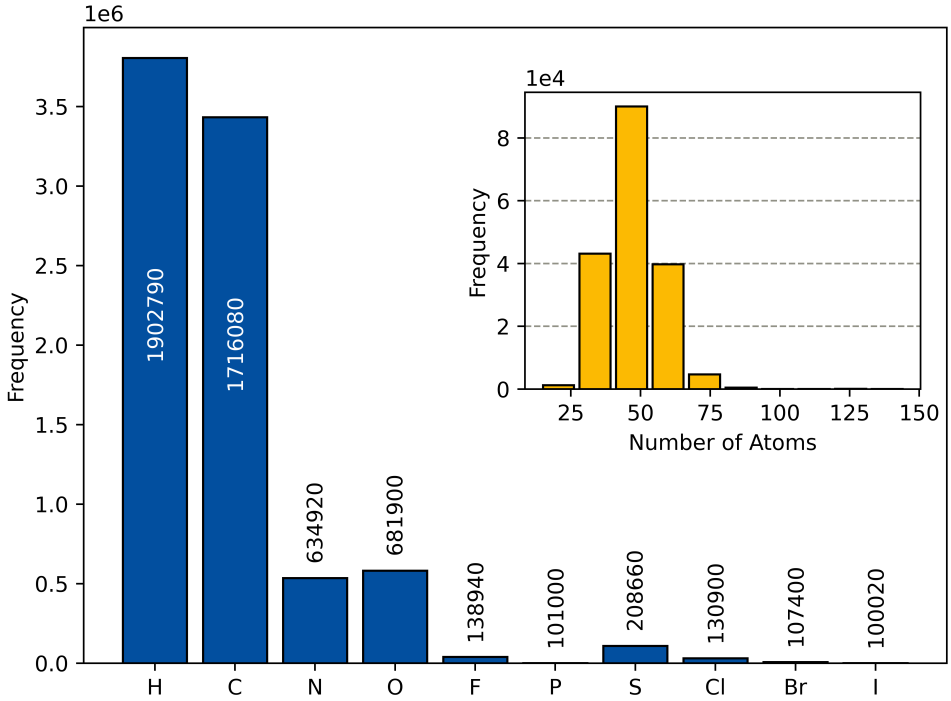

Figure B.2: Overview of the composition of the ConfRank dataset. The main plot displays the elemental composition of all samples, with absolute values indicated on each column. The inset illustrates the distribution of molecular sizes in the dataset.

and MACE.

For training and testing the preprocessed ConfRank dataset is employed (c.f. section B.4).

To allow for a fair comparison, all models are trained from randomly initialized weights on the same training dataset, are evaluated on the same validation dataset (during training) and compared on the same test set (after training). All of these subsets were selected randomly.

To that end, we adapt the respective open-source implementations to our specific training framework. Each architecture's hyperparameters are selected such that the number of trainable parameters is comparable (ranging from 100k-150k for all models) and such that the receptive fields of the models are equal. The latter is achieved by setting the number of message passing layers to 3 and the cutoff radius of each layer to 4 Å. Hence, in total the ML models' receptive fields cover a sphere with a radius of 12 Å. r^2SCAN-3c total atomic ground state energies are used to compute a constant energy contributions of the atoms in the molecule allowing for a fit of total energies. More information on the training process and hyperparameters can be found in the Supporting Information.

As evaluation metrics, regular statistical measures such as MAD, RMSD, etc. as well as ensemble-averaged correlation coefficients are evaluated. Details on the metrics and their calculation are given in the Supporting Information. Furthermore, to emphasize ranking capabilities in the context of conformer identification, we included the following metrics:

- The *sign flip probability* (SF) indicates the percentage of wrong sign assignment, i.e. $\text{sign}(E_A - E_B) \neq \text{sign}(E_A^{\text{DFT}} - E_B^{\text{DFT}})$.

- The *ranking accuracy* (Top$X$) denotes the probability of identifying the "true" lowest conformer at DFT-level within the X lowest predicted samples using the reference method.
- The *energy window accuracy* (EWX) represents the probability of finding the "true" lowest conformer at DFT-level within a given energy window X in kcal mol^{-1} to the energy of the lowest predicted sample.

An overview over the performance of different QC methods and ML models is given in the top rows of Table B.1 and Table B.2.

	MD$_\downarrow$	**MAD$_\downarrow$**	**RMSD$_\downarrow$**	R^2	ρ_p	ρ_s	τ_K
GFN-FF	0.54	4.08	5.65	-0.79	0.07	0.07	0.05
GFN2-xTB	0.02	1.90	2.59	0.62	0.81	0.75	0.62
DimeNet++ (single)	0.16	1.16	1.64	0.85	0.92	0.88	0.76
DimeNet++ (pair)	0.05	0.49	0.71	0.97	0.98	0.97	0.90
GemNet-T (single)	0.20	1.13	1.63	0.85	0.92	0.88	0.76
GemNet-T (pair)	0.02	0.46	0.66	0.98	0.99	0.97	0.90
SchNet (single)	0.21	1.61	2.38	0.68	0.85	0.81	0.67
SchNet (pair)	0.19	0.99	1.44	0.88	0.94	0.90	0.79
MACE (single)	0.41	1.65	2.29	0.70	0.84	0.80	0.65
MACE (pair)	0.07	0.56	0.83	0.96	0.98	0.96	0.88

Table B.1: Statistical performance metrics for different models and training modes on the test set. MD, MAD and RMSD of relative energies are given in kcal mol^{-1}. The coefficient of determination (R^2) and rank correlation metrics Pearson ρ_p, Spearman ρ_s and Kendall τ_K are given in absolute values. Correlation coefficients are ensemble averaged over pointwise energy predictions. Metrics annotated with a $\downarrow$ indicate that lower values are better, otherwise, higher values are preferred.

The GFN-FF model exhibits significant inaccuracies in ranking tasks, as indicated by the highest error metrics recorded among the evaluated models. Specifically, the statistical measures are extraordinarily high, with an MAD of 4.08 kcal mol^{-1} and RMSD of 5.65 kcal mol^{-1}. The high variance and negative coefficient of determination (R^2) underscore the substantial uncertainty associated with using the GFN-FF model for conformer ranking. These uncertainties support the claim that, despite its fast runtime, the GFN-FF model alone is only suitable with large energy windows for conformer ranking tasks (c.f. EW5 in Table B.2). Furthermore, the high SF rate of 0.47 effectively equates to a random guess when comparing two conformers with the GFN-FF model. The very low ranking accuracy and energy window accuracy further highlight the high rate of incorrectly assigned lowest-energy conformers. Only within an energy window of 5 kcal mol^{-1} can a substantial percentage of the lowest-energy conformers be reliably considered.

The semiempirical GFN2-xTB model significantly improves upon many of the deficiencies observed with the GFN-FF model, justifying its default setting in CREST despite its higher runtime. Nearly all statistical measures show improvements of more than 50 % compared to GFN-FF. The notably low MD of 0.02 kcal mol^{-1} compared to the MAD of 1.90 kcal mol^{-1} indicates symmetric results around zero. Despite showing good correlation in conformational energies ($\rho_p = 0.77$) and ranking ($\rho_s = 0.75$), the predictive power of GFN2-xTB remains improvable for reliable conformer ranking.

	SF$_\downarrow$	Top1	Top3	Top5	EW1	EW3	EW5
GFN-FF	0.47	0.10	0.21	0.33	0.29	0.57	0.77
GFN2-xTB	0.19	0.47	0.73	0.86	0.74	0.94	0.99
DimeNet++ (single)	0.12	0.61	0.88	0.95	0.88	0.99	1.00
DimeNet++ (pair)	0.05	0.81	0.97	1.00	0.99	1.00	1.00
GemNet-T (single)	0.12	0.61	0.89	0.95	0.89	0.98	1.00
GemNet-T (pair)	0.05	0.82	0.98	1.00	0.99	1.00	1.00
SchNet (single)	0.17	0.49	0.79	0.89	0.79	0.95	0.98
SchNet (pair)	0.11	0.66	0.90	0.96	0.93	0.99	1.00
MACE (single)	0.17	0.45	0.73	0.86	0.77	0.95	0.99
MACE (pair)	0.06	0.79	0.96	0.99	0.98	1.00	1.00

Table B.2: Advanced performance metrics for different models and training modes on the test set. Sign flip probability (SF), ranking accuracy (Top$X$), energy window accuracy (EWX, with X in kcal mol^{-1}) are given in %. Ranking accuracy and energy window accuracy are evaluated on pointwise energy predictions. Details on the metrics can be found in the main text. Metrics annotated with a $\downarrow$ indicate that lower values are better, otherwise, higher values are preferred.

Aside from GFN-FF's high errors and GFN2-xTB as a more reliable alternative, ML models trained on highly accurate data from QC calculations in a pointwise fashion (referred to as "single") demonstrate superior performance. Models such as DimeNet++, GemNet-T, SchNet and MACE outperform both GFN-FF and GFN2-xTB regarding error metrics and correlation. These pointwise trained models achieve RMSD values between 1.63 and 2.38 kcal mol^{-1}, along with R^2 values ranging from 0.68 to 0.85. Special emphasis can be paid to the best-performing ML models DimeNet++ and its successor GemNet-T. Those display very good ranking capabilities ($\rho_p = 0.92$), and allow for tight energy windows (EW1 > 0.88). However, these "single" sample trained models do not match the performance levels of their "pair"wise trained model counterparts, presented in the following section.

B.5.2 Pairwise Training

At the core of the ConfRank training workflow resides the pairwise training approach minimizing a pairwise loss function in the sense of equation (B.4).

Instead of running backpropagation through single entities as is most common in supervised machine learning, model weights are updated by minimizing a loss function depending on pairs of data points. Thereby, pairs of conformers within each ensemble are sampled, adapting the model weights to minimize the relative differences between conformers, and better capture subtle variations among conformers, leading to more accurate ranking predictions (see below).

Setup An illustrative depiction of the training setup is provided in Figure B.3. By training on relative energies, the persistent issue of different energy scales between various method classes can be overcome (esp. between GFN and DFT methods). We minimize the $L1$-loss between predicted and actual relative energies according to DFT:

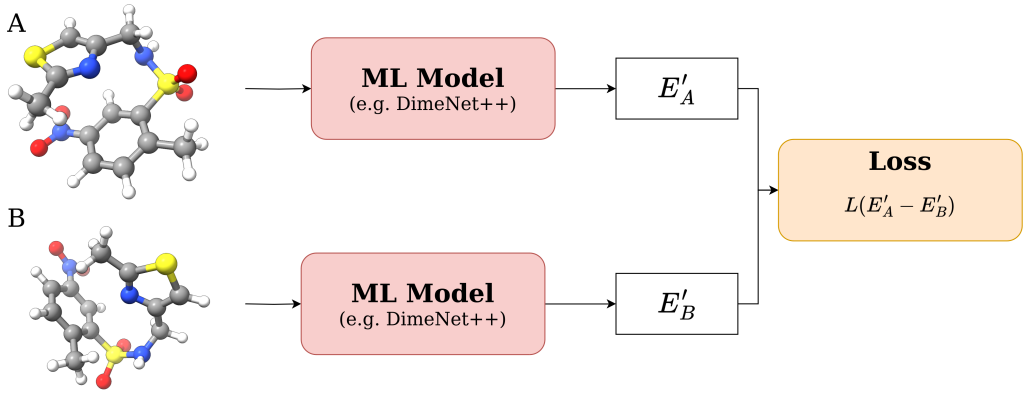

Figure B.3: Schematic approach for pairwise training. For a pair of two conformers from a given ensemble, *pseudo-energies* E'_A and E'_B are calculated using a single model (e.g. DimeNet++ architecture). The model weights are updated using a pairwise loss function.

$$\mathcal{L}\left(E^{(i)}_{\mathrm{DFT}} - E^{(j)}_{\mathrm{DFT}}, E^{(i)}_{\mathrm{ML}} - E^{(j)}_{\mathrm{ML}}\right) = \left|\left(E^{(i)}_{\mathrm{DFT}} - E^{(j)}_{\mathrm{DFT}}\right) - \left(E^{(i)}_{\mathrm{ML}} - E^{(j)}_{\mathrm{ML}}\right)\right|$$

The same model instance is used for both conformers, with weight updates performed after completing both inference steps. Note that in principle any ML architecture which operates on a single molecular input can be employed.

The sets of ensembles used for training, validation and testing are the same as in the pointwise training process. As described earlier, we restrict each ensemble to a number of 20 conformers and we sample all pairs up to permutation, i.e. we consider all $20(20-1)/2 = 190$ pairs in each ensemble. This leads to 1 396 310 pairs in the training set, 121 410 pairs in the validation set and 189 620 pairs in the test set.

Observations For an exhaustive overview over the performance differences between pointwise and pairwise training see Table B.1 and Table B.2. Pairwise trained models consistently outperform their pointwise trained counterparts across virtually all metrics. For detailed description of training settings, see Supporting Information. The pairwise-trained DimeNet++ shows a substantial improvement of ranking probability (SF: 0.05, Top1 Accuracy: 0.81) compared to its pointwise trained version (SF: 0.12, Top1 accuracy: 0.61). When selecting all conformers in an energy window of 1 kcal mol^{-1} starting from the lowest prediction the pairwise trained DimeNet++ finds the actual lowest conformation 99 % of the cases, outperforming the pointwise trained model by 11 %. For reference, randomly guessing the lowest conformation would equate to a Top1 accuracy of $\frac{1}{20} = 5\%$, as the number of conformers in each ensemble is exactly 20. Similar improvements are observed for GemNet-T, SchNet and MACE models, highlighting the superior predictive accuracy and consistency of pairwise training. Closely followed by Dimenet++ and MACE, the pairwise-trained GemNet-T stands out as the best performer, achieving the lowest MAD of 0.46 kcal mol^{-1}, the highest R^2 of 0.98 and displaying

the strongest correlation metrics (Pearson's ρ_p: 0.99, Spearman's ρ_s: 0.97, Kendall's τ_K: 0.90), indicating robust and accurate energy predictions of the relative energy well below chemical accuracy of 1.00 kcal mol^{-1} (chemical accuracy). Leveraging pairwise training reduces the SF probability to 5 %, allowing to capture 99 % of the lowest lying conformers within chemical accuracy. These results suggest that the pairwise training approach on relative energies allows for very good results for conformational ranking.

For visual comparison, see the correlation plots in Figure B.4. GFN-FF shows poor correlation with r^{2}SCAN-3c , with many relative energies exceeding 5 kcal mol^{-1} and 10 kcal mol^{-1}. It furthermore fails to distinguish between different conformers, where DFT does not (non-zero conformational energies). GFN2-xTB improves with a visible correlation and no conformational energies beyond a 10 kcal mol^{-1} window. In contrast, DimeNet++ demonstrates excellent correlation, with all relative energies well within a 5 kcal mol^{-1} window.

B.5.3 Timings

A crucial aspect of improving conformer search is runtime. Enhancements that require excessive computational time compared to the GFN-FF calculations are impractical and render possible energetic improvements futile. Therefore, we investigate the runtime of ML models that were adapted to the ConfRank setup, comparing them with conventional methods. It should be noted that certain code modifications were required to adapt open-source implementations of the architectures studied to our framework. Also, model performance might be highly dependent on the chosen model hyperparameters. Therefore, all results are only representative of our specific implementation and chosen hyperparameters.

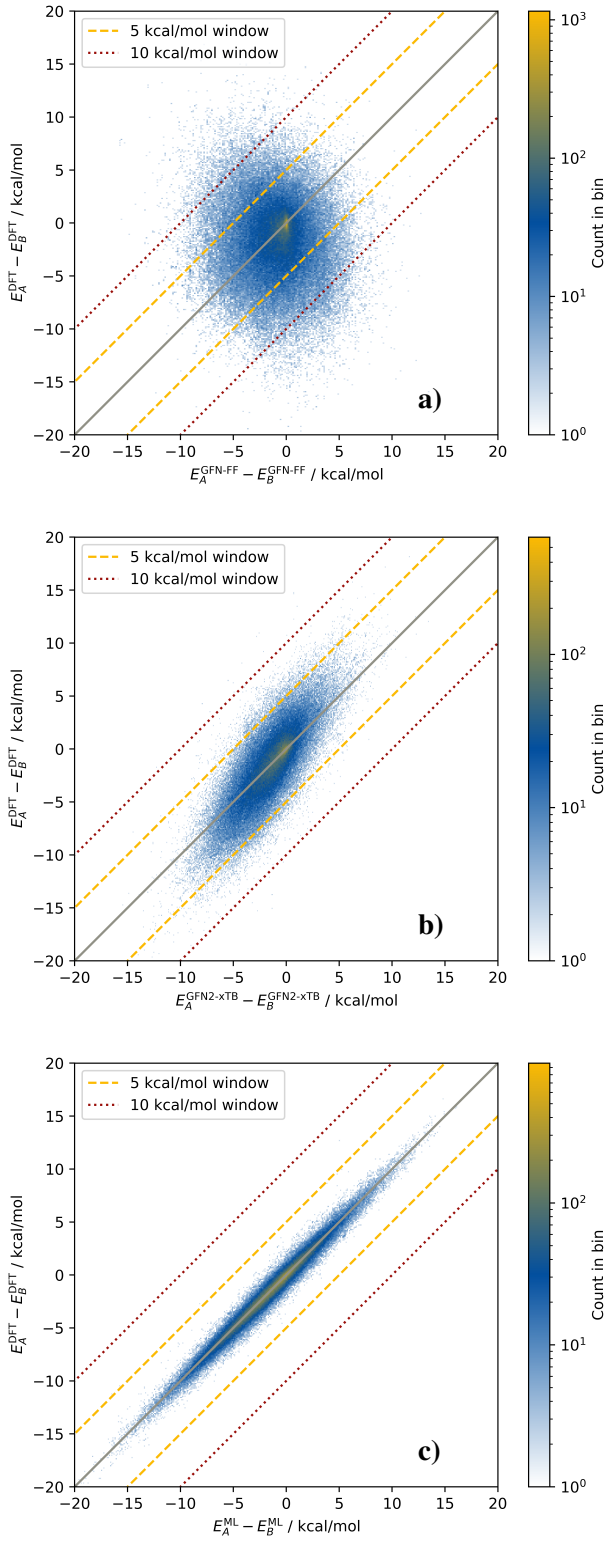

Figure B.4: Correlation of r^{2}SCAN-3c (DFT) and predicted relative energies on the test set across different methods: a) GFN-FF, b) GFN2-xTB and c) DimeNet++.

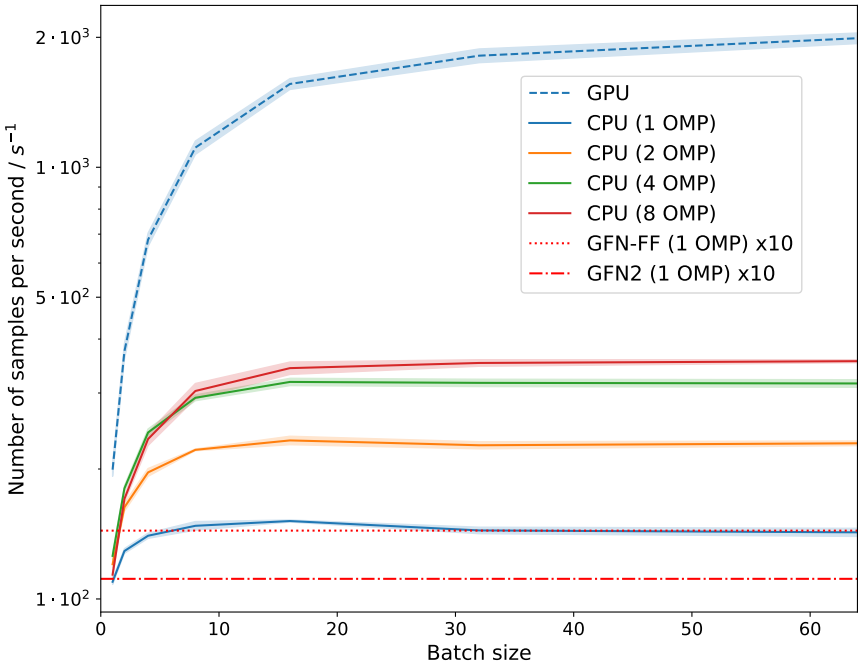

Figure B.5: Computational runtime of DimeNet++ as a function of the batch size for different numbers of Open Multi-Processing (OMP) threads and on the GPU when evaluated on its test set. Performance deterioration when using a single OMP thread for batch sizes larger than 16 is likely due to inefficiencies in workload distribution and overheads associated with processor-specific chunking of the task. Results are averaged over three runs and error bands correspond to the standard deviation of those runs.

Figure B.5 compares the runtime of DimeNet++ inference for different degrees of parallelization on CPU and GPU. Increasing the number of threads in OMP enhances the processing capacity, but GPU performance far exceeds CPU OMP, enabling an order of magnitude more samples to be processed per time. Batching also shows a greater impact on GPU performance yielding up to over 1 000 samples /s, allowing for high-throughput screening and energetic ranking of CREST runs. For comparison, GFN-FF (GFN2-xTB) singlepoint calculation achieve 143.9 (111.3) samples per second when running 10 single-thread processes in parallel. For GFN-FF and GFN2-xTB timing measurements, data is read from disk, while for ML models data is stored in PyTorch data structures already loaded in RAM. Furthermore, given that GFN-FF is expected to be over 100 times faster than GFN2-xTB, the measured runtimes are likely dominated by I/O operations and system calls. Nevertheless, the qualitative differences in runtime of traditional and ML methods become apparent. Hence, employing ConfRank on top of regular GFN-FF calculations results in practically no significant increase in runtime. Timings are measured on an Intel(R) Core(TM) i7-12700 CPU and a NVIDIA RTX 3070.

As shown in Figure B.6, different ML models exhibit varying inference timings, influenced by their individual architectures and chosen hyperparameters. We selected a comparable set of hyperparameters (e.g. cutoff radius and number of message passing layers) and number of trainable parameters for consistency. DimeNet++ demonstrates a good balance regarding accuracy and computational speed. Thus, we will focus on and report the DimeNet++ model performance in the following experiments. However, this choice is not a strict decision, as other models like MACE or GemNet-T can also be employed at an acceptable computational cost, given the significant runtime advantage over regular

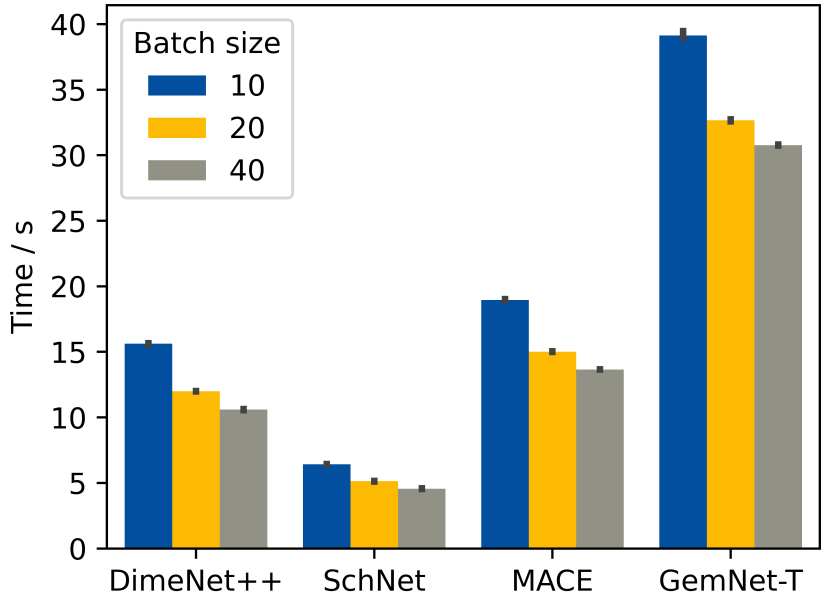

Figure B.6: Comparison of inference times for various ML models on GPU based on batch size. Single sample inferences were performed on the ConfRank test set, averaged over three runs. Black markers indicate the standard deviation.

GFN-FF calculations as mentioned before. Furthermore, detailed tweaking of each model and its hyperparameters beyond the chosen selection could yield further performance gains.

Besides the inference of pairwise energies, the runtime of the conformer sorting process is equivalent to that of standard sorting algorithms, as it relies on the intermediate energy-like outputs produced by the ML model (c.f. Figure B.3). In this sorting of pairs, pseudo-energies are directly compared and conformers sorted accordingly.

These findings highlight that the proposed ML operation for enhancing conformer ranking incurs minimal costs relative to the initial GFN-FF singlepoint calculations. Additionally, in a typical CREST run, the computational time is predominantly consumed by geometry optimizations, with singlepoint evaluations being significantly faster.

B.5.4 Out-of-Sample Performance Evaluation

QM9 + CREST

So far the ConfRank dataset is derived from conformers identified by CREST using GFN2-xTB, as documented in the GEOM dataset. In the preprocessing step we implicitly assumed that optimizing these structures at the GFN-FF level yields samples comparable to those from a CREST GFN-FF run (c.f. section B.4). However, it is important to note that a CREST run using GFN-FF may not necessarily identify the same conformer geometries as a run using GFN2-xTB due to differences in the underlying PES.

Since the target of this study is to enhance CREST GFN-FF runs, we test this hypothesis using conditions reflecting a real-world scenario. For this, we utilize the QM9 subset of the GEOM dataset

[169–171], conducting an unpruned CREST GFN-FF run on each ensemble.

In order to maintain reasonable computational effort for testing, ensembles were randomly sampled from the QM9 set. To simulate a realistic scenario where only individual structures are known but not their ensembles, a random conformer was selected from each ensemble. New ensembles are then created using CREST with GFN-FF and default settings, followed by a subsequent r^2SCAN-3c singlepoint computation for each conformer. In comparison to the accelerated training setup previously, exhaustive sampling is conducted and all found conformers are taken into account. In the following, the term "QM9-CREST" signifies CREST generated ensembles from the GEOM subset corresponding to QM9 molecules.

The so created dataset initially includes 1 162 712 conformers (110 926 217 pairs) distributed across 17 467 ensembles. The elemental composition comprises H, C, N, O, and F, mirroring that of the ConfRank training dataset, with no additional elements included. All samples are uncharged and contain up to nine heavy atoms. As a result, the size distribution of molecules covers a complementary range not represented in the ConfRank dataset used for training, leading to an out-of-sample scenario. In a second step, we remove 130 ensembles that include intermolecular non-covalent interactions according to the molecular topology generated by GFN-FF. Therefore, statistics are computed with respect to r^2SCAN-3c reference energies for 17337 ensembles, corresponding to 1 114 449 conformers (96 013 287 pairs).

As shown in Figure B.7, the pairwise-trained DimeNet++ outperforms both GFN methods in terms of ranking accuracy. Notably, DimeNet++ achieves a TOP3 accuracy of 85 % and an EW1 of 92 %. These results demonstrate the high efficiency of conformer selection when employing DimeNet++ for ranking tasks, enabling a tight selection of candidates to identify the lowest-lying DFT conformers and thus lowering computational costs in downstream calculations such as CENSO.

A full statistical overview over the performance on the QM9-CREST dataset can be found in the Supporting Information.

Confclean conformers To assess the robustness of the pairwise trained ML models on out-of-sample data, we investigate their performance across different datasets. Although a rigorous train-test split was performed on the ConfRank dataset (see section B.4), the training and testing domains remain similar due to their common origin. Therefore, in order to thoroughly evaluate the robustness across the organic chemical space, we compile the *Confclean* dataset that is mainly based on conformer subsets of the GMTKN55 dataset [89] which contains carefully curated samples over a wide range of chemical domains. These GMTKN55 subsets are organically comprised ensembles of varying molecular size and ensemble magnitude. Chemical motifs range from basic alkane chains to RNA-backbone conformers. The eight conformer sets of the original GMTKN55 database are further enhanced by the following subsets: 37conf8 [497], Glucose [498], Maltose [498], MPCONF196 [499]. Each subset containing different additions to the already present data. A tabular overview of the Confclean dataset can be found in Table B.3. For statistical evaluation, all combinatorial pairs within each subset's ensemble were considered. In contrast to previous experiments, ConfRank results are reported with respect to coupled-cluster reference energies, primarily based on CCSD(T)/CBS calculations. For a detailed explanation of the reference methods used, please refer to the corresponding subset publication cited in Table B.3. Note that parts of the GMTKN55 were used in the parametrization of the GFN-FF.

As shown in Figure B.8 and Figure B.9, DimeNet++ outperforms both GFN-FF and GFN2-xTB on most subsets. On average, the MAD (RMSD) of relative energies is improved by 30% (29%) across all subsets. With almost 62% the largest MAD improvement can be observed for the BUT14DIOL subset, achieving an MAD of 0.38 kcal mol^{-1} compared to an MAD of 0.98 kcal mol^{-1} for GFN-FF. Large

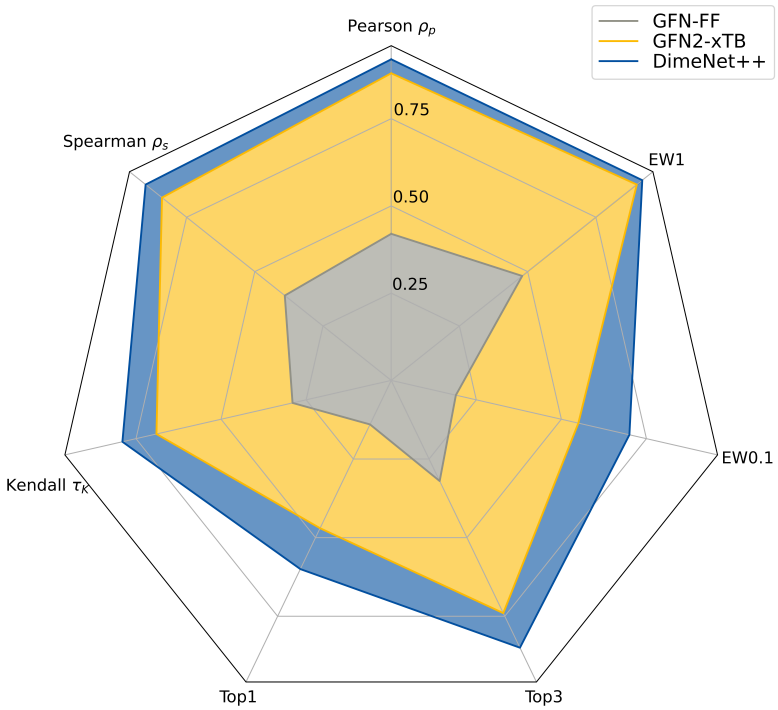

Figure B.7: Comparison of GFN methods and DimeNet++ on the QM9-CREST dataset. The units are agreeing with those reported in previous tables.

MAD improvements can also be observed for MPCONF196 (59 %), Maltose (53%) and Amino20x4 (49%), although for MPCONF196 and Maltose the MADs of 1.90 kcal mol^{-1} and 1.80 kcal mol^{-1}, respectively, are still rather large.

However, on certain subsets, DimeNet++ is not able to outperform GNF-FF. On the MCONF subset, DimeNet++ and GFN-FF perform equally well with an MAD of 0.81 kcal mol^{-1}, outperforming GFN2-xTB by 48 %. Additionally, we included the UPU23 subset, which contains charged RNA-backbone conformers. Since our model does not account for molecular charge, DimeNet++ processes these samples as neutral, despite their charged nature. Consequently, DimeNet++ shows the lowest performance on this subset, with a MAD of 3.85 kcal mol^{-1}, in comparison to GFN-FF's 2.77 kcal mol^{-1}. In contrast, both GFN-FF and GFN2-xTB can handle charged samples accurately. The investigation of models that can properly handle charged samples in combination with our ConfRank framework is left as a possible direction for future research.

When evaluated on the PCONF21 subset, DimeNet++ shows a deterioration of accuracy as well, resulting in a −23.50 % performance decrement compared to GFN-FF. The differences likely stem from the peptide nature of the PCONF21 conformers for which intramolecular hydrogen bonds play a pivotal role in this subset, which were underrepresented during training. As a result, we can consider PCONF21 as a meaningful test case for model development in the future. Interestingly, sugar subsets perform reasonably well and could be of research interest for cellular membrane molecules. A tabular

Dataset	Type	Description	N_{pairs}	Reference
37conf8	O	Industry-related organic drug-like conformers	1 036	Sharapa et al.[497]
ACONF	H	Short alkane chain conformers	73	Goerigk et al.[89]
ACONFL	H	Long alkane chain conformers	462	Ehlert et al.[500]
AMINO20x4	P	Amino acid conformers	200	Goerigk et al.[89]
BUT14DIOL	O	Butane-1,4-diol conformers	2 080	Goerigk et al.[89]
Glucose	S	Glucose conformers	13 266	Marianski et al.[498]
MCONF	O	Melatonin conformers	1 326	Goerigk et al.[89]
MPCONF196	P	(A)cyclic peptides and macrocyclic conformers	1 380	Rezáč et al.[499]
Maltose	S	α-Maltose conformers	24 753	Marianski et al.[498]
PCONF21	P	Tri- and tetrapeptide conformers	75	Goerigk et al.[89]
SCONF	S	Sugar conformers	111	Goerigk et al.[89]
UPU23	O	RNA-backbone conformers	276	Goerigk et al.[89]

Table B.3: Overview of selected GMTKN55 and additional datasets that contribute to the Confclean dataset [115]. N_{pairs} denotes the number of pairs per respective subset. In total 1 655 peptides (P), 38 130 sugars (S), 535 hydrocarbons (H) and 4 718 other molecules are included.

overview over the results and improvements are given in the Supporting Information.

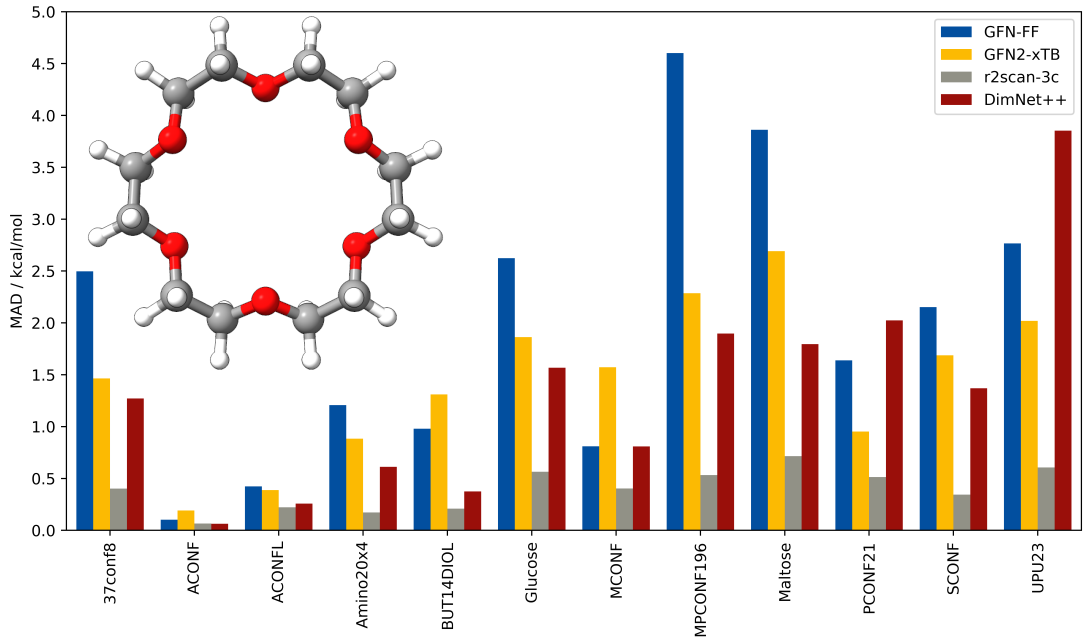

Figure B.8: Comparison of pairwise MAD for different methods on various subsets of the GMTKN55 containing conformers. The MAD is averaged across all ensembles within each subset.

Vancomycin To evaluate the applicability of the ConfRank approach to larger molecules, we included a case study with respect to vancomycin. Vancomycin is selected for two reasons: (a) vancomycin is a highly relevant molecule in the context of drug development, it being used as an

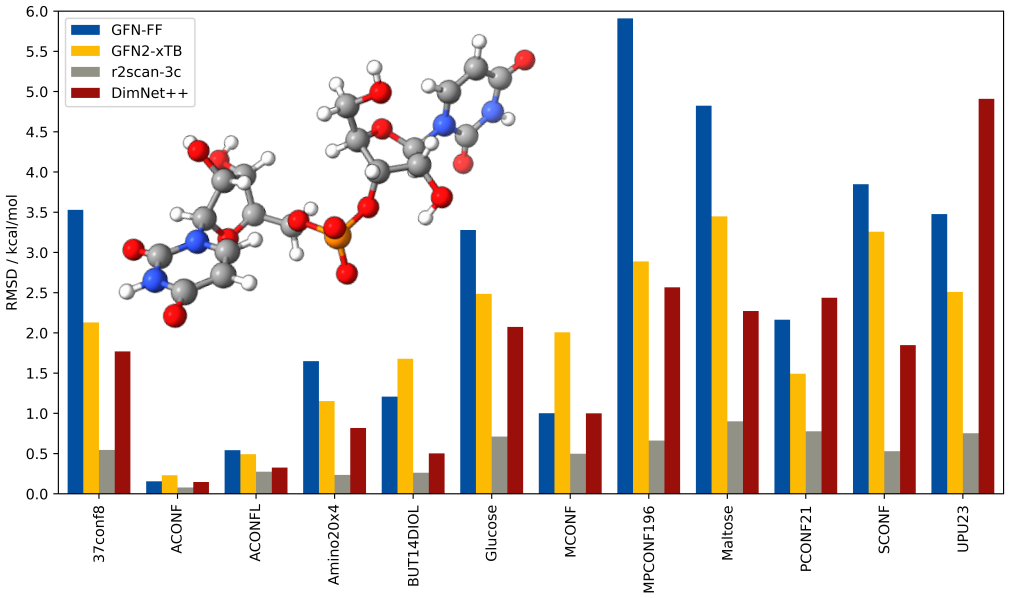

Figure B.9: Comparison of pairwise RMSD for different methods on various subsets of the GMTKN55 containing conformers. The RMSD is averaged across all ensembles within each subset.

effective antidote against multidrug-resistant Staphylococcus infections [501] and therefore being classified as critically important for human medicine by the WHO [502]; (b) with 176 atoms, vancomycin represents a sensible upper limit for performing CREST and DFT reference calculations with given computational resources. Containing almost twice as many atoms as the largest molecule in the training set, it also serves as a suitable test system for evaluating the extrapolation ability of our model. Therefore, we investigate vancomycin as a suitable proxy for evaluating the ConfRank approach with respect to bio-molecular systems. CREST on GFN-FF level was used to generate the conformer ensemble, and reference-level r^2SCAN-3c conformational energies were subsequently calculated.

As can be seen in Table B.4 the pairwise trained DimeNet++ outperforms the GFN2-xTB in both, error metrics (RMSD: 2.59 kcal mol^{-1} vs. 3.26 kcal mol^{-1}) and metrics that measure the ranking capability (spearman coefficient ρ_S: 0.95 vs. 0.92). Particularly, the capability to include only conformers within an energy window of 1.00 kcal mol^{-1} will enhance conformer generation methods like CREST. Note that both GFN2-xTB and the pairwise trained DimeNet++ model are able to find the energetically lowest conformer, as defined by the r^2SCAN-3c reference computation, within an energy window of 1 kcal mol^{-1}, whereas GFN-FF does not display sufficient accuracy in this example. As expected, we observe that the Ml model does not outperform most GFN2-xTB metrics as strongly as on the test set. This can be explained by the out-of-sample nature of vancomycin compared to the training set. Although not necessary in this example, this stresses the potential need to enrich the training set with suitable samples that are relevant for the respective application when moving too far

from the original training distribution. Retraining on custom datasets and therefore adaption of the ConfRank approach to specific applications is possible using our code. In summary, this exemplary study shows promising evidence that the ConfRank approach using e.g. the DimeNet++ architecture appears to be applicable for larger (bio-)molecular structures. However, the authors emphasize that the investigation of the ConfRank approach in a more realistic biological setting, including solvent effects and the presence of biological targets, was beyond the scope of this study and remains to be explored in future work.

	MAD$_\downarrow$	**RMSD**$_\downarrow$	R^2	**SF**$_\downarrow$	**EW1**	ρ_p	ρ_s	τ_K
GFN-FF	7.01	8.71	-0.11	0.53	0.00	-0.12	-0.10	-0.07
GFN2-xTB	2.54	3.26	0.84	0.12	1.00	0.92	0.92	0.75
DimeNet++ (pair)	2.00	2.59	0.90	0.09	1.00	0.95	0.95	0.81

Table B.4: Overview of selected performance metrics for the vancomycin ensemble. Since only a single ensemble is considered, most ensemble metrics are omitted. For detailed descriptions of the metrics, see Table B.1 and B.2.

B.6 Conclusion

In this study, we demonstrate that pairwise training of pre-developed ML models significantly enhances their ability to improve conformational ranking tasks. These models effectively identify conformational differences in organic, drug-like molecules across a diverse range of compounds. As detailed in section C.4, while the GFN-FF model exhibits poor correlation in conformer ranking, GFN2-xTB offers a more reliable albeit computationally more costly alternative. However, pairwise trained ML models outperform both methods regarding accuracy and computational timing, with the pairwise-trained GemNet-T model achieving the highest accuracy and Dimenet++ the best cost-benefit ratio.

Despite a relatively small training set of $\sim$ 160 000 molecules, the pairwise trained models demonstrate high data efficiency, achieving strong performance also due to the combinatorial increase of samples and robustness (c.f. transferability to QM9-CREST and Confclean datasets). Additionally, first tests on vancomycin indicate good performance on larger molecules relevant in the biological context. ConfRank is hence both data-efficient, requiring fewer samples, and cost-efficient, with fast inference times. Moreover, the approach yields the flexibility to integrate practically any MLIP with the pairwise learning for improved conformer ranking. Note that for this purpose, hyperparameter optimization could be conducted more extensively. In this study we focused on a fair comparison between the models rather than an undisputable final model.

Future work could explore ranking in solution and address the intrinsic incorporation of molecular charges to the ranking approach. Special attention should also be given to the accurate representation of hydrogen as well as halogen bonds. Additionally, the chemical domain space should be expanded to include non-organic compounds, such as transition metal complexes or lanthanoid structures. In passing we note, that besides improved energetic sorting, a more accurate geometry optimization is of high interest for future improvement on conformer search. Furthermore, one could investigate, the extend to which the presented results are independent of the specific ML model and reference DFT method. Overall, our results show an improvement in conformer ranking using pairwise training

for GFN-FF geometries and r^{2}SCAN-3c reference energies, paving the way for more efficient and accurate drug discovery in the future.

Data and Software Availability

The trained ML models and ConfRank dataset are available under `https://github.com/grimme-lab/confrank` and `https://zenodo.org/records/13354132`. Please refer to the information available there for instructions on setting up and loading the data and models. The dataset is stored in HDF5 format. To facilitate access, we provide a Python module for reading the data and models.

Acknowledgement

This work was supported by the DFG in the framework of the SPP 2363 on "Utilization and Development of Machine Learning for Molecular Applications - Molecular Machine Learning" and in parts by the DFG in the framework of the CRC 1639 "NuMeriQS" – project no. 511713970. We thank Dr. Astrid Maaß and Gregor Maier for their inspiring feedback. AI-based methods were used to assist in the development of the body text.

APPENDIX C

dxtb – An Efficient And Fully Differentiable Framework For Extended Tight-Binding

Marvin Friede,* Christian Hölzer,* Sebastian Ehlert,† and Stefan Grimme*

Received 30 April 2024, Published online 09 August 2024.

Reprinted in Appendix C (adapted), with the permission of AIP Publishing‡ from
M. Friede, C. Hölzer, S. Ehlert and S. Grimme, *dxtb – An efficient and fully differentiable framework for extended tight-binding*, J. Chem. Phys. **161** (2024) 062501, DOI: 10.1063/5.0216715.

* Mulliken Center for Theoretical Chemistry, University of Bonn, Beringstr. 4, 53115 Bonn, Germany
† AI4Science, Microsoft Research, Evert van de Beekstraat 354, 1118 CZ Schiphol, Netherlands

Abstract Automatic differentiation (AD) emerged as an integral part of machine learning, accelerating model development by enabling gradient-based optimization without explicit analytical derivatives. Recently, the benefits of AD and computing arbitrary-order derivatives with respect to any variable were also recognized in the field of quantum chemistry. In this work, we present *dxtb* – an open-source, fully differentiable framework for semiempirical extended tight-binding (xTB) methods. Developed entirely in Python and leveraging PyTorch for array operations, *dxtb* facilitates extensibility and rapid prototyping while maintaining computational efficiency. Through comprehensive code vectorization and optimization, we essentially reach the speed of compiled xTB programs for high-throughput calculations of small molecules. The excellent performance also scales to large systems, and batch operability yields additional benefits for execution on parallel hardware. Specifically, energy evaluations are on par with existing programs, whereas the speed of automatically differentiated nuclear derivatives is only 2 to 5 times slower compared to their analytical counterparts. We showcase the utility of AD in *dxtb* by calculating various molecular and spectroscopic properties, highlighting its capacity to enhance and simplify such evaluations. Furthermore, the framework streamlines optimization tasks and offers seamless integration of semiempirical quantum chemistry in machine learning, paving the way for physics-inspired end-to-end differentiable models. Ultimately, *dxtb* aims to further advance the capabilities of semiempirical methods, providing an extensible foundation for future developments and hybrid machine learning applications. The framework is accessible at `https://github.com/grimme-lab/dxtb`.

C.1 Introduction

With Kohn–Sham density functional theory (DFT),[11, 65] an accurate description and representation of electronic structure is available for routine calculations of energies and structures.[86, 503] However, even using systematic approximations to reduce the computational cost,[315, 504–510] DFT remains too costly for large scale calculations.[110] The recently introduced extended tight-binding (xTB) based semiempirical methods[17, 18, 148] alleviate this issue, allowing for a quantum mechanical description while retaining favorable computational cost. Correspondingly, they are routinely applied in high-throughput calculations (e.g. drug discovery[113], computation of mass spectra[511–513]), molecular and meta dynamics,[313] conformational searches,[114, 115, 514] and multilevel modeling.[116]

In these applications, semiempirical methods are particularly tasked with evaluating the more expensive nuclear derivatives: Gradients are essential for geometry optimizations and propagating forces in molecular dynamics. Hessian matrices, on the other hand, play a crucial role for characterizing stationary points on potential energy surfaces (transition state search[515, 516]) and to compute thermostatistical corrections.[517] Moreover, the importance of derivatives in quantum chemistry extends to other molecular (response) properties or general optimization problems.[518] As a result, significant efforts in method development are dedicated to the deduction and implementation of efficient analytical derivatives. Alternatively to these analytical derivatives, more costly and possibly unstable numerical differentiation algorithms are employed. The rise of machine learning (ML), however, drew attention to a third method: automatic differentiation (AD).[320, 519] Based on the chain rule, AD automatically accumulates the known elementary derivatives of any arbitrarily complicated function with machine precision, avoiding the pitfalls of both former methods. Therefore, AD holds the potential to significantly streamline and simplify method development by circumventing explicit derivative code.

Following its widespread adoption in ML (particularly through back-propagation[210]) and the accompanying plethora of frameworks, AD recently also found its way into computational chemistry (as well as other disciplines[520–523]) with diverse applications ranging from classical molecular dynamics, through established mean-field methods, to elaborate correlation methods. For example, the calculation of forces was automatized in molecular dynamics[524–526] as well as quantum Monte Carlo (QMC).[527] Other response properties were also computed with AD in auxiliary-field QMC.[528] Fully differentiable implementations of Hartree–Fock (HF[54, 529]),[326, 530–533] correlated wave function methods,[531, 532] and DFT[326, 532, 534, 535] were presented with applications to basis set optimization,[326, 530, 532] arbitrary-order derivatives,[531, 536] response properties,[537] and ML.[325, 538] Other works utilize AD in the context of coupled cluster,[539, 540] excited states,[541] or tensor-based methods.[542, 543] Despite these advancements, fully differentiable semiempirical methods, in particular the xTB methods, are still underexplored. Recent applications primarily focus on HF-based approximations,[544, 545] and the density functional tight-binding (DFTB)[137, 141–143, 546, 547] family of methods, facilitated by the TBMaLT toolkit.[321] In our recent contribution to this field, we implemented an interface to the GFN1-xTB[17] core Hamiltonian within TBMaLT.[321]

To provide a package for extended tight-binding methods, we present *dxtb* – an open-source, fully **d**ifferentiable e**x**tended **t**ight-**b**inding framework written in Python and employing PyTorch[548] for AD. Currently, *dxtb* implements only GFN1-xTB,[17] but the modular, toolbox-like structure facilitates straightforward incorporation of both existing tight-binding methods (GFN2-xTB[18]) and new variants. Besides the simple access to typical response properties already explored in HF and DFT, *dxtb*'s PyTorch-based implementation naturally furthers the recently highlighted[549] synergistic relation between semiempirical methods and ML.

In particular, the xTB Hamiltonian, or, more generally, the parametrization, represents an intriguing target for (ML-based) optimization. Indeed, the benefit of re-parametrizing semiempirical methods was already successfully explored for extended Hückel-based[117, 121] models,[550, 551] HF-based approximations,[552, 553] and DFTB.[554–557] These works employ ML-generated, dynamically augmented Hamiltonians, often leveraging fully differentiable implementations. While xTB is generally robust and versatile, its accuracy can be limited in specialized applications that benefit from tailored parameters. For instance, the IPEA-xTB variant emerged from re-fitting GFN1-xTB to DFT reference ionization potentials (IPs) and electron affinities (EAs) to enhance mass spectra calculations.[511] GFN1-xTB was also optimized for organosilicon compounds and halide perovskites.[558, 559] In fact, even adjusting a single parameter can significantly impact the treatment of uncommon chemical interactions.[560] Although some programs[328, 561] already provide rudimentary re-parametrization capabilities for xTB, *dxtb* considerably simplifies and advances this procedure.

For applications with a stronger focus on machine learning, semiempirical methods offer the necessary efficiency to be used in tandem with ML, while simultaneously embedding domain knowledge for enhanced, physics-inspired models. This favorable interplay is exemplified in the ΔML approach, where the model is trained to predict the difference between a low-level base method and a high-level target method.[562] Particularly, models rooted in semiempirical methods have demonstrated remarkable capabilities, achieving corrections towards DFT or Coupled Cluster accuracy, not only for the training data but also extending beyond.[7, 452, 563–565] With *dxtb*, full back-propagation becomes possible, paving the way for even tighter integration of quantum chemistry into ML.[566] Moreover, *dxtb* enables the extraction of quantum mechanical features such as the charge distribution via atomic partial charges, local anisotropy via multipoles, and atom-resolved energy contributions, enriching the feature set for machine learning models and facilitating a more nuanced representation

of molecular systems.

In essence, *dxtb* not only provides a crucial stepping stone to further unlock the potential of semiempirical methods in ML, but also permits the refinement of tight-binding methods for specific problems.

The paper is organized as follows. In section C.2, the theory of GFN1-xTB is explained and a detailed overview of AD is given. The implementation of *dxtb* is described in section C.3 covering design choices, performance, and framework-specific challenges. Applications are presented in section C.4. We summarize our work and consider future directions in section C.5. Finally, we want to emphasize that *dxtb* is an open-source project, which can be found at `https://github.com/grimme-lab/dxtb`, along with examples and extensive documentation.

C.2 Theory and Methods

C.2.1 Extended Tight-binding (xTB)

We derive the tight-binding energy expression by approximating a non-local KS-DFT energy functional $E[\rho]$ expanded around a known reference density ρ_0 up to a given order

$$\begin{aligned} E[\rho] = \quad & E^{(0)}[\rho_0] + E^{(1)}[\rho_0, \delta\rho] + E^{(2)}[\rho_0, (\delta\rho)^2] \\ & + E^{(3)}[\rho_0, (\delta\rho)^3] + O((\delta\rho)^4)\,, \end{aligned} \tag{C.1}$$

where the reference density ρ_0 is the superposition of spherical, neutral atomic densities, which allows expressing the energy contributions in terms of charge fluctuations $\delta\rho$ restricted to the valence electrons.

Correspondingly, the wave function for the xTB Hamiltonian is formulated in terms of a partially polarized, valence-only minimal basis set of spherical Gaussian-type atomic orbitals (GTO). The GTO expansion is obtained by approximating a Slater-type orbital (STO) using the STO-*n*G expansion.[43] We parametrize the molecular orbitals ψ_j as linear combination of atomic orbitals (LCAO)

$$\psi_j(\vec{r}) = \sum_{\mu}^{N_{\mathrm{AO}}} C_{\mu j} \phi_\mu(\vec{r}) \tag{C.2}$$

where ϕ_μ are the GTOs and $C_{\mu j}$ are the orbital coefficients obtained self-consistently solving the Roothaan–Hall type[567, 568] equation

$$\mathbf{FC} = \mathbf{SC}\boldsymbol{\varepsilon}\,, \tag{C.3}$$

where $\mathbf{S}$ is the overlap between the GTOs, $\boldsymbol{\varepsilon}$ are the orbital energies, and $\mathbf{F}$ is the tight-binding Fock matrix derived from variationally minimizing the tight-binding energy expression detailed in the subsequent paragraph. Note that the following expressions focus on GFN1-xTB[17] and are adjusted to yield atom-resolved[§] energies to reflect the implementation.

[§] In contrast to Ref. [17], the summations in the equations only run over a single atom, instead of both atoms or all atom pairs. Hence, the obtained energies do not reflect the total energy of the tight-binding component, but only the energy for a single atom.

For the zeroth order energy contribution, we find the interaction between the spherical atomic reference densities. Since we use a non-local functional as a starting point, we include the repulsive interaction between the screened nuclear charges as well as the London dispersion from instantaneous dipole polarization

$$E^{(0)}[\rho_0] = \sum_{\mathrm{A}}^{N_{\mathrm{atom}}} \left(E_{\mathrm{atom}}[\rho_0^{\mathrm{A}}] + E_{\mathrm{A}}^{\mathrm{rep}} + E_{\mathrm{A}}^{\mathrm{disp}} \right) , \tag{C.4}$$

where the atomic energies $E_{\mathrm{atom}}[\rho_0^{\mathrm{A}}]$ are usually removed since they are constant.The repulsion $E_{\mathrm{A}}^{\mathrm{rep}}$ energy is calculated by a pairwise energy expression using the screened Coulomb law between effective nuclear charges Z^{eff}

$$E_{\mathrm{A}}^{\mathrm{rep}} = \frac{1}{2} \sum_{\mathrm{B}\neq\mathrm{A}}^{N_{\mathrm{atom}}} \frac{Z_{\mathrm{A}}^{\mathrm{eff}} Z_{\mathrm{B}}^{\mathrm{eff}}}{R_{\mathrm{AB}}} \exp\left[-\sqrt{a_{\mathrm{A}} a_{\mathrm{B}}} R_{\mathrm{AB}}^{k} \right] , \tag{C.5}$$

where $a_{\mathrm{A/B}}$ describe the screening of the density and factor k an empirical scaling factor.[569] For the dispersion energy $E_{\mathrm{A}}^{\mathrm{disp}}$, we use the established DFT-D3(BJ)[146, 570] correction

$$E_{\mathrm{A}}^{\mathrm{disp}} = -\frac{1}{2} \sum_{\mathrm{B}\neq\mathrm{A}}^{N_{\mathrm{atom}}} \sum_{n}^{6,8} s_n \frac{C_n^{\mathrm{AB}}}{R_{\mathrm{AB}}^{n} + (a_1 R_{\mathrm{AB}}^{0} + a_2)^{n}} , \tag{C.6}$$

where a_1 and a_2 are the global parameters of the rational damping function,[571–573] R_{AB}^{0} are the square root of quotients of the C_6 and C_8 dispersion coefficients. The scaling factor s_6 is fixed to unity to obtain the correct asymptotic behavior, while the s_8 scaling factor is a free parameter to compensate for missing higher order contributions to the dispersion energy.

The first-order contributions from the density arise from the deformed atomic densities interacting with the zero field of the remaining atoms. This can be concisely expressed by the atomic energy arising from the core Hamiltonian $\mathbf{H}^0$

$$E_{\mathrm{A}}^{\mathrm{EHT}} = \sum_{\mu\in\mathrm{A}}^{N_{\mathrm{AO}}} \sum_{\nu}^{N_{\mathrm{AO}}} H_{\mu\nu}^{0} P_{\nu\mu} , \tag{C.7}$$

where $\mathbf{P}$ is the density matrix evaluated from the orbital coefficients of occupied molecular orbitals. The core Hamiltonian itself employs the usual extended Hückel theory (EHT[117, 121]) type approximation of calculating the diatomic elements from the scaled average of the onsite elements proportional to the overlap integral $\mathbf{S}$ (Wolfsberg–Helmholtz approximation[574, 575])

$$H_{\mu\nu}^{0} = K \cdot \frac{H_{\mu\mu} + H_{\nu\nu}}{2} \cdot S_{\mu\nu} , \tag{C.8}$$

where the $H_{\mu\mu/\nu\nu}$ are the atomic level energies. Instead of the simple Wolfsberg–Helmholtz constant, K represents an elaborate scaling function that depends on the angular momenta of the shells, interatomic distance and electronegativity to provide the necessary flexibility to capture different effects not encoded in the overlap alone. In fact, the majority of the xTB parameters (> 50%) are used to parametrize this transformation of the overlap integral to the core Hamiltonian.

Finally, the interatomic electrostatic interactions are accounted for by the second-order and

third-order Coulomb electrostatic contributions defined as

$$E_{\mathrm{A}}^{\mathrm{coul}} = E_{\mathrm{A}}^{\mathrm{coul},2} + E_{\mathrm{A}}^{\mathrm{coul},3} \tag{C.9}$$

$$= \frac{1}{2}\sum_{\mu\in\mathrm{A}}^{N_{\mathrm{AO}}}\sum_{\mu}^{N_{\mathrm{AO}}} q_\mu q_\nu J_{\mu\nu} + \frac{1}{3}\cdot\Gamma_{\mathrm{A}}\left(\sum_{\mu\in\mathrm{A}}^{N_{\mathrm{AO}}} q_\mu\right)^3 , \tag{C.10}$$

with Γ_{A} being the derivative of the Hubbard parameter $U_{\mathrm{A},\ell}$ and $J_{\mu\nu}$ as the Coulomb matrix based on the Mataga–Nishimoto–Klopman–Ohno interaction kernel[576–578] given as

$$J_{\mu\in\mathrm{A},\nu\in\mathrm{B}} = \left(R_{\mathrm{AB}}^g + f_{\mathrm{av}}(U_{\mathrm{A},\ell}, U_{\mathrm{B},\ell'})^{-g}\right)^{-1/g} , \tag{C.11}$$

where f_{av} is an average like the harmonic mean. We define the orbital partial charges using the Mulliken partition scheme via

$$q_\mu = n_\mu^0 - \sum_{\nu}^{N_{\mathrm{AO}}} S_{\mu\nu}P_{\nu\mu} , \tag{C.12}$$

where n_μ^0 is the reference population of the respective orbital in the free atom.

The monopole approximation of the electrostatic energy, however, leads to a poor description of halogen bonds.[139] To correct this deficiency, we additionally include a classical, atom-pairwise, Lennard–Jones-like halogen-bond correction

$$E_{\mathrm{X}}^{\mathrm{hal}} = \sum_{\mathrm{B}}^{N_{\mathrm{B}}} f_{\mathrm{AXB}}^{\mathrm{damp}} k_{\mathrm{X}} \left[\left(\frac{k_{\mathrm{XR}}R_{\mathrm{XB}}^{\mathrm{cov}}}{R_{\mathrm{XB}}}\right)^{12} - k_{\mathrm{X2}}\left(\frac{k_{\mathrm{XR}}R_{\mathrm{XB}}^{\mathrm{cov}}}{R_{\mathrm{XB}}}\right)^{6}\right] \left[\left(\frac{k_{\mathrm{XR}}R_{\mathrm{XB}}^{\mathrm{cov}}}{R_{\mathrm{XB}}}\right)^{12} + 1\right]^{-1} , \tag{C.13}$$

where X defines the halogen (Br, I, At), B the non-covalently bound Lewis base (N, O), and A the nearest neighbor of X. The halogen-specific parameter k_X adapts the overall strength of the bond, while k_{XR} and k_{X2} are global scaling parameters. $R_{\mathrm{XB}}^{\mathrm{cov}}$ denotes the covalent distance of the halogen bond, and $f_{\mathrm{AXB}}^{\mathrm{damp}}$ an angle-dependent damping function.

Including the electronic free energy resulting from the fractional occupation of orbitals $T^{\mathrm{el}}S^{\mathrm{el}}$,[579] yields the final energy expression for GFN1-xTB[17]

$$E_{\mathrm{total}}^{\text{GFN1-xTB}} = \sum_{\mathrm{A}}^{N_{\mathrm{atom}}} \left(E_{\mathrm{A}}^{\mathrm{rep}} + E_{\mathrm{A}}^{\mathrm{disp}} + E_{\mathrm{A}}^{\mathrm{EHT}} + E_{\mathrm{A}}^{\mathrm{coul}} + E_{\mathrm{A}}^{\mathrm{hal}}\right) + T^{\mathrm{el}}S^{\mathrm{el}} . \tag{C.14}$$

Variational minimization leads to the tight-binding Fockian, which can be decomposed into the core

Hamiltonian ($\mathbf{H}^0$) and charge-dependent contributions ($\mathbf{H}^1$).

$$F_{\mu\nu} = \quad H^0_{\mu\nu} + H^1_{\mu\nu} = \frac{1}{2} S_{\mu\nu} \Bigg(\underbrace{K \left(H_{\mu\mu} + H_{\nu\nu} \right)}_{\text{EHT}} - \underbrace{\sum\nolimits_{\kappa} \left(J_{\mu\kappa} + J_{\nu\kappa} \right) q_{\kappa}}_{2^{\text{nd}}\text{ order}} - \underbrace{\left(\Gamma_{\mu} q_{\mu}^2 + \Gamma_{\nu} q_{\nu}^2 \right)}_{3^{\text{rd}}\text{ order}} \Bigg) \tag{C.15}$$

Owing to the charge-dependency of $\mathbf{H}^1$, a self-consistent procedure is required for solving the Roothaan–Hall type equation (Equation C.3).

C.2.2 Automatic Differentiation

Although automatic differentiation (AD) has been researched since the 60s,[319, 320] it only emerged as a mainstream technique due to its wide-spread adoption by the machine learning community over the last decade.[580, 581] While its success sparked various recent applications in computational chemistry,[321, 325, 326, 524, 525, 527, 528, 530–532, 534–536, 538, 545, 582] AD still remains a rather underexplored differentiation paradigm. For this reason, we give a detailed account of its inner workings and compare to the established numerical and analytical differentiation techniques.

Analytical differentiation requires the derivation of explicit expressions, either by hand or with the help of computer tools (symbolic differentiation), and their efficient implementation. Therefore, the feasibility of analytical derivatives is governed by the complexity of the underlying formulas and often incurs tremendous human effort. In fact, (even nuclear) gradients of many intricate quantum chemistry methods were published significantly later than the method itself[583–585] or are still lacking.[586, 587] The main advantage of analytical differentiation lies in the computational speed of closed-form expressions, which rationalizes the prevalence of this technique in computational chemistry. In machine learning, however, the variety and rapid development of model architectures renders analytical derivatives impractical.

Numerical differentiation, on the other hand, approximates derivatives using discretized points, usually by employing finite difference methods. Consequently, this technique becomes indifferent toward the underlying problem or theory, which allows black-box-like application. On the downside, numerical differentiation is inherently ill-conditioned, prone to step size-related errors, and quickly becomes computationally expensive due to its scaling with the gradient size.[580] Owing to this inefficiency, numerical derivatives often only serve as temporary solutions in computational chemistry when analytical derivatives are unavailable. Moreover, the scaling even represents an insurmountable obstacle for state-of-the-art machine learning models, where the number of parameters, and consequently, the gradient entries, can exceed billions.[588, 589]

These major shortcomings of analytical and numerical differentiation are alleviated by AD – a set of techniques that augment a computer program such that derivatives are computed without explicit derivative statements or programmer intervention. The key concept behind AD exploits the chain rule of differential calculus: Any arbitrarily complicated function can be decomposed into a sequence of primitives (elementary arithmetic operations, mathematical functions), and combining their known analytical derivatives following the chain rule gives the derivative of the whole function.

This not only implies derivative evaluation with machine precision but, in principle, also effortless differentiation through control flow statements (loops, conditionals, recursion) and complete algorithms. Correspondingly, general derivative evaluation of arbitrary order is made possible out-of-the-box, which can considerably simplify method development by relinquishing tedious implementation work.

Despite existing theoretical complexity bounds for automatic derivative evaluation (Baur–Strassen theorem[590]), judging the general efficiency of AD is not straightforward. The observed cost usually lies between analytical and numerical differentiation, but is ultimately governed by the underlying problem, the flavor and implementation of AD. Its recent success in machine learning sparked the development of a plethora of Python-based (general-purpose) AD frameworks.[548, 591–598] The Python frameworks utilize the core concepts of the initial, non-ML-oriented AD tools developed in Fortran or C++, i.e, employing either source code transformation techniques[599–602] or the operator overloading approach[602–604] for derivative computation. Both variants have differing, almost opposing benefits regarding flexibility, accessibility and efficiency,[605] that are naturally inherited by the Python adaptations. The distinction is often expressed in the nature of the computational graph (static or dynamic) – a directed acyclic graph representation of the complete program with operations and data as nodes and data flow encoded by edges.

In TensorFlow[595] and Theano[596], for example, programs are constructed in a framework-specific sub-language to facilitate (source code) transformation into a static computational graph (intermediate representation). Before execution, this low-level graph representation is compiled/interpreted, permitting structural and algebraic optimizations.[581] The increased efficiency, however, is bought at the expense of programmatic flexibility and development convenience.

Frameworks like PyTorch[548] and JAX[593, 594], on the other side, provide more pythonic, general-purpose AD capabilities. Here, operator overloading facilitates on-the-fly construction of a dynamic computational graph. While this incurs a performance decrease due to additional overhead at runtime and missing compiler optimization, PyTorch-like frameworks grant full flexibility with arbitrary control flow statements, abolish the need for another compiler or interpreter, and uphold Python's convenient and simple syntax. Nowadays, most applications in computational chemistry utilize PyTorch-like frameworks, not least because of the ubiquitous nature of control flow statements like loops and conditionals.[321, 325, 326, 524, 525, 528, 531, 532, 534, 538, 545, 582] In fact, even TensorFlow introduced a paradigm shift from static graph building to eager execution in their second major release.

Besides the specific implementation, the efficiency of AD also depends on the execution order of the derivative evaluation. Permitted by the associativity of the chain rule, AD is divided into forward and reverse (backward) mode. In forward mode, derivatives are accumulated along with each primitive evaluation (constant memory complexity), which translates to solving the chain rule from right to left or from inputs to outputs. Reverse mode starts from the outputs and traverses the chain rule in opposite direction (left to right), which necessitates storage of intermediates. Correspondingly, the memory complexity scales with the number of intermediates. More importantly, however, the runtime complexities of forward and reverse mode scale with the number of inputs and outputs, respectively. For this reason, machine learning applications almost exclusively employ reverse mode AD (backpropagation[210]), as the inputs are usually large features vectors while the output is a scalar-valued loss. Computational chemistry often follows the same approach due to the central importance of energy derivatives (e.g. nuclear gradients), although arguments in favor of forward mode differentiation have also been brought forward.[528, 530] Problems regarding memory consumption can be alleviated by checkpointing.[320, 528]

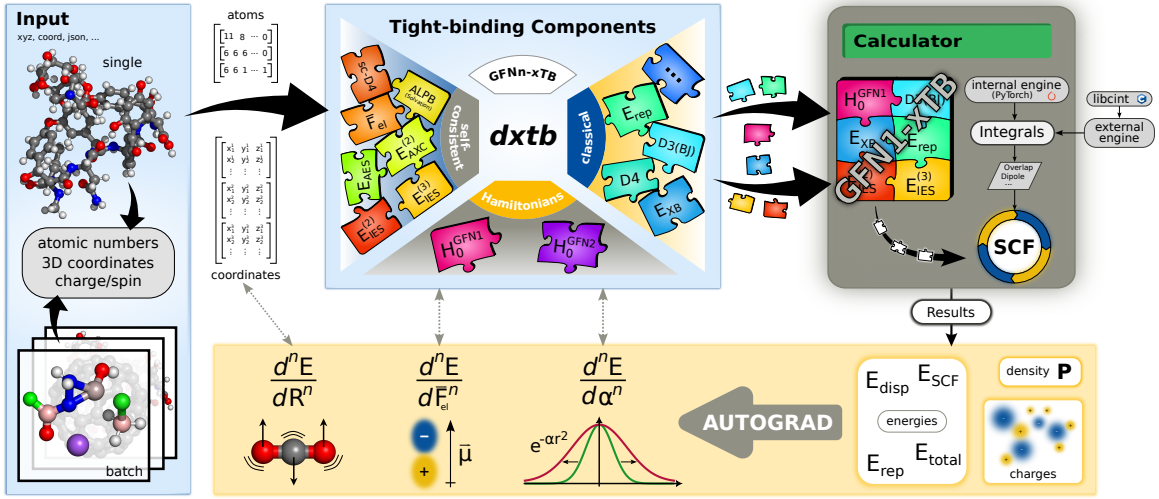

Figure C.1: Schematic overview of the *dxtb* framework. Starting from the left, the user simply provides the atom types and Cartesian coordinates of a single or multiple structures as input (blue). Secondly, a tight-binding method is chosen for the calculation. While *dxtb* implements convenient shortcuts for known methods (GFN1-xTB), other Hamiltonians and energy contributions can also be selected, as depicted by the puzzle pieces. Finally, the calculator (gray) is tasked with computing properties like energy, vibrations, or dipole moments (yellow). Note that the calculator does not implement any derivatives but utilizes PyTorch's autograd engine to obtain arbitrary order derivatives of any quantity.

With this in mind, the following section details the implementation and design of our fully differentiable extended tight-binding package *dxtb*, which uses PyTorch as a framework for reverse mode AD and machine learning integration. Furthermore, we discuss important considerations and caveats of developing tight-binding models in a fully differentiable fashion.

C.3 Implementation

C.3.1 Structure and Design

The main goal of *dxtb* lies in providing an easily accessible and flexible framework for extended tight-binding methods that lays the foundation for machine learning applications and method optimization. Correspondingly, *dxtb* is written in Python, which currently takes the place of the most popular and fastest-growing programming language. Additionally, Python established itself as the preferred choice for ML applications, which is bolstered by the multitude of powerful frameworks and tools.[548, 591–598, 606] From this vast pool, we chose PyTorch[548] due to its easy-to-use, pythonic nature and flexible AD engine, as outlined in the previous section. All tight-binding expressions are fully implemented in PyTorch syntax to support unrestricted AD. The overall *dxtb* framework is constructed in a modular fashion that abstracts implementation details and provides a robust, consistent and intuitive user interface in order to facilitate extensibility and library integration. The software design concept is visualized in Figure C.1 and detailed in the following.

The modular structure is inspired by the different contributions to the final tight-binding energy

expression (Equation C.14). We split the energy contributions into two overarching base classes: While the non-self-consistent CLASSICALS (e.g. repulsion or dispersion energy) are simply additive, the self-consistent INTERACTIONS (e.g. electrostatic terms) are charge-dependent and contribute to the Hamiltonian. The general workflow of all classes encompasses three steps: (1) Initialization with the parameters of the chosen tight-binding method, (2) building its cache object that avoids superfluous calculations, and (3) running the energy or potential evaluations.

Since all components of the tight-binding model inherit from one of the two base classes, they only differ in the specific implementations of their energy or potential evaluation, which allows further abstraction by collecting all contributions in two lists of component objects. Hence, any new tight-binding term, or even other (external) interactions, that follow the template classes can simply be appended to the corresponding lists. We provide a detailed example for the implementation of the electric field as an additional interaction in the documentation.[607] Finally, we note that all classes implement functionality for changing the data type (floating-point accuracy) and computation device, similar to PyTorch's tensor and model types. Per default, all calculations utilize double-precision floating-point accuracy, following the standard approach in quantum chemistry and ensuring consistency with the original Fortran program.

All components of the tight-binding model are consolidated in an *ASE*-inspired[608], general CALCULATOR object that acts as the primary entry point for calculations. Upon instantiation, the CALCULATOR is supplied with a parametrization, from which it creates all required contributions that define the given tight-binding method. The whole parametrization is conveniently stored in a single TOML file, although JSON and YAML formats are also supported. The data is then collected and validated through a recursive Pydantic[609] model. The CALCULATOR also accepts additional energy contributions that are not defined through the parametrization, like solvation models or external fields. After initialization, the desired quantities can be requested from the CALCULATOR by calling the corresponding methods supplied with the target systems. All necessary calculation steps from classical repulsion energy evaluation to solving the self-consistent charge equations happen in the background and do not require user intervention. Permitted by the full PyTorch implementation of the entire workflow, derivatives are available for all results and parameters simply by invoking the AD engine.

C.3.2 Performance

To uphold the low computational cost associated with semiempirical methods, an implementation that approaches the efficiency of the original *xtb* Fortran implementation[486] is desirable. Unfortunately, a Python implementation usually sacrifices performance for convenience and flexibility. To reduce computation time, we extensively utilize PyTorch's efficient handling of operations on multidimensional arrays and formulate the vast majority of the code in a vectorized fashion. This avoids slow Python loop structures by outsourcing the bulk of the computations to PyTorch's backend, and simplifies execution on massively parallel hardware thanks to PyTorch's native graphical processing unit (GPU) support. Additionally, AD itself benefits from vectorization,[610, 611] and the implementation in terms of multidimensional arrays lends itself well for an extension to batch operability, which further exploits vectorization. In batched calculations, n k-dimensional arrays are concatenated to form a single $(k + 1)$-dimensional array, eliminating one additional Python loop. Note that the arrays in the batch dimension must have the same size, i.e. prior to concatenation, arrays sizes must be normalized using padding. Although additional efficiency for batched calculations may be achieved with sparse

array methods, their support in PyTorch is currently limited. *dxtb* is batch-agnostic, i.e. fully supports single-system as well as batched calculations.

Unfortunately, certain mathematical structures hinder or even prevent full vectorization of calculations. Fortunately, only two building blocks of the tight-binding methods are affected. Firstly, the halogen bond correction in GFN1-xTB requires a nearest neighbor search, which is intrinsically challenging to vectorize. In fact, within the ML community, substantial efforts are devoted to optimizing general geometric computations, often employing specialized kernels or libraries.[612–616] Secondly, the integral computation also poses problems, because the shape of the intermediate integral arrays varies depending on the angular momenta. While the lack of vectorization in the context of the halogen bond correction is mitigated by the typically small number of halogen bond donors and acceptors relative to the overall molecular size, the computation of integrals (and their derivatives) becomes a bottleneck in the absence of additional enhancements. We tested several algorithms and strategies for integral calculation, which will be detailed and assessed subsequently. Note that tight-binding methods neglect more complicated integrals, such as electron repulsion integrals, and only require two-center one-electron multipole integrals, in particular the monopole (overlap), dipole, and quadrupole integrals. The timings of the tested approaches are shown for the overlap integral in Figure C.2.

Integrals

The simplest and most naive starting point for the integral implementation constitutes a one-to-one translation of the loop-based Fortran code to Python. This approach is expectedly plagued by extreme inefficiency (Figure C.2, "loop-based") and was quickly discarded, but also revealed further complications arising from the integral algorithm itself. While the horizontal Obara–Saika scheme[617, 618] is used in the Fortran implementation due to its favorable performance for low angular momenta,[619] the mathematical structure prohibits simple vectorization. Therefore, *dxtb* implements a pure PyTorch version of the flexible McMurchie–Davidson algorithm[620] that allows for better vectorization.[619] Note that we explicitly write out the expansion coefficients for the Hermite Gaussians instead of using the usual recursion relations in order to avoid in-place operations breaking the computational graph. Otherwise, recursive schemes must be avoided,[531] or custom derivatives must be written. While the vectorized McMurchie–Davidson ansatz marks a notable improvement over the naive loop-based version, it remains substantially outperformed by the Fortran code, displaying a performance deficit of two orders of magnitude across all tested systems (Figure C.2, "vectorized").

Additional computational efficiency is achieved by selecting unique orbital pairs (e.g. carbon 2s-orbital and carbon 2p-orbital) and collecting their corresponding coordinates within a system or batch thereof. The unique pair ansatz effectively enables vectorization over the centers of GTOs, but can also prevent repeated calculations in loop-based codes.[621] Taking C_{60} (carbon 2s- and 2p-orbital) as an example, this vectorization over the GTO centers of unique pairs reduces the number of calls into the integral algorithm from 180 to 3. A graphical depiction of the algorithm is shown in Figure S5. The unique atom/shell/orbital pair ansatz is also exploited for all parameter-related computations, albeit the performance gain is only marginal. With this vectorization step, the overlap calculation can be sped up by another order of magnitude, but only for the medium and large test molecules (yellow, blue). For smaller molecules (gray), the limited number of unique pairs, coupled with the overhead introduced by identifying these pairs, diminishes the performance gains. Note that despite the McMurchie–Davidson algorithm containing no Python loops, it always returns matrices of

shape $l_\mathrm{a} \times l_\mathrm{b}$, with $l_\mathrm{a/b}$ being the angular momentum of the bra and ket side of the integral, respectively. Due to these varying shapes, the write-back to the full square integral matrix of shape $n_\mathrm{AO} \times n_\mathrm{AO}$ (n_AO: total number of atomic orbitals) can never circumvent a Python loop, limiting even the fastest implementation. The problem of non-vectorized assignment was already pointed out in related work.[530]

Although the computational speed appears acceptable for a pure Python implementation, several caveats remain. Firstly, data sets for machine learning in chemistry primarily contain (millions of) small molecules,[169, 170, 186, 622] which makes the slow integral evaluation for this size particular punishing. Secondly, the performance penalty incurred by Python loops in the integral evaluation directly causes inefficient AD, because the computational graph will explicitly contain every iteration's operations, instead of squashing higher-level operations into a single node. The negative impact on both runtime and memory consumption can be clearly observed for the "loop-based" version, which requires 6.7 GB of peak memory for the second-order geometric overlap derivative of vancomycin. While this clearly constitutes the worst case, even the most efficient approach ("unique pair") consumes 1.0 GB of memory for this derivative.

We attempted to address these runtime and memory issues by embedding analytical first-order nuclear overlap derivatives in the AD engine, using the "unique pair" ansatz. While expectedly outperforming the automatically differentiated first-order derivatives, higher-order derivatives become considerably slower, because the analytical version employs Python loops again (Figure S1). Additionally, these Python loops increase memory consumption for higher-order derivatives, with the second-order overlap derivative of vancomycin now requiring up to 1.2 GB. Considering runtime efficiency again, even fully vectorized analytical implementations do not necessarily outperform AD (Figure S2), underscoring the limited applicability of selected analytical derivatives. Lastly, the extension of a purely PyTorch-based integral code remains tedious, especially considering the necessary hand-crafted optimizations.

To overcome these limitations, we delegate all integral-related computations to the high-performance C integral library *libcint*.[324] Specifically, we adopt the PyTorch wrappers of the DQC program package[325, 326] that bridge the gap between Python and C similar to PySCF.[623] The integration of external libraries is facilitated by the customization of primitives in AD frameworks: Any sequence of operations can readily be grouped to a custom primitive, given a corresponding user-defined derivative. PyTorch itself exploits this feature in linear algebra operations, like matrix multiplication or decomposition, to improve efficiency and numerical stability of derivatives. Note that vectorized code extensively utilizes linear algebra operations, which makes not only the calculation itself, but also the AD much more efficient.[610, 611] If the user-defined derivative is written in pure PyTorch, the AD engine can directly compute higher derivatives. However, when using the C library *libcint*, all work must be conducted externally. To calculate higher-order derivatives, DQC's recursive strategy to connect integral (derivative) primitives and (higher-order) integral derivatives is adopted: The associated first derivative of the integral primitive is a primitive itself; the first derivative primitive is paired with the second-order integral derivative, which in turn is also a primitive, and the pattern continues. This scheme only hinges on the derivatives available in *libcint*, a collection that can be easily extended with the integrated automatic code generator. With the *libcint* interface, the overlap computation finally reaches computational efficiency on par with the Fortran implementation for all system sizes, as illustrated in Figure C.2 ("libcint interface"). Concerning memory consumption, obtaining the second-order derivative of vancomycin, for example, requires only 350 MB of (peak) memory compared with the 1.0 GB needed for the pure PyTorch version. Moreover, the interface

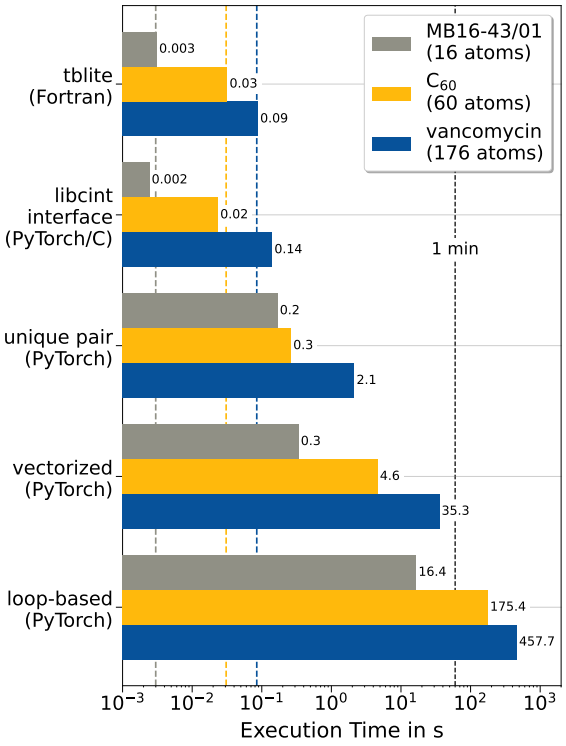

Figure C.2: Timings for various overlap integral implementations for three molecules, each belonging to different size regimes: a small "mindless"[624] 16-atom molecule from the MB16-43 benchmark set[89] with the sum formula $H_6B_2N_2O_2FNaAlCl$ (gray), the medium-sized C_{60} fullerene (yellow), and vancomycin, a drug containing 176 atoms (blue). Starting from the bottom, "loop-based" refers to the one-to-one translation from Fortran to PyTorch. "vectorized" describes the optimized McMurchie–Davidson algorithm. The best pure Python performance is reached with the "unique pair" ansatz. Only the version utilizing the "libcint interface" runs as efficient as the original Fortran "xtb" implementation. The colored dotted vertical lines also mark the Fortran reference speed. All timings are obtained on a single core. For more technical details, see Supporting Information.

allows effortless access to higher-order multipole integrals essential for evaluating molecular properties or multipole electrostatics.

In summary, the efficiency of a pure PyTorch integral implementation is inherently constrained by Python loops, resulting in execution times that are one to two orders of magnitude slower than the original Fortran version. However, integrating the high-performance *libcint* library into the AD engine yields the desired Fortran-level performance, while simultaneously granting access to arbitrary integrals and derivatives, perfectly complementing the extensible toolbox concept of *dxtb*.

C.3.3 Self-consistent Field Iterations

Particular attention must also be dedicated to the self-consistent field (SCF) or charge (SCC) procedure, which usually presents the bottleneck of tight-binding calculations due to its formally cubic time complexity. While *dxtb*'s SCF is nearly as efficient as the Fortran reference, since it primarily involves

efficiently handled matrix operations, AD of the SCF presents challenges. This issue even extends beyond the realm of quantum chemistry, because, from a mathematical standpoint, converging the SCF equates to identifying a fixed point $x^*(\theta)$, or more simply, root finding

$$f\left(x^*(\theta), \theta\right) = x^*(\theta) \tag{C.16}$$

$$g\left(x^*(\theta), \theta\right) = f\left(x^*(\theta), \theta\right) - x^*(\theta) = 0\,, \tag{C.17}$$

where θ represents the independent variables (parameters) with respect to which we differentiate. Such optimization problems are routinely solved with various iterative algorithms, but their AD only offers two pathways: Explicit differentiation of all iterations (unrolling)[319, 320] or implicit differentiation[320] of the optimality condition.

Explicit Differentiation

Unrolling iterations plainly exploits the AD engine, eliminating the need for derivative-specific code. Indeed, *dxtb* implements the regular SCF procedure with simple (linear) and Anderson mixing[625] for convergence acceleration. However, recording the computational graph through each iteration increases memory consumption linearly with the number of required iterations in reverse-mode AD. This poses an obstacle for calculations of large (batches of) systems, which are clearly in the scope of semiempirical methods. Minor improvements can be reached by "culling" in batched calculations:[321] We remove already converged systems from the corresponding batched arrays to reducing the array sizes and prevent over-convergence. A "perfect guess" short-cut was also put forward, in which the SCF is executed completely outside the computational graph and only re-connected with the converged charges, i.e. one extra SCF iteration with gradient tracking enabled and already converged charges as inputs is carried out.[321] Unfortunately, this ansatz does not yield exact derivatives, as the converged charges do not depend on previous charges anymore. In fact, Zhang and Chan demonstrated that a "perfect guess" produces the largest errors, because the iterative probing of the system's response incrementally improves the accuracy of the derivative.[532] While typical calculations require enough iterations to reproduce the analytical derivatives with explicit AD of the SCF, no rigorous measure for convergence can be evaluated or even accessed and a dependency on the initial guess remains.

An additional challenge for explicit differentiation originates from the repeated diagonalization of the Fock matrix in the SCF. Although stable derivatives of eigendecompositions exist,[610, 626] the reverse-mode backward pass is undefined for degenerate eigenvalues. In practice, this issue is circumvented by regularization schemes,[321, 327, 522, 528, 532, 542, 626, 627] albeit at the cost of introducing small errors in the derivatives. *dxtb* adopts Lorentzian and conditional broadening from TBMaLT[321] (see Supporting Information). Alternatively, the SCF iterations can be replaced by eigendecomposition-free direct minimization approaches.[326, 628]

Implicit Differentiation

In contrast, implicit differentiation, by virtue of the implicit function theorem, necessitates derivative information solely at the solution point, not throughout the entire iteration process.[320] By differentiating the optimality condition (Equation C.17) with respect to θ, the task reduces to solving a linear

system.

$$0 = \frac{\partial g\left(x^*, \theta\right)}{\partial \theta} + \frac{\partial g\left(x^*, \theta\right)}{\partial x^*} \frac{\partial x^*}{\partial \theta} \tag{C.18}$$

$$= \frac{\partial f\left(x^*, \theta\right)}{\partial \theta} + \frac{\partial f\left(x^*, \theta\right)}{\partial x^*} \frac{\partial x^*}{\partial \theta} - \frac{\partial x^*}{\partial \theta} \tag{C.19}$$

$$\underbrace{\frac{\partial x^*}{\partial \theta}}_{J} = -\underbrace{\left(\frac{\partial g\left(x^*, \theta\right)}{\partial x^*}\right)^{-1}}_{G^{-1}} \cdot \underbrace{\frac{\partial g\left(x^*, \theta\right)}{\partial \theta}}_{B_\mathrm{G}} \tag{C.20}$$

$$= \underbrace{\left(\mathbb{1} - \frac{\partial f\left(x^*, \theta\right)}{\partial x^*}\right)^{-1}}_{F^{-1}} \cdot \underbrace{\frac{\partial f\left(x^*, \theta\right)}{\partial \theta}}_{B_\mathrm{F}} \tag{C.21}$$

Note that we suppressed the implicit dependence of x^* on θ. Further technical simplifications are possible in reverse-mode AD (see Supporting Information).[532] Requiring only information at the solution point brings significant advantages for implicit differentiation. First and foremost, the guess-dependency vanishes and the error becomes well-defined, as it directly depends on the SCF convergence threshold. Secondly, the underlying iterative algorithm can be chosen freely, which paves the way for efficient convergence accelerators. Lastly, implicit differentiation only requires constant memory compared to the linear cost of iteration unrolling. Due to these notable benefits, libraries[327, 629–631] for user-friendly, black-box implicit differentiation are built on top of existing frameworks, supporting the recent transition from explicit[530, 538] to implicit[326, 532] differentiation. Additionally, implicit differentiation facilitates the construction of complex ML model.[632–634]

For our PyTorch-based tight-binding framework, we employ the *xitorch* library[327] that provides access to a differentiable fixed point solver (*equilibrium*) and other functionals. In the context of the SCF, the equilibrium function f comprises all operations of one SCF iteration (building the Fock matrix, diagonalization, etc.). The dependent variable x corresponds to the self-consistent variable, i.e. the quantity to converge. *dxtb* extends the possibility to iterate not only charges but also the potential and the Fock matrix, facilitating more convergence options and fine-tuning. Finally, the independent variables θ depict parametric dependencies, such as the nuclear coordinates or the tight-binding parametrization. Due to the considerable advantages of implicit differentiation, it is the default option in *dxtb* for routine automatic SCF differentiation. Note that *xitorch* already implements a set of convergence accelerators, namely linear and Broyden (default) mixing, which we extend with Anderson mixing.

Higher-order Derivatives

Due to the ubiquitous nature of derivatives in quantum chemistry,[518] access to Jacobians and higher-order derivatives, such as Hessians, is essential for *dxtb*. PyTorch, however, does not provide direct access to explicit derivative quantities, since the AD engine is primarily designed for the efficient computation of vector–Jacobian (or Jacobian–vector) in deep learning contexts, where full Jacobians

are often not required. While the Jacobian can be built row- or column-wise by supplying different unit vectors, PyTorch's novel composable function transforms offer a streamlined and vectorized alternative.[635] These modular, stackable operations return an altered or enhanced version of the original function, which, for example, computes its derivative or efficiently operates on a batch of inputs simultaneously. Function transforms can be arbitrarily combined for the (batched) computation of Hessians and higher-order derivatives.

Unfortunately, this feature is still under development and does not yet fully support some specific requirements and edge cases of *dxtb*. A notable limitation is the incompatibility with the *xitorch* library, which crucially handles implicit differentiation of the SCF. This necessitates a fallback to explicit differentiation methods for Jacobian and Hessian computations, resulting in increased memory demands. In the future, we aim to substitute *xitorch* with a tailored implicit SCF differentiation method. Concurrently, we plan to incorporate ongoing advancements in PyTorch to broaden *dxtb*'s capabilities and enhance its efficiency in handling derivative computations.

C.4 Results

C.4.1 Computational Efficiency

In order to realistically assess the computational efficiency of *dxtb*, we selected two representative test cases for tight-binding methods. First, we examine the computational demand for a large molecule, which is challenging for the routine application of more sophisticated methods due to prohibitive computational costs. This test highlights the scalability of our implementation and exposes potential bottlenecks. Second, we investigate the opposite case, i.e. the computation of large numbers of small molecules, which is relevant for screening applications, benchmarking and machine learning.

Both test scenarios include the computation of nuclear gradients, allowing us to compare the cost of AD in *dxtb* against analytical nuclear gradients from the *tblite*[328] library, a Fortran-based reference implementation for the GFN1-xTB method. We also explore *dxtb*'s efficiency regarding the overlap computation, presenting timings for three different approaches: a semivectorized, pure PyTorch version using either AD for gradient computation ("AD") or a custom analytical nuclear derivative ("analytical"), and an interface with the high-performance *libcint* integral library ("libcint").

Finally, we also explore the effectiveness of batched calculations and compare with the standard sequential evaluation of GFN1-xTB.

Note that although the code runs on and is tested on GPUs, the following results are obtained on CPUs. We provide some preliminary GPU timings in the Supporting Information (Figure S3, Figure S4), and a more comprehensive analysis will be the subject of future work, leveraging the recently launched state-of-the-art cluster at the University of Bonn.

Large Systems

We start by considering 2xHB238 – a large, 538-atom NCI complex from the LNCI16[329] benchmark set that consists of two dipolar donor-acceptor dye molecules deposited on a graphene sheet. In Figure C.3, we compare execution times for all approaches, including the *tblite* reference. We report single-core execution times (first, left bar) and four-core shared memory parallelization (second, right bar, hatched) results. The timings are broken down into the components of the tight-binding method, namely the electrostatics (SCF, blue), the overlap integral (yellow), the classical repulsion

(green), the D3(BJ) dispersion correction (red), and the nuclear derivative (gradient, gray). Since 2xHB238 does not contain halogens, the halogen-bond correction is not required, and hence, omitted. Although the Fortran code unsurprisingly outperforms the *dxtb* approaches, there are significant differences between the tight-binding contributions. Repulsion and dispersion calculations only consume a marginal fraction of the total execution time and run similarly fast in Fortran and Python thanks to full vectorization and precomputed parameters. The SCF, on the other hand, emerges as the computational bottleneck of the energy evaluation due to its formally cubic scaling. Note that the limiting diagonalization of the Fock matrix is performed by the same optimized linear algebra backend (LAPACK[636]) in both *tblite* and *dxtb*, yielding comparable timings and allowing straightforward parallelization. Finally, integral and analytical gradient evaluation show remarkable efficiency in *tblite*, achieving almost perfect parallelization by reducing the total execution time nearly fourfold when increasing from one to four cores (41 s to 14 s). Comparing the various integral approaches in *dxtb*, only the *libcint* interface matches Fortran's integral performance. The semivectorized integral algorithm in the "AD" and "analytical" approaches is significantly slower, barely benefits from simple parallelization, and necessitates explicit parallel code. Additionally, Python loops significantly hinder efficient AD, which is especially evident in "AD" (first bar, gray). While the "analytical" gradient reduces computational demands and parallelizes more effectively, it remains inferior to the "libcint" version. Moreover, implementing and optimizing an analytical gradient undermines the purpose of the AD framework, involving laborious coding while still suffering from Python-induced slowdowns in higher derivatives (cf. Figure S1). Finally, we note that all approaches are significantly faster than numerical differentiation: Considering the single-core timing for the singlepoint calculation in Fortran of 38.4 s, the $6N_{\text{atoms}}$ evaluations for the numerical gradient would require roughly 35 h.

Small Systems

For the investigation of small systems, we consider two datasets: QM9 and GMTKN55. QM9, comprising approximately 134,000 molecules with up to 29 atoms, includes only the elements H, C, N, O, and F.[170] Although GMTKN55 has significantly fewer structures (2,462 in total), it features larger molecules (up to 81 atoms) and a greater variety of elements.[89] Figure C.4 shows the distribution of single-core execution times for energy and nuclear gradient evaluation across the QM9 (left) and GMTKN55 (right) data sets. Consistent with the assessment for large molecules, we include the Fortran reference (blue) and the three PyTorch approaches ("libcint": yellow, "AD": red, "analytical": green). Starting with QM9, the Fortran implementation (blue) again surpasses all PyTorch-based implementations, averaging only 40 ms per molecule and requiring 1.5 h overall. The execution time is closely aligned with the distribution of molecular sizes, which is mostly symmetrical but with some irregularities in larger sizes. The "libcint" interface (yellow) provides the second-best efficiency, doubling (overall: 3.5 h) to tripling (average: 100 ms) the execution time. The pure PyTorch implementations display broader execution time distributions and are another factor of two slower. Interestingly, the "AD" method slightly outperforms the "analytical" integral derivatives, which differs from the large molecule tests. This can be attributed to shorter Python loops due to lower number of basis functions having a less pronounced impact on runtime, underscoring the scalability issue of Python loops. The GMTKN55 results mirror those of QM9, with *tblite* and "libcint" showing sharp execution time distributions and "AD" and "analytical" exhibiting broader distributions shifted to longer times. The right-skewed molecular size distribution in GMTKN55 results in long tails for the timings of the pure PyTorch implementations, hinting towards the scalability challenges of the overlap

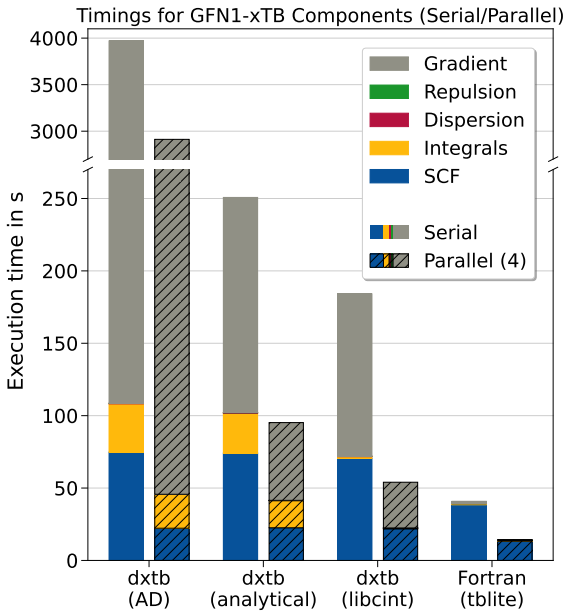

Figure C.3: Execution times (in seconds) for energy and nuclear gradient evaluation for a large, 538-atom NCI complex. The first (left) bar of each category displays single-core timings, the second (right) bars are obtained on four cores with shared-memory parallelization. The four categories (from left to right) reflect the pure PyTorch implementation in *dxtb* with AD for overlap derivatives, another pure PyTorch approach with analytical nuclear overlap derivatives, *dxtb* with an interface to *libcint* for all integral-related computation, and finally, the analytical Fortran implementation from the *tblite* library.

implementation. However, average timings across all methods are comparable between QM9 and GMTKN55.

Batched Evaluation

To enhance *dxtb*'s synergy with ML approaches, the framework fully supports batched calculations, allowing multiple inputs to be processed simultaneously. In contrast to sequential processing, where all target systems are fed into the program separately, batching can optimize computational resource usage by stacking inputs for evaluation with a single program call. To show the efficiency of the batched approach, we investigate the execution times of energy calculations for four cases covering different molecular and batch sizes. First, we draw a random subset of 1000 and 2000 molecules from the QM9 data set. Second, we use *crest*[114, 115] to create a large conformer ensemble of the highly flexible alkane *n*-icosane and a small conformer ensemble of vancomycin, containing 586 and 50 conformers, respectively. Note that due to different system sizes in QM9, the batched tensor contains padding, which is not required for the conformer sets. We present single-core execution times as well as timings for shared-memory parallelism using four cores in Table C.1.

Starting with small systems (QM9 subsets), we find that batched calculations are significantly faster for both single-core and parallel execution. This efficiency gain can be largely attributed to the elimination of repeated setup and overhead inherent in sequential processing. Notably, sequential

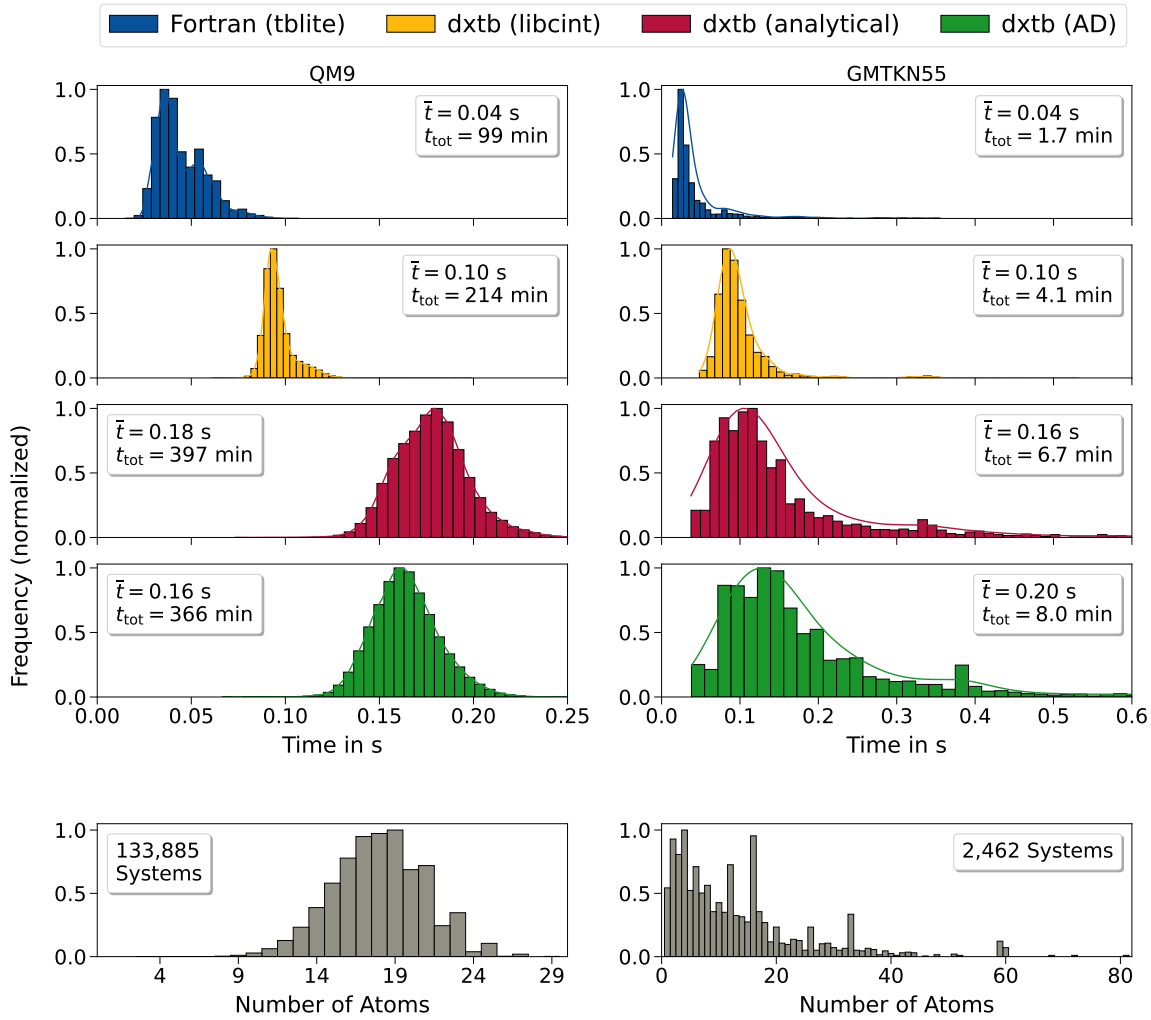

Figure C.4: Distribution of execution times (in seconds) for energy and nuclear gradient calculations across the QM9 (left) and GMTKN55 (right) data sets. In the bottom panel, the distribution of molecular sizes in the data sets is shown. Execution times are obtained with the Fortran reference (*tblite*, blue), *dxtb* with the *libcint* interface (yellow), and the pure PyTorch implementations using either AD for overlap gradients (red) or an analytical derivative (green).

evaluation of large batches of molecules does not benefit from parallelization. In fact, utilizing four cores leads to slightly longer runtimes due to communication overhead. Batching, on the other hand, clearly benefits from parallel execution. Moving to larger system sizes (n-icosane, vancomycin), the differences between sequential and batched evaluation decrease for single-core execution, mainly because the diagonalization of the tight-binding Fock matrix becomes the clear bottleneck. In other words, the proportion of overhead in the total execution time decreases. Nevertheless, in sequential evaluation, n-icosane barely benefits from parallelization and only vancomycin exhibits an appreciable speed-up. This shows that a certain system size is required to offset the setup and communication overhead. Clearly, the batched approach scales better across all four cases, highlighting its suitability

System	Number of Atoms	Batch Size	Sequential		Batched	
			1 Core	4 Cores	1 Core	4 Cores
QM9 subset	27	1000	55.4	64.0	22.2	15.9
QM9 subset	27	2000	111.0	127.6	46.4	31.9
n-icosane	62	586	65.4	61.9	62.9	33.8
vancomycin	176	50	81.3	42.0	93.2	37.0

Table C.1: CPU execution time (in seconds) for sequential and batched energy calculations using a single core and shared-memory parallelism with 4 cores. The QM9 subset is obtained by randomly selecting 1000 and 2000 molecules from the whole set. Note that different system sizes are expanded by padding, i.e. the number of atoms corresponds to the padded size. The conformer ensembles of *n*-icosane and vancomycin are generated with *crest*.[114, 115]

for parallel computing environments. Finally, to validate the practicality of batching in terms of memory usage, we measured the memory consumption for the first-order AD nuclear derivative for all four ensembles in batch mode. We find peak memory requirements of 3.7 GB and 7.1 GB for the QM9 subsets with 1000 and 2000 structures, respectively. For the *n*-icosane conformers, 10.9 GB are necessary, while the vancomycin ensemble utilizes 9.5 GB. Given the large system sizes as well as the cost of reverse-mode AD, these results are both expected and manageable.

In conclusion, *dxtb* with the "libcint" interface achieves an efficiency close to the (compiler-)optimized Fortran code, particularly for energy evaluations. The framework can be applied across different size regimes for high-throughput calculations as well as large molecules, and benefits from straightforward shared-memory parallelization. Employing batching in parallel computations further enhances performance, making execution on massively parallel hardware particularly effective. While pure PyTorch implementations provide reasonable efficiency for small molecules, scalability is always limited by the merely partially vectorized integral code, which extends to the computation of (higher) derivatives. As expected, AD-based gradient computation generally lags behind analytical Fortran gradients. However, we only observe an increase of computation time by a factor of two to five. Considering the AD engine's inability to recognize simplifications or zero-valued terms (e.g. due to the self-consistency of the SCF solution), these performance shortcomings are the cost of convenient derivative calculation, which we will demonstrate for molecular properties next.

C.4.2 Molecular Properties

AD facilitates the computation of molecular properties, defined as derivatives of the total energy E with respect to a perturbation. Within the Born–Oppenheimer approximation, geometric derivatives are of central interest, namely gradients and Hessians. These derivatives are pivotal for characterizing stationary points on the potential energy surface, a necessity for tasks such as geometry optimization and transition state searches. Moreover, an eigendecomposition of the mass-weighted Hessian $\mathbf{H}^{\mathrm{MW}}$

yields harmonic vibrational frequencies ω and the corresponding normal modes $\mathbf{q}$

$$\mathbf{H}^{\mathrm{MW}}\mathbf{q} = \omega\mathbf{q} \tag{C.22}$$

$$\text{with} \quad \mathbf{H}^{\mathrm{MW}} = \mathbf{M}^{-1}\mathbf{H}\mathbf{M}^{-1} = \mathbf{M}^{-1}\left(\frac{\partial^2 E}{\partial \mathbf{R}^2}\right)\mathbf{M}^{-1}, \tag{C.23}$$

where $\mathbf{M}$ represents a diagonal matrix containing the square root of the atomic masses, and $\mathbf{R}$ is the $n \times 3$ matrix of nuclear Cartesian coordinates.

To demonstrate the effectiveness of AD, we perform a vibrational analysis on the planar ammonia molecule (transition state in the umbrella inversion), using GFN1-xTB within *dxtb*. Notably, this represents the first instance of employing a non-numerical (exact) Hessian within xTB, as the Fortran implementations do not support higher analytical derivatives. The calculated vibrational frequencies, listed in Table C.2, successfully capture the characteristic imaginary frequency. Furthermore, they align perfectly with the numerical approach (finite-differences Hessian) and are in good agreement with the DFT reference frequencies (ωB97X-D4[300–302, 484, 637]/def2-QZVP[638]).

frequency	*dxtb* AD	*dxtb* numerical	DFT
1	**-1115**	**-1115**	**-1130**
2	1276	1276	1410
3	1392	1392	1489
4	3615	3615	3730
5	4409	4409	4408
6	4499	4499	4526

Table C.2: Vibrational frequencies in cm^{-1} for planar ammonia. For *dxtb* (GFN1-xTB), the Hessian is calculated both with automatic differentiation (AD) and using finite differences (numerical). The DFT reference employs the ωB97X-D4[300–302, 484, 637]/def2-QZVP[638] level of theory. For the vibrational analysis, the translational and rotational modes are projected out.

Another class of important (static) properties arises from the response of the system to an external electric field ε, with practical applications in spectroscopy and non-linear optics.[639–641] The first-order response is given by the permanent electric dipole moment μ, which is defined as the first derivative of the total energy with respect to the external electric field. The second derivative yields the electric dipole polarizability tensor α, and the third-order derivative gives the electric dipole hyperpolarizability tensor β.

$$\mu = -\frac{\partial E}{\partial \varepsilon} \tag{C.24}$$

$$\alpha = -\frac{\partial^2 E}{\partial \varepsilon^2} = \frac{\partial \mu}{\partial \varepsilon} \tag{C.25}$$

$$\beta = -\frac{\partial^3 E}{\partial \varepsilon^3} = \frac{\partial \mu^2}{\partial \varepsilon^2} = \frac{\partial \alpha}{\partial \varepsilon} \tag{C.26}$$

While rarely practically relevant, even higher derivatives are readily accessible via AD. Note that

dipole integrals are required for all properties related to the electric field, necessitating the *libcint* backend. Combining both types of derivatives, spectroscopic intensities can be computed within the double harmonic approximation from the first-order geometric derivative of the respective polarization property. In particular, the square of the first-order response of the electric dipole moment and the electric dipole polarizability tensor with respect to a displacement along the normal coordinate determines the infrared (IR) and Raman intensities, I_{IR} and I_{Raman}, respectively.

$$I_{\mathrm{IR}} \propto \left(\frac{\partial \mu}{\partial \mathbf{q}}\right)^2 \propto \left(\frac{\partial^2 E}{\partial \mathbf{R} \partial \varepsilon}\right)^2 \tag{C.27}$$

$$I_{\mathrm{Raman}} \propto \left(\frac{\partial \alpha}{\partial \mathbf{q}}\right)^2 \propto \left(\frac{\partial^3 E}{\partial \mathbf{R} \partial \varepsilon^2}\right)^2 \quad \left(= \chi^2\right) \tag{C.28}$$

Hence, I_{IR} and I_{Raman} are second and third-order properties relating to mixed electric field and geometric derivatives. In Raman spectroscopy, this derivative is often denoted as the Raman susceptibility tensor χ. Detailed formulas[642–644] are given in the Supporting Information. Figure C.5 displays the IR spectrum of capsaicin (spice compound in chili peppers), calculated with GFN1-xTB (*dxtb*, blue) and, for a qualitative comparison, with DFT (ωB97X-D4/def2-QZVP, yellow). The outline of the *dxtb* spectrum is obtained from a fully numerical Hessian, electric dipole moment and geometric dipole derivative, showing perfect agreement with the result from AD. Considering the level of theory, the GFN1-xTB spectrum is also in reasonable agreement with the DFT reference.[645] Although Raman spectra are also accessible, semiempirical methods with (mostly) minimal basis sets are usually not accurate enough,[646–648] and at least polarized double-ζ-type basis sets are necessary.[151] Therefore, the corresponding Raman spectrum of capsaicin is only shown in the Supporting Information for completeness (Figure S6).

C.5 Summary and Outlook

To further advance the applicability of semiempirical quantum chemical methods, we introduced *dxtb* – a fully differentiable framework for extended tight-binding methods. Our PyTorch-based[548] implementation enables the evaluation of arbitrary-order derivatives through automatic differentiation. We have demonstrated its effectiveness by obtaining various molecular properties and spectroscopic quantities without the need for explicit derivative code. Leveraging the access to arbitrary derivatives within *dxtb*, we emphasize the potential for streamlined parameter optimization of new tight-binding methods and the re-parameterization of existing approaches for specialized applications. Moreover, *dxtb*'s integration with the PyTorch framework facilitates the combination of machine learning with semiempirical quantum chemistry. Such hybrid models offer improved accuracy, reduced data requirements and better transferability, which makes them a promising approach for the exploration of chemical space.

To retain the favorable computational cost of semiempirical methods, we dedicated significant efforts towards vectorization and code optimization. While our pure PyTorch implementation already ensures respectable efficiency, outsourcing the limiting integral calculation to a differentiable *libcint*[324] interface significantly improves execution times for energy evaluations, closely matching those of the Fortran reference implementation for both small and large molecules. Automatic nuclear derivatives

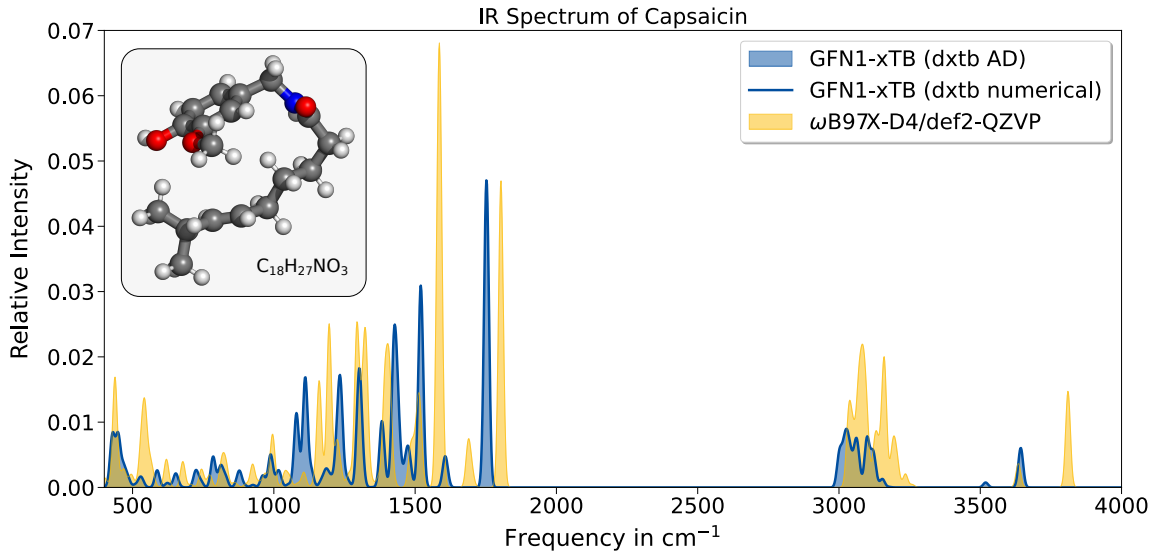

Figure C.5: IR spectrum of capsaicin. The blue spectrum is calculated with GFN1-xTB using *dxtb* with automatic differentiation (AD). Its outline, however, is computed fully numerically, showing perfect agreement. For reference, a DFT (ωB97X-D4/def2-QZVP) spectrum is also shown (yellow).

are only two to five times slower than their analytical (Fortran) counterparts, underscoring the effectiveness of automatic differentiation and Python-based frameworks in quantum chemistry. Further speed-ups can be achieved through parallel execution, especially in combination with batching.

dxtb is designed as a modular, open-source framework to maximize user convenience, encourage rapid prototyping, and ease development. While the current version of *dxtb* focuses on GFN1-xTB,[17] an extension to other tight-binding methods (GFN2-xTB[18]) is straightforward and will be the subject of future work. Furthermore, we plan on improving GPU performance to advance the application of extended tight-binding methods to state-of-the-art massively parallel hardware. We will also work on reducing memory demands in automatic differentiation to broaden the framework's applicability to larger systems and more complex derivative computations. Ultimately, our goal is for *dxtb* to unlock the full potential of semiempirical tight-binding methods and serve as a catalyst for developing (hybrid) machine learning applications.

Acknowledgement

M. Friede thanks T. Froitzheim for many fruitful discussions. This work was supported by the DFG in the framework of the priority program SPP 2363 on "Utilization and Development of Machine Learning for Molecular Applications – Molecular Machine Learning" (Project No. 497190956).

Data Availability

dxtb is published as an open-source package with a permissive Apache-2.0 license at `https://github.com/grimme-lab/dxtb`. It can easily be installed using either Python's package-management system

pip or the language-agnostic package manager *conda*. The source code is available on GitHub[649] and extensively documented.[607] To improve accessibility, *dxtb* runs under Python 3.8 and all higher versions and is compatible with PyTorch 1.11.0 and above. At the time of writing, the supported version comprise Python 3.8, 3.9, 3.10, 3.11 and PyTorch 1.11.0, 1.12.x, 1.13.x, 2.0.x, 2.1.x, 2.2.x. Besides its library functionality, *dxtb* also ships with a convenient command line interface that directly allows running calculations. The whole program is accompanied by an extensive test suite, where we check against the original implementations[328, 485, 486, 650] but also verify the correct treatment of erroneous inputs and exception handling.

On a side note: We acknowledged that a PyTorch implementation of the DFT-D3 and DFT-D4 dispersion models as well as the electronegativity equilibration model (EEQ) may be useful outside *dxtb*. Hence, we provide all projects as standalone libraries,[651–653] which are again accessible through *pip* and *conda*. Common utility functions are also collected in a separate project. All PyTorch-related implementations can be found at `https://github.com/tad-mctc`.

The data that support the findings of the study, as well as additional information, are available within the article and its supporting material.

Bibliography

[1] W. S. McCulloch and W. Pitts, Bull. Mat. Biophys. **5** (1943) 115.

[2] F. Rosenblatt, Psycho. Rev. **65** (1958) 386.

[3] D. Crevier, Bull. of Sci., Technol. & Soc. **14** (1993) 224.

[4] J. Jumper et al., Nature **596** (2021) 583.

[5] A. Collaboration, Phys. Lett. B **726** (2013) 88.

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

[7] Z. Qiao et al., J. Chem. Phys. **153** (2020) 124111.

[8] D. Anstine, R. Zubatyuk and O. Isayev, ChemRxiv (2023) 296.

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

[10] W. Heitler and F. London, Z. Phys. **44** (1927) 455.

[11] W. Kohn and L. J. Sham, Phys. Rev. **140** (1965) A1133.

[12] R. G. Parr and Y. Weitao, *Density-Functional Theory of Atoms and Molecules*, Oxford University Press, **1995**.

[13] A. V. Sadybekov and V. Katritch, Nature **616** (2023) 673.

[14] D. Dell'Angelo, *Computational chemistry and the study and design of catalysts*, Elsevier, **2022**.

[15] R. LeSar, *Introduction to Computational Materials Science: Fundamentals to Applications*, Cambridge University Press, **2013**.

[16] S. Spicher and S. Grimme, Angew. Chem. Int. Ed. **59** (2020) 15665.

[17] S. Grimme, C. Bannwarth and P. Shushkov, J. Chem. Theory Comput. **13** (2017) 1989.

[18] C. Bannwarth, S. Ehlert and S. Grimme, J. Chem. Theory Comput. **15** (2019) 1652.

[19] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry*, Dover Publications, **1996**.

[20] F. Jensen, *Introduction to Computational Chemistry*, Wiley & Sons Ltd, **2017**.

[21] K. Kamalasanan and C. Sharma, *Nanomedicine in Translational Research*, Academic Press, **2024**.

[22] I. Newton, *Philosophiæ Naturalis Principia Mathematica*, Royal Society, **1687**.

[23] P. A. M. Dirac, Math. Proc. Cambridge Phil. Soc. **35** (1939) 416.

[24] P. A. M. Dirac and R. H. Fowler, Proc. R. Soc. London, Ser. A **117** (1928) 610.

[25] P. Pyykko, Chem. Rev. **88** (1988) 563.

[26] P. Schwerdtfeger, Chem. Phys. Chem. **12** (2011) 3143.

[27] F. Calvo et al., Angew. Chem. Int. Ed. **52** (2013) 7583.

[28] C. Addison, *The Chemistry of the Liquid Alkali Metals*, Wiley & Sons Ltd, **1984**.

[29] R. Ahuja et al., Phys. Rev. Lett. **106** (2011) 018301.

[30] A. Sommerfeld, Naturwissenschaften **28** (1940) 417.

[31] A. Sommerfeld, *Atombau und Spektrallinien*, Friedrich Vieweg und Sohn, **1919**.

[32] W. E. Lamb and R. C. Retherford, Phys. Rev. **72** (1947) 241.

[33] E. Fermi, Z. Phys. **60** (1930) 320.

[34] H. Schüler and T. Schmidt, Z. Phys. **94** (1935) 457.

[35] A. Böhm, *Quantum Mechanics*, Springer, **1979**.

[36] G. E. Moore, Electronics **38** (1965) 114.

[37] M. Born and R. Oppenheimer, Ann. Phys. **389** (1927) 457.

[38] J. C. Slater, Phys. Rev. **36** (1930) 57.

[39] S. F. Boys and A. C. Egerton, Proc. R. Soc. London, Ser. A **200** (1950) 542.

[40] F. Neese, WIREs Comput. Mol. Sci. **2** (2012) 73.

[41] F. Furche et al., WIREs Comput. Mol. Sci. **4** (2014) 91.

[42] H. B. Schlegel and M. J. Frisch, Int. J. Quant. Chem. **54** (1995) 83.

[43] W. J. Hehre, R. F. Stewart and J. A. Pople, J. Chem. Phys. **51** (1969) 2657.

[44] W. J. Hehre, R. Ditchfield and J. A. Pople, J. Chem. Phys. **56** (1972) 2257.

[45] F. Weigend and R. Ahlrichs, Phys. Chem. Chem. Phys. **7** (2005) 3297.

[46] T. H. Dunning Jr., J. Chem. Phys. **90** (1989) 1007.

[47] R. A. Kendall, T. H. Dunning Jr. and R. J. Harrison, J. Chem. Phys. **96** (1992) 6796.

[48] G. Kresse and J. Furthmüller, Phys. Rev. B **54** (1996) 11169.

[49] C. Cerjan, *Numerical Grid Methods and Their Application to Schrödinger's Equation*, Springer, **1993**.

[50] R. Behera and M. Mehra, J. of Mult. Model. **06** (2015) 1450001.

[51] H. Kruse and S. Grimme, J. Chem. Phys. **136** (2012) 04B613.

[52] J. G. Brandenburg et al., J. Phys. Chem. A **117** (2013) 9282.

[53] D. R. Hartree, Math. Proc. Cambridge Phil. Soc. **24** (1928) 89.

[54] V. Fock, Z. Phys. **61** (1930) 126.

[55] J. C. Slater, Phys. Rev. **34** (1929) 1293.

[56] W. Heisenberg, Z. Phys. **38** (1926) 411.

[57] P. A. M. Dirac and R. H. Fowler, Proc. R. Soc. London, Ser. A **112** (1926) 661.

[58] W. Pauli, Phys. Rev. **58** (1940) 716.

[59] C. C. J. Roothaan, Rev. Mod. Phys. **23** (1951) 69.

[60] G. Hall, Proc. R. Soc. London, Ser. A **205** (1951) 541.

[61] J.-M. Mewes, *Development and Application of Methods for the Description of Photochemical Processes in Condensed Phase*, PhD thesis: Ruprecht-Karls-Universität Heidelberg, **2015**.

[62] P.-O. Löwdin, Phys. Rev. **97** (1955) 1509.

[63] E. Wigner, Phys. Rev. **46** (1934) 1002.

[64] L. H. Thomas, Math. Proc. Cambridge Phil. Soc. **23** (1927) 542.

[65] P. Hohenberg and W. Kohn, Phys. Rev. **136** (1964) B864.

[66] W. Koch and M. C. Holthausen, *A Chemist's Guide to Density Functional Theory*, Wiley & Sons Ltd, **2001**.

[67] T. Ziegler, Chem. Rev. **91** (1991) 651.

[68] A. D. Becke, J. Chem. Phys. **98** (1993) 5648.

[69] A. M. Teale et al., Phys. Chem. Chem. Phys. **24** (2022) 28700.

[70] G. Giuliani and G. Vignale, *Quantum Theory of the Electron Liquid*, Cambridge University Press, **2005**.

[71] A. Sommerfeld, Z. Phys. **47** (1928) 1.

[72] M. Gell-Mann and K. A. Brueckner, Phys. Rev. **106** (1957) 364.

[73] J. P. Perdew, Phys. Rev. B **33** (1986) 8822.

[74] A. D. Becke, Phys. Rev. A **38** (1988) 3098.

[75] M. Nightingale and C. J. Umrigar, *Quantum Monte Carlo Methods in Physics and Chemistry*, Springer, **1999**.

[76] B. L. Hammond, W. A. Lester and P. J. Reynolds, *Monte Carlo Methods in Ab Initio Quantum Chemistry*, World Scientific, **1994**.

[77] D. M. Ceperley and B. J. Alder, Phys. Rev. Lett. **45** (1980) 566.

[78] P. A. M. Dirac, Math. Proc. Cambridge Phil. Soc. **26** (1930) 376.

[79] J. C. Slater, Phys. Rev. **81** (1951) 385.

[80] S. H. Vosko, L. Wilk and M. Nusair, Canadian J. Phys. **58** (1980) 1200.

[81] E. Engel and R. M. Dreizler, *Density Functional Theory: An Advanced Course*, Springer, **2011**.

[82] J. P. Perdew, K. Burke and M. Ernzerhof, Phys. Rev. Lett. **77** (1996) 3865.

[83] S. F. Sousa, P. A. Fernandes and M. J. Ramos, J. Phys. Chem. A **111** (2007) 10439.

[84] J. Tao et al., Phys. Rev. Lett. **91** (2003) 146401.

[85] J. W. Furness et al., J. Phys. Chem. Lett. **11** (2020) 8208.

[86] M. Bursch et al., Angew. Chem. Int. Ed. **61** (2022) e202205735.

[87] K. Eichkorn et al., Chem. Phys. Lett. **242** (1995) 652.

[88] J. Harris, Phys. Rev. A **29** (1984) 1648.

[89] L. Goerigk et al., Phys. Chem. Chem. Phys. **19** (2017) 32184.

[90] P. J. Stephens et al., J. Phys. C **98** (1994) 11623.

[91] C. Adamo and V. Barone, J. Chem. Phys. **110** (1999) 6158.

[92] J. P. Perdew, M. Ernzerhof and K. Burke, J. Chem. Phys. **105** (1996) 9982.

[93] V. N. Staroverov et al., J. Chem. Phys. **119** (2003) 12129.

[94] V. N. Staroverov et al., J. Chem. Phys. **121** (2004) 11507.

[95] J.-D. Chai and M. Head-Gordon, J. Chem. Phys. **128** (2008) 084106.

[96] J.-D. Chai and M. Head-Gordon, Phys. Chem. Chem. Phys. **10** (2008) 6615.

[97] E. Livshits and R. Baer, Phys. Chem. Chem. Phys. **9** (2007) 2932.

[98] T. Koopmans, Physica **1** (1934) 104.

[99] N. Mardirossian and M. Head-Gordon, Phys. Chem. Chem. Phys. **16** (2014) 9904.

[100] J. Heyd, G. E. Scuseria and M. Ernzerhof, J. Chem. Phys. **124** (2006) 219906.

[101] J. Heyd et al., J. Chem. Phys. **123** (2005) 174101.

[102] T. M. Henderson et al., J. Chem. Theory Comput. **4** (2008) 1254.

[103] S. Grimme, J. Chem. Phys. **118** (2003) 9095.

[104] S. Grimme, J. Chem. Phys. **124** (2006) 034108.

[105] F. Neese, T. Schwabe and S. Grimme, J. Chem. Phys. **126** (2007) 124115.

[106] S. Grimme and F. Neese, J. Chem. Phys. **127** (2007) 154116.

[107] C. Møller and M. S. Plesset, Phys. Rev. **46** (1934) 618.

[108] M. Bursch, *Evaluation and Application of Efficient Quantum Chemical Methods for Sophisticated Simulation of Inorganic Molecular Chemistry*,
PhD thesis: Rheinische Friedrich-Wilhelms-Universität Bonn, **2021**.

[109] S. Dohm et al., J. Chem. Theory Comput. **14** (2018) 2596.

[110] L. E. Ratcliff et al., WIREs Comput. Mol. Sci. **7** (2017) e1290.

[111] G. Hanschmann and H.-J. Köhler, Z. Phys. Chem. **255** (1974) 1192.

[112] P. Birner, H.-J. Hofmann and C. Weiss, "Die Elektronenkorrelation",
MO-theoretische Methoden in der organischen Chemie, De Gruyter, **1979** 81.

[113] N. D. Yilmazer and M. Korth, Comp. Struc. Biotech. J. **13** (2015) 169.

[114] P. Pracht, F. Bohle and S. Grimme, Phys. Chem. Chem. Phys. **22** (2020) 7169.

[115] P. Pracht et al., J. Chem. Phys. **160** (2024) 114110.

[116] S. Grimme et al., J. Phys. Chem. A **125** (2021) 4039.

[117] E. Hückel, Z. Phys. **70** (1931) 204.

[118] E. Hückel, Z. Phys. **72** (1931) 310.

[119] C. Coulson et al., *Hückel Theory for Organic Chemists*, Academic Press, **1978**.

[120] D. G. Evans, Inorg. Chem. **25** (1986) 4602.

[121] R. Hoffmann, J. Chem. Phys. **39** (1963) 1397.

[122] W. Thiel, WIREs Comput. Mol. Sci. **4** (2014) 145.

[123] J. C. Slater and G. F. Koster, Phys. Rev. **94** (1954) 1498.

[124] W. Harrison,
Electronic Structure and the Properties of Solids: The Physics of the Chemical Bond,
Dover Publications, **1989**.

[125] K. Ohno, K. Esfarjani and Y. Kawazoe,
Computational Materials Science: From Ab Initio to Monte Carlo Methods, Springer, **2018**.

[126] N. W. Ashcroft and N. D. Mermin, *Solid State Physics*, Harcourt, **1976**.

[127] J. A. Pople and D. L. Beveridge, *Approximate Molecular Orbital Theory*,
McGraw-Hill, **1970**.

[128] P. O. Dral, *Quantum Chemistry in the Age of Machine Learning*, Elsevier, **2022**.

[129] M. J. S. Dewar and W. Thiel, J. Am. Chem. Soc. **99** (1977) 4899.

[130] M. J. S. Dewar and W. Thiel, J. Am. Chem. Soc. **99** (1977) 4907.

[131] M. J. S. Dewar et al., J. Am. Chem. Soc. **107** (1985) 3902.

[132] J. J. P. Stewart, J. Comput. Chem. **10** (1989) 209.

[133] J. J. P. Stewart, J. Comput. Chem. **10** (1989) 221.

[134] J. J. P. Stewart, J. Mol. Model. **13** (2007) 1173.

[135] J. J. P. Stewart, J. Mol. Model. **19** (2012) 1.

[136] J. Řezáč et al., J. Chem. Theory Comput. **5** (2009) 1749.

[137] D. Porezag et al., Phys. Rev. B **51** (1995) 12947.

[138] G. Seifert, D. Porezag and T. Frauenheim, Int. J. Quantum Chem. **58** (1996) 185.

[139] A. S. Christensen et al., Chem. Rev. **116** (2016) 5301.

[140] B. Aradi, B. Hourahine and T. Frauenheim, J. Phys. Chem. A **111** (2007) 5678.

[141] M. Elstner et al., Phys. Rev. B **58** (1998) 7260.

[142] Y. Yang et al., J. Phys. Chem. A **111** (2007) 10861.

[143] M. Gaus, Q. Cui and M. Elstner, J. Chem. Theory Comput. **7** (2011) 931.

[144] B. Hourahine et al., J. Chem. Phys. **152** (2020) 124101.

[145] M. Elstner et al., Chem. Phys. **263** (2001) 203.

[146] S. Grimme et al., J. Chem. Phys. **132** (2010) 154104.

[147] J. Řezáč, J. Comput. Chem. **40** (2019) 1633.

[148] C. Bannwarth et al., WIREs Comput. Mol. Sci. **11** (2021) e1493.

[149] P. Pracht et al., ChemRxiv (2019) 8326202.

[150] M. Hülsen, A. Weigand and M. Dolg, Theor. Chem. Acc. **122** (2009) 23.

[151] S. Grimme, M. Müller and A. Hansen, J. Chem. Phys. **158** (2023) 124111.

[152] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, **2016**.

[153] T. M. Mitchell, *Machine Learning*, McGraw-Hill, **1997**.

[154] J. McCarthy et al.,
A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence, **1955**.

[155] M. Rupp et al., Phys. Rev. Lett. **108** (2012) 058301.

[156] F. Noé et al., Annu. Rev. of Physical Chem. **71** (2020) 361.

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

[158] A. Glielmo et al., Chem. Rev. **121** (2021) 9722.

[159] H. Hadipour et al., BMC Bioinformatics **23** (2022) 132.

[160] J. Schmidhuber, Neural Networks **61** (2015) 85.

[161] I. Goodfellow et al., "Generative Adversarial Nets",
Conference on Neural Information Processing Systems, **2014**.

[162] J. Sohl-Dickstein et al.,
"Deep Unsupervised Learning using Nonequilibrium Thermodynamics",
Conference on Machine Learning, **2015**.

[163] A. Merchant et al., Nature **624** (2023) 80.

[164] A. A. Volk et al., Nature Communications **14** (2023) 1403.

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

[166] J. C. Snyder et al., J. Chem. Phys. **139** (2013) 224104.

[167] F. Brockherde et al., Nature Communications **8** (2017) 872.

[168] R. Nagai, R. Akashi and O. Sugino, npj Comput. Mater. **6** (2020) 43.

[169] L. Ruddigkeit et al., J. Chem. Inf. Model. **52** (2012) 2864.

[170] R. Ramakrishnan et al., Scientific Data **1** (2014) 140022.

[171] S. Axelrod and R. Gómez-Bombarelli, Scientific Data **9** (2022) 185.

[172] L. Medrano Sandonas et al., Scientific Data **11** (2024) 742.

[173] K. Spiekermann, L. Pattanaik and W. H. Green, Scientific Data **9** (2022) 417.

[174] D. Lowe, *Chemical reactions from US patents*, **2017**.

[175] D. Weininger, J. Chem. Inf. Comp. Sci. **28** (1988) 31.

[176] D. Weininger, A. Weininger and J. L. Weininger, J. Chem. Inf. Comp. Sci. **29** (1989) 97.

[177] D. Weininger, J. Chem. Inf. Comp. Sci. **30** (1990) 237.

[178] M. Glavatskikh et al., Journal of Cheminformatics **11** (2019) 69.

[179] S. Kaufman et al., ACM Trans. Knowl. Discov. Data **6** (2012).

[180] P. Eastman et al., Scientific Data **10** (2023) 11.

[181] P. Eastman et al., J. Chem. Theory Comput. **20** (2024) 8583.

[182] D. Balcells and B. B. Skjelstad, J. Chem. Inf. Model. **60** (2020) 6135.

[183] C. Hölzer et al., J. Chem. Inf. Model. **64** (2024) 825.

[184] L. Wittmann et al., Phys. Chem. Chem. Phys. **26** (2024) 21379.

[185] A. Najibi and L. Goerigk, J. Comput. Chem. **41** (2020) 2562.

[186] J. S. Smith, O. Isayev and A. E. Roitberg, Scientific Data **4** (2017) 170193.

[187] J. Řezáč, K. E. Riley and P. Hobza, J. Chem. Theory Comput. **7** (2011) 2427.

[188] A. S. Christensen and O. A. von Lilienfeld, arXiv **2007** (2020) 09593.

[189] T. Hastie, R. Tibshirani and J. Friedman,
The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, **2009**.

[190] J. A. Hanley et al., The American Statistician **55** (2001) 150.

[191] S. Boyd et al., *Subgradients*, Lecture notes for EE364b, Stanford University, **2022**.

[192] J. R. Terven et al., arXiv **2307** (2024) 02694.

[193] P. J. Huber, The Annals of Mathematical Statistics **35** (1964) 73.

[194] R. A. Saleh and A. K. M. E. Saleh, arXiv **2208** (2024) 04564.

[195] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*, MIT Press, **2005**.

[196] D. P. Kingma and J. Ba, arXiv **1412** (2014) 6980.

[197] I. Newton, *De analysi per aequationes numero terminorum infinitas*, ed. by W. Jones,
Samuel Smith and Benjamin Walford, **1711**.

[198] J. Wallis, *A Treatise of Algebra, both Historical and Practical*, John Playford, **1685**.

[199] J. Raphson, *Analysis AEquationum Universalis*, Thomas Bradyll, **1697**.

[200] C. G. Broyden, Journal of the Institute of Mathematics and Its Applications **6** (1970) 76.

[201] R. Fletcher, The Computer Journal **13** (1970) 317.

[202] D. Goldfarb, Mathematics of Computation **24** (1970) 23.

[203] D. F. Shanno, Mathematics of Computation **24** (1970) 647.

[204] D. C. Liu and J. Nocedal, Mathematical Programming **45** (1989) 503.

[205] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*, Springer, **2006**.

[206] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, **1996**.

[207] S. J. Reddi, S. Kale and S. Kumar, "On the Convergence of Adam and Beyond",
Conference on Learning Representations, **2018**.

[208] Y. Zhang et al., "Adam Can Converge Without Any Modification On Update Rules",
Conference on Neural Information Processing Systems, **2022**.

[209] K. O. Stanley and R. Miikkulainen, Evolutionary computation **10** (2002) 99.

[210] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Nature **323** (1986) 533.

[211] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", *Conference on Machine Learning*, **2015**.

[212] S. Hochreiter and J. Schmidhuber, Neural Computation **9** (1997) 1735.

[213] S. Hochreiter, *Untersuchungen zu dynamischen neuronalen Netzen*, MA thesis: Technische Universität München, **1991**.

[214] K. He et al., "Deep Residual Learning for Image Recognition", *Conference on Computer Vision and Pattern Recognition*, **2016**.

[215] A. Veit, M. Wilber and S. Belongie, arXiv **1605** (2016) 06431.

[216] J. L. Ba, J. R. Kiros and G. E. Hinton, arXiv **1607** (2016) 06450.

[217] R. Tibshirani, J. Royal Stat. Soc. B **58** (1996) 267.

[218] F. Santosa and W. W. Symes, SIAM J. Sci. Stat. Comp. **7** (1986) 1307.

[219] A. E. Hoerl and R. W. Kennard, Technometrics **12** (1970) 55.

[220] A. E. Hoerl and R. W. Kennard, Technometrics **12** (1970) 69.

[221] D. E. Hilt and D. W. Seegrist, *Ridge, a computer program for calculating ridge regression estimates*, Upper Darby, **1977**.

[222] G. E. Hinton et al., arXiv **1207** (2012) 0580.

[223] N. Srivastava et al., Journal of Machine Learning Research **15** (2014) 1929.

[224] L. Wan et al., "Regularization of Neural Networks using DropConnect", *Conference on Machine Learning*, **2013**.

[225] E. J. Bjerrum, arXiv **1703** (2017) 07076.

[226] J. Arus-Pous et al., Journal of Cheminformatics **11** (2019) 71.

[227] K. Hansen et al., J. Phys. Chem. Lett. **6** (2015) 2326.

[228] J. Behler, J. Chem. Phys. **134** (2011) 074106.

[229] H. Huo and M. Rupp, Machine Learning: Science and Technology **3** (2017) 045017.

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

[231] H. L. Morgan, Journal of Chemical Documentation **5** (1965) 107.

[232] D. Rogers and M. Hahn, J. Chem. Inf. Model. **50** (2010) 742.

[233] R. Trudeau, *Introduction to Graph Theory*, Dover Publications, **2013**.

[234] V. I. Minkin, Pure and Applied Chemistry **71** (1999) 1919.

[235] J. Gilmer et al., arXiv **1704** (2017) 01212.

[236] D. Duvenaud et al., arXiv **1509** (2015) 09292.

[237] K. Xu et al., arXiv **1810** (2019) 00826.

[238] K. T. Schütt et al., J. Chem. Phys. **148** (2018) 241722.

[239] S. Batzner et al., Nature Communications **13** (2022).

[240] I. Batatia et al., "MACE: Higher Order Equivariant Message Passing Neural Networks for Fast and Accurate Force Fields", *Conference on Neural Information Processing Systems*, **2022**.

[241] Y. Bengio, A. Courville and P. Vincent, arXiv **1206** (2014) 5538.

[242] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, **1999**.

[243] E. A. Nadaraya, Theory of Probability & Its Applications **9** (1964) 141.

[244] G. S. Watson, Indian Journal of Statistics A **26** (1964) 359.

[245] D. MacKay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, **2003**.

[246] T. Bayes, Philosophical Transactions of the Royal Society of London **53** (1763) 370.

[247] A. P. Bartók et al., Phys. Rev. Lett. **104** (2010) 136403.

[248] M. J. Burn and P. L. A. Popelier, J. Comput. Chem. **43** (2022) 2084.

[249] M. Bauer, M. van der Wilk and C. E. Rasmussen, arXiv **1606** (2017) 04820.

[250] M. P. Deisenroth and J. W. Ng, arXiv **1502** (2015) 02843.

[251] G. Cybenko, Mathematics of Control, Signals and Systems **2** (1989) 303.

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

[253] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, **2006**.

[254] S. Käser et al., Digital Discovery **2** (2023) 28.

[255] P. Reiser et al., Communications Materials **3** (2022) 93.

[256] M. Kulichenko et al., J. Phys. Chem. Lett. **12** (2021) 6227.

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

[258] R. Pascanu, T. Mikolov and Y. Bengio, arXiv **1211** (2013) 5063.

[259] G. B. Goh, N. O. Hodas and A. Vishnu, arXiv **1701** (2017) 04503.

[260] O. T. Unke and M. Meuwly, J. Chem. Theory Comput. **15** (2019) 3678.

[261] F. Scarselli et al., IEEE Transactions on Neural Networks **20** (2009) 61.

[262] L. Wu et al., *Graph Neural Networks: Foundations, Frontiers, and Applications*, Springer, **2022**.

[263] W. L. Hamilton, Synthesis Lectures on AI and ML **14** (2020) 1.

[264] K. Yang et al., arXiv **1904** (2019) 01561.

[265] J. M. Stokes et al., Cell **180** (2020) 688.

[266] J. Gasteiger, J. Groß and S. Günnemann, "Directional Message Passing for Molecular Graphs", *Conference on Learning Representations*, **2020**.

[267] J. Gasteiger et al., "Fast and Uncertainty-Aware Directional Message Passing for Non-Equilibrium Molecules", *Conference on Neural Information Processing Systems*, **2020**.

[268] J. Gasteiger, F. Becker and S. Günnemann,
"GemNet: Universal Directional Graph Neural Networks for Molecules",
Conference on Neural Information Processing Systems, **2021**.

[269] A. M. Schweidtmann et al., Computers & Chemical Engineering **172** (2023) 108202.

[270] W. Pronobis et al., The European Physical Journal B **91** (2018) 178.

[271] R. Ying et al., arXiv **1806** (2019) 08804.

[272] K. Oono and T. Suzuki, arXiv **1905** (2021) 10947.

[273] F. D. Giovanni et al., arXiv **2306** (2024) 03589.

[274] J. Topping et al., arXiv **2111** (2022) 14522.

[275] P. Veličković et al., arXiv **1710** (2018) 10903.

[276] A. Kosmala et al., arXiv **2303** (2023) 04791.

[277] H. Gao and S. Ji, arXiv **1905** (2019) 05178.

[278] V. G. Satorras, E. Hoogeboom and M. Welling, "E(n) Equivariant Graph Neural Networks",
Conference on Machine Learning, **2021**.

[279] K. T. Schütt, O. T. Unke and M. Gastegger, arXiv **2102** (2021) 03150.

[280] R. Magar et al., Machine Learning: Science and Technology **3** (2022) 045015.

[281] V. Zaverkin et al., Phys. Chem. Chem. Phys. **25** (2023) 5383.

[282] L. Kollias et al., J. Am. Chem. Soc. **144** (2022) 11099.

[283] A. Pareja et al., arXiv **1902** (2019) 10191.

[284] P. Shah et al., Computers & Chemical Engineering **194** (2025) 108926.

[285] A. I. Gircha et al., Scientific Reports **13** (2023) 8250.

[286] C. Qu et al., Artificial Intelligence Chemistry **1** (2023) 100025.

[287] C. Qu et al., J. Chem. Phys. **159** (2023) 071101.

[288] L. Bass et al., J. Chem. Theory Comput. **20** (2024) 396.

[289] H. Tu et al., Applied Energy **329** (2023) 120289.

[290] L. Armelao et al., Coordination Chemistry Reviews **254** (2010) 487.

[291] A. de Bettencourt-Dias, Dalton Trans. (22 2007) 2229.

[292] P. Fang et al., Light: Science & Applications **12** (2023) 170.

[293] M. Bottrill, L. Kwok and N. J. Long, Chem. Soc. Rev. **35** (2006) 557.

[294] S. Lacerda and É. Tóth, Chem. Med. Chem. **12** (2017) 883.

[295] R. D. Dicken, A. Motta and T. J. Marks, ACS Catalysis **11** (2021) 2715.

[296] C. F. L. Jr. and R. J. Beaver, Nuclear Applications **4** (1968) 399.

[297] M. Ernzerhof and G. E. Scuseria, J. Chem. Phys. **110** (1999) 5029.

[298] D. Rappoport and F. Furche, J. Chem. Phys. **133** (2010) 134105.

[299] D. Rappoport, J. Chem. Phys. **155** (2021) 124102.

[300] N. Mardirossian and M. Head-Gordon, J. Chem. Phys. **144** (2016) 214110.

[301] E. Caldeweyher, C. Bannwarth and S. Grimme, J. Chem. Phys. **147** (2017) 034112.

[302] E. Caldeweyher et al., J. Chem. Phys. **150** (2019) 154122.

[303] F. L. Hirshfeld, Theoretica Chimica Acta **44** (1977) 129.

[304] M. Müller, A. Hansen and S. Grimme, J. Chem. Phys. **159** (2023) 164108.

[305] R. S. Mulliken, J. Chem. Phys. **23** (1955) 1833.

[306] P.-O. Löwdin, J. Chem. Phys. **18** (1950) 365.

[307] C. Hölzer et al., J. Chem. Inf. Model. **64** (2024) 8909.

[308] A. C. Brown and T. R. Fraser, Transactions of the Royal Society of Edinburgh **25** (1868) 151.

[309] M. Martin-Smith, G. A. Smail and J. B. Stenlake,
Journal of Pharmacy and Pharmacology **19** (1967) 649.

[310] A. Carotenuto et al., Journal of Medicinal Chemistry **49** (2006) 5072.

[311] M. De Vivo et al., Journal of Medicinal Chemistry **59** (2016) 4035.

[312] P. Pracht, F. Bohle and S. Grimme, Phys. Chem. Chem. Phys. **22** (2020) 7169.

[313] S. Grimme, J. Chem. Theory Comput. **15** (2019) 2847.

[314] K. T. Schütt et al., "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions", *Conference on Neural Information Processing Systems*, **2017**.

[315] S. Grimme et al., J. Chem. Phys. **154** (2021) 064103.

[316] *Commandline ENergetic SOrting of Conformer Rotamer Ensembles*,
https://github.com/grimme-lab/censo, **2021**.

[317] M. Friede et al., J. Chem. Phys. **161** (2024) 062501.

[318] S.-C. Li et al., J. Am. Chem. Soc. **146** (2024) 23103.

[319] R. E. Wengert, Commun. ACM **7** (1964) 463.

[320] A. Griewank and A. Walther,
Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation,
Society for Industrial and Applied Mathematics, **2008**.

[321] A. McSloy et al., J. Chem. Phys. **158** (2023) 034801.

[322] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*, CreateSpace, **2009**.

[323] A. Paszke et al., *Automatic differentiation in PyTorch*, **2017**.

[324] Q. Sun, J. Comput. Chem. **36** (2015) 1664.

[325] M. F. Kasim and S. M. Vinko, Phys. Rev. Lett. **127** (2021) 126403.

[326] M. F. Kasim, S. Lehtola and S. M. Vinko, J. Chem. Phys. **156** (2022) 084801.

[327] M. F. Kasim and S. M. Vinko, arXiv **2010** (2020) 01921.

[328] *Light-weight tight-binding framework*, https://github.com/tblite/tblite, **2024**.

[329] J. Gorges et al., Synlett **34** (2022) 1135.

[330] T. Rose et al., Inorg. Chem. **63** (2024) 19364.

[331] M. Müller et al., J. Phys. Chem. A **128** (2024) 10723.

[332] Z. Liu et al., arXiv **2404** (2024) 19756.

[333] S. Jin et al., Light: Science & Applications **11** (2022) 52.

[334] D. A. Gálico, C. M. Santos Calado and M. Murugesu, Chem. Sci. **14** (2023) 5827.

[335] H. S. Mader et al., Current Opinion in Chemical Biology **14** (2010) 582.

[336] H. Dong et al., Chem. Rev. **115** (2015) 10725.

[337] X. Qiu et al., Acc. Chem. Res. **55** (2022) 551.

[338] J. Rocha et al., Chem. Soc. Rev. **40** (2011) 926.

[339] Y. Cui et al., Chem. Rev. **112** (2012) 1126.

[340] F. A. Almeida Paz et al., Chem. Soc. Rev. **41** (2012) 1088.

[341] Y. Yao and K. Nie, "Lanthanides: Homogeneous Catalysis", *Encyclopedia of Inorganic and Bioinorganic Chemistry*, Wiley & Sons, Ltd, **2012**.

[342] F. T. Edelmann, "Homogeneous Catalysis Using Lanthanide Amidinates and Guanidinates", *Molecular Catalysis of Rare-Earth Elements*, Springer, **2010** 109.

[343] J. A. J. Cotruvo, ACS Central Science **5** (2019) 1496.

[344] H. Liu, S. Saha and M. S. Eisen, Coordination Chemistry Reviews **493** (2023) 215284.

[345] A. Stwertka, *A Guide to the Elements*, Oxford University Press, **2002**.

[346] L. I. Vazquez-Salazar et al., J. Chem. Theory Comput. **17** (2021) 4769.

[347] J. Kirkpatrick et al., Science **374** (2021) 1385.

[348] K. Raghavachari et al., Chem. Phys. Lett. **157** (1989) 479.

[349] L. C. Blum and J.-L. Reymond, J. Am. Chem. Soc. **131** (2009) 8732.

[350] G. Montavon et al., New Journal of Physics **15** (2013) 095003.

[351] R. Ramakrishnan et al., J. Chem. Phys. **143** (2015) 084111.

[352] H. Kneiding et al., Digital Discovery **2** (2023) 618.

[353] H. Kneiding, A. Nova and D. Balcells, ChemRxiv (2023) k3tf2.

[354] Cambridge Structural Database, Cambridge Crystallographic Data Centre, https://www.ccdc.cam.ac.uk/. Accessed: 2023-09-18.

[355] M. G. Taylor et al., Nature Communications **14** (2023) 2786.

[356] ORCA – an ab initio, density functional and semiempirical program package, V. 5.0.4, F. Neese, MPI für Kohlenforschung, Mülheim a. d. Ruhr (Germany), **2022**.

[357] B. Helmich-Paris et al., J. Chem. Phys. **155** (2021) 104109.

[358] O. A. Vydrov and T. Van Voorhis, J. Chem. Phys. **133** (2010) 244103.

[359] M. Dolg, H. Stoll and H. Preuss, J. Chem. Phys. **90** (1989) 1730.

[360] M. Dolg et al., Theoretica chimica acta **75** (1989) 173.

[361] X. Cao and M. Dolg, J. Chem. Phys. **115** (2001) 7348.

[362] *Architector 3D chemical structure generation*, https://github.com/lanl/Architector, **2023**.

[363] *Architector-Wrapper package for handling Architector*, https://github.com/grimme-lab/ArchitectorWrapper, **2023**.

[364] *HDF Group*, https://www.hdfgroup.org/, **2023**.

[365] Python Programming Language, V. 3.11.4, Guido van Rossum and Python Software Foundation, https://www.python.org/, **2023**.

[366] M. Vasiliu, K. A. Peterson and D. A. Dixon, J. Phys. Chem. A **124** (2020) 6913.

[367] N. N. Greenwood and A. Earnshaw, *Chemistry of the Elements*, Butterworth-Heinemann, **1997**.

[368] A. F. Holleman, E. Wiberg and N. Wiberg, *Lehrbuch der Anorganischen Chemie*, Walter de Gruyter, **1995**.

[369] C. E. Housecroft and A. G. Sharpe, *Inorg. Chem.* Prentice Hall, **2004**.

[370] W. M. Haynes, *Handbook of Chemistry and Physics*, CRC Press, **2016**.

[371] NIST Atomic Spectra Database Ionization Energies, https://physics.nist.gov/asd. Accessed: 2023-10-23.

[372] N. Andreadi et al., Inorg. Chem. **59** (2020) 13383.

[373] S. A. Ghasemi et al., Phys. Rev. B **92** (2015) 045131.

[374] I. Mayer, Chem. Phys. Lett. **97** (1983) 270.

[375] O. Fizer et al., J. Mol. Model. **24** (2018) 141.

[376] R. F. Bader, Acc. Chem. Res. **18** (1985) 9.

[377] R. F. Bader, Chem. Rev. **91** (1991) 893.

[378] S. P. Perlepes, "Bioinorganic Aspects of Lanthanide(III) Coordination Chemistry: Modelling the Use of Lanthanides(III) as Probes at Calcium(II) Binding Sites", *Cytotoxic, Mutagenic and Carcinogenic Potential of Heavy Metals Related to Human Environment*, Springer, **1997**.

[379] C. W. am Ende et al., ChemBioChem **11** (2010) 1738.

[380] J. A. Mattocks, J. L. Tirsch and J. A. Cotruvo, "Determination of affinities of lanthanide-binding proteins using chelator-buffered titrations", *Rare-Earth Element Biochemistry: Characterization and Applications of Lanthanide-Binding Biomolecules*, Academic Press, **2021**.

[381] D. Joss and D. Häussinger, Prog. Nucl. Mag. Res. Spect. **114-115** (2019) 284.

[382] I. D. Herath et al., Chem. Eur. J. **27** (2021) 13009.

[383] D. Joss, F. Winter and D. Häussinger, Chem. Commun. **56** (2020) 12861.

[384] P. Pyykkö and M. Atsumi, Chem. Eur. J. **15** (2009) 186.

[385] A. C. Brown and T. R. Fraser, Transactions of the Royal Society of Edinburgh **25** (1869) 693.

[386] R. E. Taylor et al., J. Am. Chem. Soc. **125** (2003) 26.

[387] P. L. Gentili, Biomimetics **9** (2024) 121.

[388] H. A. Scheraga, “Calculations of Conformations of Polypeptides”, Advances in Physical Organic Chemistry, Academic Press, **1968**.

[389] K. Munkerup et al., Journal of Saudi Chemical Society **23** (2019) 1206.

[390] M. Heger et al., Phys. Chem. Chem. Phys. **17** (2015) 9899.

[391] A. Lakdawala et al., BMC Chemical Biology **1** (2001) 2.

[392] V. Dragojlovic, ChemTexts **1** (2015) 14.

[393] A. Nivedha et al., J. Comput. Chem. **35** (2014) 526.

[394] D. H. R. Barton, Experientia **6** (1950) 316.

[395] V. Pophristic and L. Goodman, Nature **411** (2001) 565.

[396] F. Nitta and H. Kaneko, Molecular Informatics **40** (2021) 2000123.

[397] I.-J. Chen and N. Foloppe, J. Chem. Inf. Model. **48** (2008) 1773.

[398] A. M. Monzon et al., PLOS Computational Biology **13** (2017) 1.

[399] R. D. Cramer, D. E. Patterson and J. D. Bunce, J. Am. Chem. Soc. **110** (1988) 5959.

[400] K. Monde et al., J. Am. Chem. Soc. **128** (2006) 6000.

[401] M. Stahn et al., Environ. Sci.: Processes Impacts **24** (2022) 2153.

[402] D. W. C. S. V. Mani and R. D. Braddock, Critical Reviews in Environmental Control **21** (1991) 217.

[403] M. Alexander and B. K. Lustigman, Journal of Agricultural and Food Chemistry **14** (1966) 410.

[404] S. P. Jarvis et al., Nature Communications **6** (2015) 8338.

[405] L. Derdour and D. Skliar, Chemical Engineering Science **106** (2014) 275.

[406] H. P. G. Thompson and G. M. Day, Chem. Sci. **5** (2014) 3173.

[407] E. A. Stone et al., ACS Catalysis **11** (2021) 4395.

[408] J. M. Gallagher et al., Chem **10** (2024) 855.

[409] I. Iribarren and C. Trujillo, J. Chem. Inf. Model. **62** (2022) 5568.

[410] T. Rahaman, T. Vasiljevic and L. Ramchandran, Trends in Food Science & Technology **49** (2016) 24.

[411] Y. Han, J. H. Cheng and D. W. Sun, Crit. Rev. Food Sci. Nutr. **59** (2019) 794.

[412] M. Manzoor et al., Food and Bioprocess Technology **17** (2024) 2131.

[413] D. K. Sahoo et al., Journal of Photochemistry and Photobiology A **453** (2024) 115671.

[414] P. C. D. Hawkins, J. Chem. Inf. Model. **57** (2017) 1747.

[415] J.-P. Ebejer, G. M. Morris and C. M. Deane, J. Chem. Inf. Model. **52** (2012) 1146.

[416] A. T. McNutt et al., J. Chem. Inf. Model. **63** (2023) 6598.

[417] G. Janson et al., Nature Communications **14** (2023) 774.

[418] M. J. Vainio and M. S. Johnson, J. Chem. Inf. Model. **47** (2007) 2462.

[419] V. Marinova et al., J. Chem. Inf. Model. **61** (2021) 2263.

[420] J. Zhu et al., International Journal of Molecular Sciences **24** (2023) 6896.

[421] J. V. Diez et al., Machine Learning: Science and Technology **5** (2024) 025010.

[422] R. Jiang et al., J. Comput. Chem. **43** (2022) 1880.

[423] G. Simm, R. Pinsler and J. M. Hernandez-Lobato, "Reinforcement Learning for Molecular Design Guided by Quantum Mechanics", *Conference on Machine Learning*, **2020**.

[424] Z. Shamsi, K. J. Cheng and D. Shukla, J. Phys. Chem. B **122** (2018) 8386.

[425] D. E. Kleiman and D. Shukla, J. Chem. Theory Comput. **18** (2022) 5422.

[426] Y. Wang et al., arXiv **2403** (2024) 14088.

[427] C. Zeni et al., arXiv **2312** (2024) 03687.

[428] J. Fan et al., J. Chem. Inf. Model. **64** (2024) 8414.

[429] D. D. Beusen et al., J. Mol. Struct. (Theochem) **370** (1996) 157.

[430] N. Sauton et al., BMC Bioinformatics **9** (2008) 184.

[431] T. Kalvoda et al., J. Phys. Chem. B **126** (2022) 5949.

[432] A. Smellie et al., J. Comput. Chem. **24** (2003) 10.

[433] N. M. O'Boyle et al., Journal of Cheminformatics **3** (2011) 8.

[434] T. Seidel et al., J. Chem. Inf. Model. **63** (2023) 5549.

[435] L. Chan, G. R. Hutchison and G. M. Morris, Journal of Cheminformatics **11** (2019) 32.

[436] S. Riniker and G. A. Landrum, J. Chem. Inf. Model. **55** (2015) 2562.

[437] P. C. D. Hawkins et al., J. Chem. Inf. Model. **50** (2010) 572.

[438] P. C. D. Hawkins and A. Nicholls, J. Chem. Inf. Model. **52** (2012) 2919.

[439] K. Anggara et al., J. Am. Chem. Soc. **142** (2020) 21420.

[440] O. Sperandio et al., European Journal of Medicinal Chemistry **44** (2009) 1405.

[441] M. A. Miteva, F. Guyon and P. Tuffery, Nucleic Acids Res. **38** (2010) W622.

[442] M. Rosa et al., J. Chem. Theory Comput. **12** (2016) 4385.

[443] K. S. Watts et al., J. Chem. Inf. Model. **50** (2010) 534.

[444] P. Pracht and S. Grimme, Chem. Sci. **12** (2021) 6551.

[445] P. Pracht, C. A. Bauer and S. Grimme, J. Comput. Chem. **38** (2017) 2618.

[446] J.-M. Mewes, *Personal communication*, **2024**.

[447] A. K. Rappé et al., J. Am. Chem. Soc. **114** (1992) 10024.

[448] T. A. Halgren, J. Comput. Chem. **17** (1996) 490.

[449] D. A. Case et al., J. Chem. Inf. Model. **63** (2023) 6183.

[450] O. T. Unke et al., Nature Communications **12** (2021) 7273.

[451] D. Folmsbee and G. Hutchison, Int. J. Quant. Chem. **121** (2021) e26381.

[452] A. S. Christensen et al., J. Chem. Phys. **155** (2021) 204103.

[453] S. Axelrod and R. Gómez-Bombarelli,
Machine Learning: Science and Technology **4** (2023) 035025.

[454] A. Christmann and D.-X. Zhou, Journal of Complexity **37** (2016) 1.

[455] Y. Lei, S.-B. Lin and K. Tang, "Generalization Bounds for Regularized Pairwise Learning",
Conference on Artificial Intelligence, **2018**.

[456] Y. Lei, A. Ledent and M. Kloft, "Sharper Generalization Bounds for Pairwise Learning",
Conference on Neural Information Processing Systems, **2020**.

[457] S. Agarwal and P. Niyogi, Journal of Machine Learning Research **10** (2009) 441.

[458] W. Rejchel, Journal of Machine Learning Research **13** (2012) 1373.

[459] M. Köppel et al., *Pairwise Learning to Rank by Neural Networks Revisited: Reconstruction, Theoretical Analysis and Practical Performance*, Springer, **2020**.

[460] R. Heckel et al., "Approximate Ranking from Pairwise Comparisons",
Conference on Artificial Intelligence and Statistics, **2018**.

[461] I. F. D. Oliveira, N. Ailon and O. Davidov,
Journal of Machine Learning Research **19** (2018) 1.

[462] R. M. Neeser, B. Correia and P. Schwaller, arXiv **2312** (2023) 12737.

[463] M. Tynes et al., J. Chem. Inf. Model. **61** (2021) 3846.

[464] S. J. Wetzel et al., Applied AI Lett. **3** (2022) e78.

[465] S. J. Wetzel, R. G. Melko and I. Tamblyn,
Machine Learning: Science and Technology **3** (2022) 045007.

[466] L. von Rueden et al., IEEE Transactions on Knowledge and Data Engineering **35** (2023) 614.

[467] B. Anderson, T. S. Hy and R. Kondor, "Cormorant: Covariant Molecular Neural Networks",
Conference on Neural Information Processing Systems, **2019**.

[468] A. Musaelian et al., Nature Communications **14** (2023) 579.

[469] C. Chen and S. P. Ong, Nature Computational Science **2** (2022) 718.

[470] B. Deng et al., Nature Machine Intelligence **5** (2023) 1031.

[471] M. Chen et al., Medicinal Research Reviews **44** (2024) 1147.

[472] P. O. Dral, Chem. Commun. **60** (2024) 3240.

[473] P. Veličković, "Message passing all the way up",
International Conference on Learning Representations, **2022**.

[474] C. K. Joshi et al., "On the Expressive Power of Geometric Graph Neural Networks",
International Conference on Machine Learning, **2023**.

[475] G. Kim et al., “Benchmark of Machine Learning Force Fields for Semiconductor Simulations: Datasets, Metrics, and Comparative Analysis”, *Conference on Neural Information Processing Systems*, **2023**.

[476] V. Bihani et al., Digital Discovery **3** (2024) 759.

[477] A. Duval et al., arXiv **2312** (2024) 07511.

[478] I. Batatia et al., arXiv **2205** (2022) 06643.

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

[480] M. M. Bronstein et al., arXiv **2104** (2021) 13478.

[481] S. Grimme et al., Chem. Rev. **116** (2016) 5105.

[482] W. J. Mortier, S. K. Ghosh and S. Shankar, J. Am. Chem. Soc. **108** (1986) 4315.

[483] N. L. Allinger, Y. H. Yuh and J. H. Lii, J. Am. Chem. Soc. **111** (1989) 8551.

[484] E. Caldeweyher et al., Phys. Chem. Chem. Phys. (22 2020) 8499.

[485] *Generally Applicable Atomic-Charge Dependent London Dispersion Correction*, https://github.com/dftd4/dftd4, **2024**.

[486] *Semiempirical Extended Tight-Binding Program Package xtb*, https://github.com/grimme-lab/xtb, **2024**.

[487] *Conformer-Rotamer Ensemble Sampling Tool*, https://github.com/grimme-lab/crest, **2021**.

[488] Python Programming Language, V. 3.11.5, Guido van Rossum and Python Software Foundation, https://www.python.org/, **2023**.

[489] J. Ansel et al., “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation”, *International Conference on Architectural Support for Programming Languages and Operating Systems*, **2024**.

[490] *PyTorch Geometric*, https://github.com/pyg-team/pytorch_geometric, **2024**.

[491] *MACE*, https://github.com/ACEsuit/mace, **2024**.

[492] *GemNet*, https://github.com/TUM-DAML/gemnet_pytorch, **2023**.

[493] A. M. Richard et al., Chemical Research in Toxicology **29** (2016) 1225.

[494] H. Mun, W. Lorpaiboon and J. Ho, J. Phys. Chem. A **128** (2024) 4391.

[495] N. van Staalduinen and C. Bannwarth, ChemRxiv (2024) k40v5.

[496] I. Y. Kanal, J. A. Keith and G. R. Hutchison, Int. J. Quant. Chem. **118** (2018) e25512.

[497] D. I. Sharapa et al., Chem. Phys. Chem. **20** (2019) 92.

[498] M. Marianski et al., J. Chem. Theory Comput. **12** (2016) 6157.

[499] J. Řezáč et al., J. Chem. Theory Comput. **14** (2018) 1254.

[500] S. Ehlert, S. Grimme and A. Hansen, J. Phys. Chem. A **126** (2022) 3521.

[501] D. H. Williams and B. Bardsley, Angew. Chem. Int. Ed. **38** (1999) 1172.

[502] World Health Organization, *Critically Important Antimicrobials for Human Medicine*, World Health Organization, **2019**.

[503] A. D. Becke, J. Chem. Phys. **140** (2014) 18A301.

[504] J. G. Brandenburg et al., J. Chem. Phys. **148** (2018) 064104.

[505] S. Grimme et al., J. Chem. Phys. **143** (2015) 054107.

[506] M. Müller, A. Hansen and S. Grimme, J. Chem. Phys. **158** (2023) 014103.

[507] K. Miyamoto, T. F. I. Miller and F. R. Manby, J. Chem. Theory Comput. **12** (2016) 5811.

[508] J. Witte, J. B. Neaton and M. Head-Gordon, J. Chem. Phys. **146** (2017) 234105.

[509] J. Hostaš and J. Řezáč, J. Chem. Theory Comput. **13** (2017) 3575.

[510] A. Otero-de-la-Roza and G. A. DiLabio, J. Chem. Theory Comput. **13** (2017) 3505.

[511] V. Ásgeirsson, C. A. Bauer and S. Grimme, Chem. Sci. **8** (2017) 4879.

[512] J. Koopman and S. Grimme, ACS Omega **4** (2019) 15120.

[513] J. Koopman and S. Grimme, J. Am. Soc. Mass Spectrom. **32** (2021) 1735.

[514] T. Kamachi and K. Yoshizawa, J. Chem. Inf. Model. **56** (2016) 347.

[515] P. M. Zimmerman, J. Chem. Phys. **138** (2013) 184102.

[516] L. D. Jacobson et al., J. Chem. Theory Comput. **13** (2017) 5780.

[517] S. Grimme, Chem. Eur. J. **18** (2012) 9955.

[518] P. Pulay, WIREs Comput. Mol. Sci. **4** (2014) 169.

[519] M. Bartholomew-Biggs et al., J. Comput. Appl. Math. **124** (2000) 171.

[520] M. Sambridge et al., Geophys. J. Int. **170** (2007) 1.

[521] M. Minkov et al., ACS Photonics **7** (2020) 1729.

[522] S. Colburn and A. Majumdar, Commun. Phys. **4** (2021) 1.

[523] D. Puzzuoli et al., J. Open Source Softw. **8** (2023) 5853.

[524] S. Doerr et al., J. Chem. Theory Comput. **17** (2021) 2355.

[525] S. S. Schoenholz and E. D. Cubuk, "JAX MD",
Conference on Neural Information Processing Systems, **2019**.

[526] M. C. Kaymak et al., J. Chem. Theory Comput. **18** (2022) 5181.

[527] S. Sorella and L. Capriotti, J. Chem. Phys. **133** (2010) 234111.

[528] A. Mahajan et al., J. Chem. Phys. **159** (2023) 184101.

[529] D. R. Hartree, Math. Proc. Cambridge Philos. Soc. **24** (1928) 111.

[530] T. Tamayo-Mendoza et al., ACS Cent. Sci. **4** (2018) 559.

[531] A. S. Abbott et al., J. Phys. Chem. Lett. **12** (2021) 3232.

[532] X. Zhang and G. K.-L. Chan, J. Chem. Phys. **157** (2022) 204801.

[533] J. M. Arrazola et al., arXiv **2111** (2023) 09967.

[534] P. A. M. Casares et al., J. Chem. Phys. **160** (2024) 062501.

[535] C. W. Tan, C. J. Pickard and W. C. Witt, J. Chem. Phys. **158** (2023) 124801.

[536] U. Ekström et al., J. Chem. Theory Comput. **6** (2010) 1971.

[537] R. Bast et al., Phys. Chem. Chem. Phys. **13** (2011) 2627.

[538] L. Li et al., Phys. Rev. Lett. **126** (2021) 036401.

[539] F. Pavošević and S. Hammes-Schiffer, arXiv **2011** (2020) 11690.

[540] J. S. Kottmann, A. Anand and A. Aspuru-Guzik, Chem. Sci. **12** (2021) 3497.

[541] J. A. R. Shea and E. Neuscamman, J. Chem. Phys. **149** (2018) 081101.

[542] H.-J. Liao et al., Phys. Rev. X **9** (2019) 031041.

[543] C. Song, T. J. Martínez and J. B. Neaton, J. Chem. Phys. **155** (2021) 024108.

[544] R. Steiger et al., Future Gener. Comput. Syst. **21** (2005) 1324.

[545] G. Zhou et al., J. Chem. Theory Comput. **16** (2020) 4951.

[546] M. Elstner, J. Phys. Chem. A **111** (2007) 5614.

[547] M. Elstner, Theor. Chem. Acc. **116** (2005) 316.

[548] A. Paszke et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Conference on Neural Information Processing Systems*, **2019**.

[549] N. Fedik et al., J. Chem. Phys. **159** (2023) 110901.

[550] T. Zubatiuk et al., J. Chem. Phys. **154** (2021) 244108.

[551] R. A. Vargas-Hernández et al., J. Chem. Phys. **158** (2023) 104801.

[552] P. O. Dral, O. A. von Lilienfeld and W. Thiel, J. Chem. Theory Comput. **11** (2015) 2120.

[553] G. Zhou et al., Proc. Natl. Acad. Sci. U.S.A. **119** (2022) e2120333119.

[554] C.-P. Chou et al., J. Chem. Theory Comput. **12** (2016) 53.

[555] J. J. Kranz et al., J. Chem. Theory Comput. **14** (2018) 2341.

[556] C. Panosetti et al., J. Chem. Theory Comput. **16** (2020) 2181.

[557] M. Stöhr, L. Medrano Sandonas and A. Tkatchenko, J. Phys. Chem. Lett. **11** (2020) 6835.

[558] L. Komissarov and T. Verstraelen, J. Chem. Inf. Model. **61** (2021) 5931.

[559] S. Raaijmakers et al., J. Phys. Chem. C **126** (2022) 9587.

[560] J.-M. Mewes, A. Hansen and S. Grimme, Angew. Chem., Int. Ed. **60** (2021) 13144.

[561] L. Komissarov et al., J. Chem. Inf. Model. **61** (2021) 3737.

[562] R. Ramakrishnan et al., J. Chem. Theory Comput. **11** (2015) 2087.

[563] K. Atz et al., Phys. Chem. Chem. Phys. **24** (2022) 10775.

[564] L. D. Jacobson et al., J. Chem. Theory Comput. **18** (2022) 2354.

[565] P. Zheng et al., Nat. Commun. **12** (2021) 7022.

[566] H. Li et al., J. Chem. Theory Comput. **14** (2018) 5764.

[567] C. C. J. Roothaan, Rev. Mod. Phys. **23** (1951) 69.

[568] G. G. Hall, Proc. R. Soc. London, Ser. A **205** (1951) 541.

[569] S. Grimme, J. Chem. Theory Comput. **10** (2014) 4497.

[570] S. Grimme, S. Ehrlich and L. Goerigk, J. Comput. Chem. **32** (2011) 1456.

[571] A. D. Becke and E. R. Johnson, J. Chem. Phys. **123** (2005) 154101.

[572] E. R. Johnson and A. D. Becke, J. Chem. Phys. **123** (2005) 024101.

[573] E. R. Johnson and A. D. Becke, J. Chem. Phys. **124** (2006) 174104.

[574] R. S. Mulliken, J. Chim. Phys. **46** (1949) 675.

[575] M. Wolfsberg and L. Helmholz, J. Chem. Phys. **20** (1952) 837.

[576] K. Nishimoto and N. Mataga, Z. Phys. Chem. **12** (1957) 335.

[577] G. Klopman, J. Am. Chem. Soc. **86** (1964) 4550.

[578] K. Ohno, Theoret. Chim. Acta **2** (1964) 219.

[579] N. D. Mermin, Phys. Rev. **137** (1965) A1441.

[580] A. G. Baydin et al., Journal of Machine Learning Research **18** (2018) 1.

[581] B. van Merrienboer et al.,
"Automatic Differentiation in ML: Where We Are and Where We Should Be Going",
Conference on Neural Information Processing Systems, **2018**.

[582] X. Gao et al., J. Chem. Inf. Model. **60** (2020) 3408.

[583] A. Veillard and E. Clementi, Theoret. Chim. Acta **7** (1967) 133.

[584] M. K. MacLeod and T. Shiozaki, J. Chem. Phys. **142** (2015) 051103.

[585] B. Vlaisavljevich and T. Shiozaki, J. Chem. Theory Comput. **12** (2016) 3781.

[586] R. Improta et al., J. Chem. Phys. **125** (2006) 054103.

[587] T. Froitzheim, S. Grimme and J.-M. Mewes, J. Chem. Theory Comput. **18** (2022) 7702.

[588] T. Brown et al., "Language Models Are Few-Shot Learners",
Conference on Neural Information Processing Systems, **2020**.

[589] B. Workshop et al., arXiv **2211** (2023) 05100.

[590] W. Baur and V. Strassen, Theor. Comput. Sci. **22** (1983) 317.

[591] S. Tokui et al., "Chainer: A Next-Generation Open Source Framework for Deep Learning",
Conference on Neural Information Processing Systems, **2015**.

[592] D. Maclaurin,
Modeling, Inference and Optimization with Composable Differentiable Procedures,
PhD thesis: Harvard University, **2016**.

[593] J. Bradbury et al., *JAX: Composable Transformations of Python+NumPy Programs*, **2018**.

[594] R. Frostig, M. J. Johnson and C. Leary,
Compiling Machine Learning Programs via High-Level Tracing, **2018**.

[595] M. Abadi et al., arXiv **1605** (2016) 08695.

[596] R. Al-Rfou et al., arXiv **1605** (2016) 02688.

[597] T. Chen et al., *MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems*, **2015**.

[598] Y. Jia et al., *Caffe: Convolutional Architecture for Fast Feature Embedding*, **2014**.

[599] C. Bischof et al., Sci. Program. **1** (1992) 717832.

[600] L. Hascoet and V. Pascual, ACM Trans. Math. Softw. **39** (2013) 1.

[601] R. Giering and T. Kaminski, ACM Trans. Math. Softw. **24** (1998) 437.

[602] C. Bischof, B. Lang and A. Vehreschild, PAMM **2** (2003) 50.

[603] A. Griewank, D. Juedes and J. Srinivasan, ACM Trans. Math. Software **22** (1996).

[604] C. Bendtsen and O. Stauning,
FADBAD, a Flexible C++ Package for Automatic Differentiation, Technical Report, **1996**.

[605] C. Bischof and H. Bücker, "Computing Derivatives of Computer Programs",
Modern Methods and Algorithms of Quantum Chemistry,
John von Neumann Institute for Computing, **2000**.

[606] F. Pedregosa et al., Journal of Machine Learning Research **12** (2011) 2825.

[607] *Fully Differentiable Approach to Semiempirical Extended Tight Binding*,
https://dxtb.readthedocs.io/, **2024**.

[608] A. H. Larsen et al., J. Phys. Condens. Matter **29** (2017) 273002.

[609] S. Colvin, *pydantic - Data validation using Python type hints*,
https://github.com/pydantic/pydantic, **2023**.

[610] M. B. Giles, "Collected Matrix Derivative Results for Forward and Reverse Mode Algorithmic Differentiation", *Advances in Automatic Differentiation*, Springer, **2008**.

[611] M. Giles, *An Extended Collection of Matrix Derivative Results for Forward and Reverse Mode Automatic Differentiation*, Technical Report, **2008**.

[612] J. Johnson, M. Douze and H. Jégou, *Billion-Scale Similarity Search with GPUs*, **2017**.

[613] M. Fey et al., *SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels*, **2018**.

[614] J. Feydy et al., "Fast Geometric Learning with Symbolic Matrices",
Conference on Neural Information Processing Systems, **2020**.

[615] K. M. Jatavallabhula et al., arXiv **1911** (2019) 05063.

[616] N. Ravi et al., arXiv **2007** (2020) 08501.

[617] S. Obara and A. Saika, J. Chem. Phys. **84** (1986) 3963.

[618] S. Obara and A. Saika, J. Chem. Phys. **89** (1988) 1540.

[619] F. Neese, J. Comput. Chem. **44** (2023) 381.

[620] L. E. McMurchie and E. R. Davidson, J. Comput. Phys. **26** (1978) 218.

[621] S. G. Balasubramani et al., J. Chem. Phys. **152** (2020) 184107.

[622] Z. Wu et al., Chem. Sci. **9** (2018) 513.

[623] Q. Sun et al., WIREs Comput. Mol. Sci. **8** (2018) e1340.

[624] M. Korth and S. Grimme, J. Chem. Theory Comput. **5** (2009) 993.

[625] D. G. Anderson, J. ACM **12** (1965) 547.

[626] M. Seeger et al., arXiv **1710** (2019) 08717.

[627] W. Jin et al., ACS Photonics **7** (2020) 2350.

[628] N. Yoshikawa and M. Sumita, J. Phys. Chem. A **126** (2022) 8487.

[629] M. Blondel et al., "Efficient and Modular Implicit Differentiation", *Conference on Neural Information Processing Systems*, **2022**.

[630] J. Ren et al., Journal of Machine Learning Research **24** (2023) 1.

[631] L. Pineda et al., "Theseus: A Library for Differentiable Nonlinear Optimization", *Conference on Neural Information Processing Systems*, **2022**.

[632] J. Lorraine, P. Vicol and D. Duvenaud, "Optimizing Millions of Hyperparameters by Implicit Differentiation", *Conference on Artificial Intelligence and Statistics*, **2020**.

[633] B. Amos and J. Z. Kolter, "OptNet: Differentiable Optimization as a Layer in Neural Networks", *Conference on Machine Learning*, **2017**.

[634] S. Bai, J. Z. Kolter and V. Koltun, "Deep Equilibrium Models", *Conference on Neural Information Processing Systems*, **2019**.

[635] H. He and R. Zou, *Functorch: JAX-like composable function transforms for PyTorch*, https://github.com/pytorch/functorch, **2021**.

[636] E. Anderson et al., *LAPACK Users' Guide*, Society for Industrial and Applied Mathematics, **1999**.

[637] A. Najibi and L. Goerigk, J. Comput. Chem. **41** (2020) 2562.

[638] F. Weigend, F. Furche and R. Ahlrichs, J. Chem. Phys. **119** (2003) 12753.

[639] J. Hu, N. Whiting and P. Bhattacharya, J. Phys. Chem. C **122** (2018) 10575.

[640] M. S. Bin-Alam et al., Nano Lett. **21** (2021) 51.

[641] F. A. Santos et al., Photonics **10** (2023) 545.

[642] G. Placzek, Z. Phys. **70** (1931) 84.

[643] D. J. Swanton, G. B. Bacskay and N. S. Hush, Chem. Phys. **83** (1984) 69.

[644] D. P. O'Neill, M. Kállay and J. Gauss, Mol. Phys. **105** (2007) 2447.

[645] P. Pracht, D. F. Grant and S. Grimme, J. Chem. Theory Comput. **16** (2020) 7044.

[646] M. Fanti, G. Orlandi and F. Zerbetto, J. Phys. B **29** (1996) 5065.

[647] M. D. Halls and H. B. Schlegel, J. Chem. Phys. **111** (1999) 8819.

[648] S. Kaminski et al., J. Phys. Chem. A **116** (2012) 9131.

[649] *Fully Differentiable Approach to Semiempirical Extended Tight Binding*, https://github.com/grimme-lab/dxtb, **2024**.

[650] *Reimplementation of the DFT-D3 program*, https://github.com/dftd3/simple-dftd3, **2024**.

[651] *PyTorch Autodiff Multicharge (Classical Charge Models).* https://github.com/tad-mctc/tad-multicharge, **2024**.

[652] *Torch autodiff DFT-D3 implementation*, https://github.com/dftd3/tad-dftd3, **2024**.

[653] *Torch autodiff DFT-D4 implementation*, https://github.com/dftd4/tad-dftd4, **2024**.

List of Figures

List of Tables

Acknowledgements

This thesis would not have been possible without the support, collaboration, and inspiration of many wonderful people, to whom I would like to express my sincere gratitude.

First and foremost, I thank my committee, especially Prof. Stefan Grimme and Prof. Thomas Bredow, for their guidance and insightful discussions. Particularly, I thank Stefan for valuable lessons on research prioritization, leadership, and motivation. Furthermore, I sincerely appreciated the fruitful collaborations throughout this thesis: At Fraunhofer SCAI, special thanks to Dr. Jan Hamaekers for project guidance combined with his approachable and positive manner. Many thanks also to Rick Oerder for enjoyable teamwork, as well as to Gregor Maier and Dr. Astrid Maaß. Moreover, I gratefully acknowledge the collaboration with Merck KGaA, especially Dr. Martin Fitzner for continuous scientific exchange, and Dr. Jan Gerit Brandenburg for interesting discussions and his inspiring personality. Special thanks go to scientific advisors, Prof. Christoph Bannwarth, Dr. Jan-Michael Mewes, Dr. Markus Bursch and Dr. Sebastian Ehlert for their invaluable scientific input and guidance. On this note, I warmly thank those who helped proofread this thesis: Dr. Jan-Michael Mewes, Rick Oerder, Dr. Gerrit Bickendorf, Dr. Markus Bursch, and Dr. Sebastian Ehlert. Special thanks to Dr. Jan-Michael Mewes for insights beyond chemistry, our regular Satort evenings and demonstrating that getting older certainly doesn't mean becoming boring. I warmly thank Dr. Sebastian Ehlert, whose support left a lasting impression on me and whose inspiring attitude has made him a true role model for me. Heartfelt thanks to my colleagues at the MCTC institute, especially to Johannes Gorges for great coffee breaks, fantastic pub-quiz performances, and honest conversations, Dr. Sebastian Spicher for updates on chemistry and road biking, as well as Marcel Müller for great discussions about skiing trips. Thanks also to my office neighbors Dr. Jeroen Koopman and Dr. Hagen Neugebauer, and to Dr. Joachim Laun for memorable rooftop evenings in summer. Also a heartily shout-out to the group's newcomers, whom I have not yet had the opportunity to get to know better. For interactions across working groups and cheerful darts sessions, special thanks to Paul Zaby and Dr. Jan Blasius. I am very happy and grateful that during my time at the MCTC I met so many people I could learn from. A huge thanks to our group's staff, who keep everything running smoothly behind the scenes: especially Claudia Kronz for impressively managing all organizational aspects (Claudia, if you're reading this: thank you very much, you hold everything together!), Jens Mekelburger for always trying to be helpful, and Dr. Andreas Hansen, whose tennis conversations sparked my initial interest in joining the group.

Last but certainly not least, I deeply thank my family and friends for their unconditional support, patience, and encouragement throughout this endeavor. I also thank the 1. FC Köln for showing me that some struggles surpass by far those encountered during a PhD, Paul Kalkbrenner and Hans Zimmer for providing the soundtrack of this journey, and the Tibetan Imbiss that without their culinary support, this thesis truly would not have been possible (Bonn-based scientists can relate to).