Machine learning for molecular geometry optimization and 3D structure generation

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Bioinformatics

by

Modee Rohit Laxman
20172153
rohit.m@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

We hereby certify that the matter embodied in this thesis entitled **Machine learning for molecular geometry optimization and 3D structure generation** has been carried out by Modee Rohit Laxman at the Center for Computational Natural Sciences and Bioinformatics, International Institute of Information Technology, Hyderabad, India under my supervision, and that it will not be submitted elsewhere for the award of any other degree.

Date

Adviser: Prof. U. Deva Priyakuamr

This thesis is dedicated to my Family.

For their endless love, support and encouragement.

Acknowledgments

Completing a Ph.D. journey is a wonderful achievement, and it would not have been possible without the encouragement, support, and affection of many people. During my Ph.D., I have realized that the most important components for research are the 3Ps, i.e., perseverance, patience, and passion. The journey of PhD has taught me these 3Ps and much more. I thank Prof. U. Deva. Priyakumar, my advisor and head of the Center for Computational Natural Sciences and Bioinformatics (CCNSB) and I-Hub Data IIIT-H, for his invaluable support and guidance throughout my journey. He enhanced my research aptitude and channeled my efforts in the correct direction by asking the right questions.

I thank the CCNSB at IIIT-H for supporting me in various forms throughout my research. I thank Dr. Abhijit Mitra and Dr. Nita Parekh for making me part of the metabolomics project at IIIT-H; without them, I would not have come to IIIT. I thank the faculty members of the center, Prof. Abhijit Mitra, Dr. Nita Parekh, Dr. Prabhakar Bhimalapuram, Prof. Gopalakrishnan Bulusu, and Dr. Vinod P.K. for their valuable suggestions, advice, encouragement, and learning's that helped me enhance my knowledge across various fields. I thank TCS for partially funding my Ph.D. I am grateful to Dr. Semparithi Aravindan for his help with the high-performance computing facility. I thank the R&D office staff, Mr. BalSantosh, Mrs. Sirisha, Mrs. Pushpalatha, Mr. Y kishore, Mrs. Nagamani and Mrs. Prathima Mandapati, for their help. I thank Mrs. Indhu for her help during uploading the paper for publication. I am grateful to Umesh and his team for keeping the workspace clean and organized.

On such a long journey toward a doctorate, my friends and colleagues have played a significant role and were always there in times of need. I want to thank my group personnel at IIIT-H, Dr. Shampa, Dr. Navneet, Dr. Tanashree, Dr. Siladitya, Shruthi, Pradeep, Vishal, Dr. Rakesh, and CND students (a long list...) for their lively discussion and group meetings, which always inspired me to explore new ideas during my research work. I thank Siddhartha, Sarvesh, Sheena (NCL-Pune), Ashwini (NCL-Pune), and Dr. Kavita Joshi (NCL-Pune) for being the co-authors in the publications.

I thank all my friends and labmates, Shweta, Pradeep, Bhanu, Aniket, Cyrin, Kuntal, Arvind, Sanchari, Tanashree, Vishal, Vijay, Manisri, Nishtha, Broto, Suba, Antrip, Srilakshmi, Akansha, Rami, for creating a healthy work environment bustling with positive energy. I want to thank Shweta for being the bestest of my friends, for the delicious food she cooked, and

for her constant support. I thank Pradeep, Kuntal, and Bhanu for being my "Sadda Adda." Without them, life during Covid would have been monotonous and difficult. I thank Broto for encouraging me to apply for Ph.D. in Devalab. I don't have words to thank all my friends but THANK YOU ALL for being a part of my life and Ph.D. journey.

I thank my mother and aunt for their constant support, love, care, encouragement, and guidance; without them, this journey would not have been possible. I thank Dr. Elizabeth and Vidhya aunty for their support.

Abstract

Artificial intelligence (AI) has infiltrated all fields of science, from high-energy particle physics to biology to computational chemistry. In the last couple of decades, there has been tremendous advancement in machine learning (ML) applications in computational chemistry. Deep learning (DL) has achieved some success in the automation of feature design, physico-chemical property prediction, accelerated chemical space search, and the design of new drug-like molecules. Much work is still needed in terms of property prediction of inorganic molecules, along with the search and design of new molecules and material design with desired properties.

This research aims to develop machine learning methods for 3D structure generation and molecular geometry optimization. Use of neural network potential (NNPs) can accelerate the process of 3D structure generation and molecular geometry optimization. Various neural network potentials (NNPs) have been reported in the literature to be as fast as force fields and as accurate as DFT. There has been a lack of standard comparative evaluation of these NNPs, which motivated us to do a benchmark study on NNPs. In this benchmark study, we evaluate and compare four NNPs, i.e., ANI, PhysNet, SchNet, and BAND-NN, for their accuracy in energy prediction, transferability to larger molecules, ability to produce accurate PES, and applicability in geometry optimization. In the context of 3D structure generation (Molecules and material design), there are two major components: search algorithm and property predictor. We need a fast and accurate method to predict the energy of the given system to accelerate the search in conformational space. For this, we developed a model known as DART, which predicts the energy of Gallium clusters using a Topological Atomistic Descriptor (TAD). TAD is a very simple and elegant descriptor that tries to encode structural information by dividing the connectivity information using distance cutoffs. We show the DART models ability to predict the energies of Gallium clusters accurately. For the second component, i.e., the search algorithm, we developed an RL-based model, MeGen, to generate 3D low-energy isomers of Gallium clusters, which uses DART as a reward function. Here we showed that MeGen is significantly more efficient than the conventional workflow for generating ground-state geometries as well as low-lying isomers in terms of time and computational resources. Following a similar train of thought, we developed the MolOpt model. This multi-agent RL-based search algorithm can perform molecular geometry optimization (MGO) by searching for the low-energy structure on the potential energy surface. We show that MolOpt trained on ethane and butane can be

used to optimize larger alkanes up to octane. We compare our model with other optimizers and show that MolOpt outperforms the MDMin optimizer and performs similarly to the FIRE optimizer. We further developed an improved version of MolOpt known as MolOpt2. We have made algorithmic changes in MolOpt2, and MolOpt2 is trained on a diverse set of molecules. Hence, due to algorithmic changes, our new model (MolOpt2) can perform MGO on molecules containing elements CHNO and having a size of up to nine heavy atoms. Similar to MolOpt, we compare MolOpt2 with other optimizers and show that MolOpt2 outperforms the MolOpt, MDMin optimizer and performs similarly to the FIRE optimizer.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

If we were to name the most powerful assumption of all, which leads one on and on in an attempt to understand life, it is that all things are made of atoms, and that everything that living things do can be understood in terms of the jigglings and wigglings of atoms.

-Richard Phillips Feynman

In 1929, Dirac remarked that all the necessary information to depict chemical processes and phenomena is included in the Schrödinger equation (SE).[1] However, the SE cannot be solved for more than one electron system. Nevertheless, various computational and numerical approximations have been devised for the simplest systems to extract significant knowledge about a chemical process and/or a system. The meaning of "accuracy" may change depending on the observable interest. For all practical purposes, a system containing several thousand atoms would be considered "large" to be studied using DFT. It would be impractical even to do a single-point energy calculation of such a large system. In reality, system size where the number of atoms is less than a thousand atoms is computationally feasible when considering high-accuracy or ground state structures or *ab-Initio* methods. Quantum mechanical (QM) calculations, although very accurate, are computationally expensive and cannot be used for larger systems with tens of thousands of atoms.[2] Hence, the molecular mechanics force field (MM-FF) has been the method of choice when it comes to modeling biomolecular systems and processes. In MD simulations, forces are calculated using a molecular mechanics force field model. These FFs have a mathematical form. For example, generally, FFs combine bonded and non-bonded terms to capture interatomic interactions. Essentially FFs are parametrized for specific experimental measurements using QM calculations and are inherently approximate.[3, 4, 5]

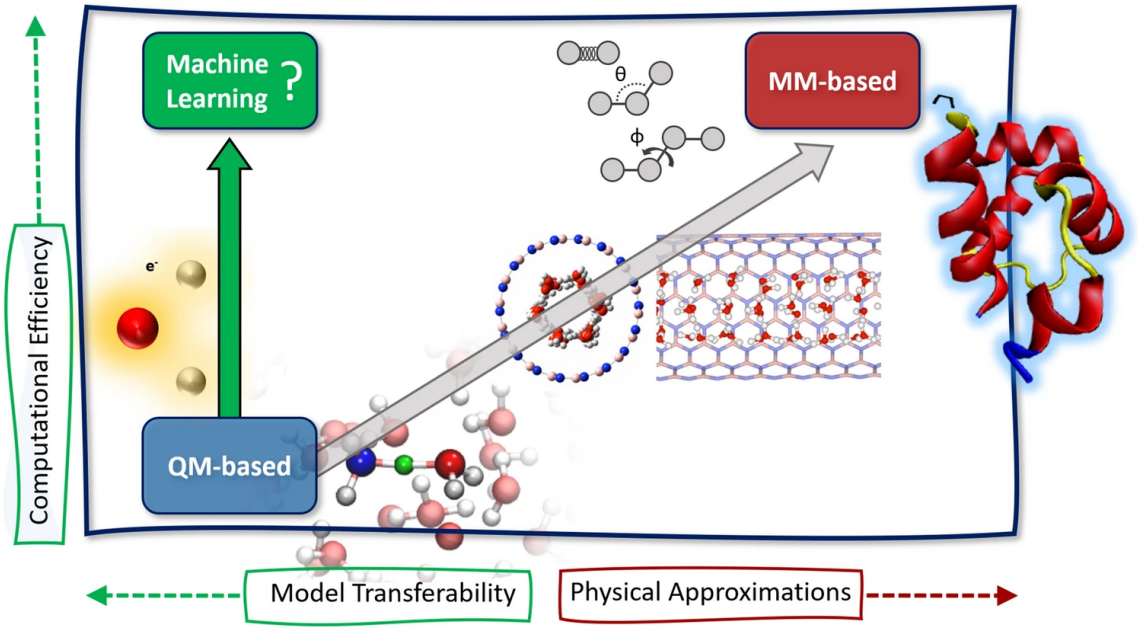


Figure 1.1: Atomistic simulation techniques can be classified into two categories: 1. Electronic structure-based QM methods and 2. predefined functional forms-based molecular mechanics (MM) methods. QM-based simulations are restricted to smaller systems as they are computationally more expensive. At the same time, MM-based methods can be efficiently used for larger systems but rely on many approximations. The objective of NNPs trained on QM data is to decrease the computational cost of QM methods while maintaining their accuracy and transferability. (T. Morawietz et al. [6] 2021)

General empirical FFs are employed in large-scale atomistic simulations with significantly decreased computational costs (Figure 1.1). Additionally, accurate FFs have been pursued for decades to improve MD simulations' accuracy and sampling efficiency. Nevertheless, as these FFs are empirically parametrized, they work only for near-equilibrium structures. FFs are also not as accurate as QM methods, and they lack the ability to model the breaking and formation of chemical bonds. Thus, such potentials are usually unsuitable for transition states and chemical reaction studies. There has been much progress in overcoming the drawbacks mentioned above, such as FFs parameters can be adjusted for specific systems to achieve within fractions of 1 kcal/mol accuracy. Particularly potentials developed for metals, bond-order-based (reactive) potentials, and reactive FFs for particular systems have become well known. Nevertheless, it still makes sense to generalize these potentials to all the above drawbacks instead of developing different potentials for different problems.

In recent years, ML methods have been used to circumvent the problem of solving Schrödinger equation (SE) altogether. ML methods learn the high dimensional function (HDF), f by training the computer on large amounts of precomputed data (properties like energy, force, etc). The ability to learn the HDF (f), which can then be used to map function $f(Z_i, r_i) \rightarrow E$, which gives the energy of the system (E). Hence only based on the nuclear charges and coordinates of the system one can predict its physicochemical properties without solving SE. The HDF is

similar to PES such that when given a set of atomic charges and coordinates of a molecule, it can predict the energy value.[2]

High dimensional function (HDF) f can be learned using supervised ML algorithms to predict the output value for a given input. One such algorithm class is artificial neural networks (ANN). Fully connected feed-forward neural networks (FFNN) are a class of ANN that are great function approximators. FFNN can project the input onto a latent space that relates to the output. The latent space is learned successively via several layers and is generally highly nonlinear. Hence, FFNNs are suited for learning $f : \{Z_i, r_i\} \longrightarrow E$, which approximates PES. An ideal neural network potential (NNP) should have accuracy comparable to QM methods and as fast as FFs. Developed NNP should also be differentiable and transferable to chemical environments other than those it is trained on. NNP should generalize well to different problems, such as bond-breaking/bond-formation problems ("reactive PES"). There has been much progress in developing NNPs that fulfill some of these requirements. NNPs that fulfill all the above properties do not exist yet.

Lately, large amounts of data have been generated due to rapid advances in computational power coupled with large-scale experiments using density functional theory (DFT) and coupled-cluster singles and doubles (CCSD) methods.[7, 8] Currently, the major drawback is that classical methods lack the ability to use large amounts of data. It is well suited to apply ML methods to develop potentials as they are robust, flexible, and can efficiently use a large amount of available data. The general idea is to encode the 3D geometries of the molecule into a descriptor and feed the descriptor to a FFNN, which will predict physicochemical properties.

In the last decade, there have been a host of neural network potentials developed with the aim to predict physicochemical properties of the molecules given their 3D geometries. Neural networks have been used to approximate PES for small molecules with the idea of many-body expansion. While accurate, these methods scale poorly due to many body expansions requiring many individual NN for each term. Many-body expansion problem led to the rise of high dimensional atomic NN (HDNN). Atomic HDNN allows one to utilize the same size NN for different-size systems by defining molecule energy as the sum of atomic energy, $E_{tot} = \sum_{i=1}^{N} E_i$, where E_i is the energy contribution of an atom i of the molecule with a total number of N atoms. HDNNs are also known as Neural network potentials.

In this work, we have focused on developing machine learning methods for 3D structure generation and molecular geometry optimization. Thesis is organised as follows -

chapter 2 gives a brief overview of the fundamental principles of quantum mechanics and classical mechanics. chapter 2 also includes an introduction to machine learning methods and algorithms, thereby setting the stage for a deeper dive into the ML techniques developed in this thesis. The development of various NNPs and the lack of standard comparative evaluation of these NNPs motivated us to do a benchmark study on NNPs. Hence, in Chapter 3, titled *"Benchmark study on Deep Neural Network Potentials for Small Organic Molecules"* we evaluate

and compare four NNPs, i.e., ANI, PhysNet, SchNet, and BAND-NN, for their accuracy in energy prediction, transferability to larger molecules, ability to produce accurate PES, and applicability in geometry optimization. These models were originally trained and tested on different data sets, which makes their applicability and comparison difficult. Chapter 3 provides a standard comparative evaluation of these models by training and testing them on the same data sets.

Now with the knowledge that NNPs can predict the physicochemical properties of the molecules given their 3D geometries, we wanted to address the problem of 3D structure generation (molecules and materials design). Molecules and material design is an optimization problem where one needs to search the chemical and conformation space for the molecule and materials with desired properties. Molecules and material design have two major components: search algorithm and property predictor. The search algorithm searches/generates new 3D structures. The property predictor aims to predict the physicochemical properties, thereby guiding or creating a bias in the search algorithm based on the desired range of physicochemical properties. For this effect, we developed a property predictor known as DART, which we describe in Chapter 4. We developed a reinforcement learning (RL) based search algorithm, MeGen, which uses the DART model to generate low-energy Gallium clusters described in Chapter 5.

Further, with our combined knowledge of ML methods for physicochemical property prediction as well as RL-based search algorithms guided by property predictor we developed an RL based molecular geometry optimizer (MGO) which we call MolOpt. We have described MolOpt in chapter 6 which serves as proof of concept for the potential of multi-agent reinforcement learning (MARL) in MGO of alkanes. In chapter 7 we describe MolOpt2 an extension of MolOpt. MolOpt2 can perform MGO on molecules containing elements CHNO and has imporved performance as compared to MolOpt. In chapter 8, we summarize and discuss the findings from this thesis.

Chapter 2

Methods

In computational chemistry, the molecular system is modeled using principles of quantum mechanics and classical mechanics, which aids in understanding the energetics and dynamic properties of the molecular system. Computing a molecular system's energetic and dynamic properties involves solving numerous mathematically complex equations. Recently, ML algorithms have gained popularity in computational chemistry as they circumvent the problem of solving Schrödinger equation (SE) altogether. ML methods learn the high dimensional function (HDF), f, by training the computer on large amounts of precomputed data (properties like energy, force, etc). This chapter covers the fundamental principles of quantum mechanics and classical mechanics. Additionally, it introduces machine learning methods and algorithms, setting the stage for a deeper dive into the ML techniques developed in this thesis.

2.1 Quantum mechanics

We briefly discussed molecular dynamic simulations using molecular mechanics in the previous section. MM-based classical force fields (FF) offer a computationally efficient alternative to QM-based methods, allowing for the modeling of larger systems and high-throughput exercises, albeit with a trade-off in accuracy. Nonetheless, these MM-based methods cannot accurately capture the electronic rearrangements in the system. In computational chemistry, QM-based methods are used to calculate the electronic structure of molecules, which is crucial for predicting their chemical properties and reactivity.[9, 10, 11, 12, 13, 14]

The central equation in QM is the time-independent Schrödinger equation, which describes the behavior of a quantum system in terms of its wavefunction Ψ:

$$\hat{H}\Psi = E\Psi \tag{2.1}$$

Where $\hat{H}$ is the Hamiltonian Operator, and E is the energy of the system. $\hat{H}$ consists of the kinetic energy terms and the potential energy terms[15]:

$$\hat{H} = T_N(R) + V_{NN}(R) + V_{eN}(e, R) + T_e(r) + V_{ee}(r) \tag{2.2}$$

In Equation 2.2, $T_N(R)$ and $T_e(r)$ are the kinetic energy components of nuclei and electron respectively. $V_{NN}(R)$ and $V_{ee}(r)$ are the nuclear-nuclear and electron electron repulsion terms. $V_{eN}(r, R)$ is the nuclei-electron attraction term.

SE can only be solved exactly for one-electron systems. Analytical solutions are impossible in biological systems with many atoms or chemical systems having a single atom with more than one electron due to the non-separability of variables describing the interactions between electrons. Consequently, approximations have to be made to solve the Schrödinger equation. Two basic approximations are the Born-Oppenheimer (BO) approximation and the Hartree-Fock[16] approximation.

2.1.1 Born-Oppenheimer (BO) approximation

The Born-Oppenheimer (BO) approximation is a fundamental concept in QM that decouples the motion of atomic nuclei from that of electrons in molecules. It assumes that the motion of atomic nuclei is much slower compared to that of electrons, allowing the electron configuration to adjust instantaneously to any change in nuclear positions. BO approximation allows the separation of the Schrödinger equation into electronic and nuclear components. By treating nuclei as fixed in position, one can neglect the $T_N(R)$ term in Equation 2.2 while solving for the electronic wavefunction and energy. While BO approximation simplifies the computational complexity of the system, additional approximations concerning electron-electron repulsions are necessary before it can be applied to systems consisting of many atoms.

2.1.2 Hartree-Fock (HF) approximation

For a given set of nuclear coordinates, the computation of the electronic Schrödinger Equation (SE) is computationally expensive. The Hartree-Fock (HF) approximation simplifies this process by averaging the potential exerted by all electrons around a single electron, reducing the complexity of estimating electron-electron repulsion. The HF method seeks one-electron wave functions that minimize the total energy of the system using the self-consistent field (SCF) method. The overall wave function is represented as a Slater determinant of these one-electron wave functions, ensuring that it is anti-symmetric in nature due to the Pauli exclusion principle. The HF method serves as a foundational approach, leading to semi-empirical and more accurate *ab-initio* methods. Semi-empirical methods, such as AM1[17], MNDO[18], and PM3[19], utilize minimal basis sets and employ various approximations to electron integrals. This approach significantly enhances computational efficiency, often by a factor of 10^2 to 10^3, compared to typical implementations of *ab-initio* quantum mechanics (QM) and density functional theory

(DFT) methods.[20] These methods also use empirical parameters obtained from experimental data to improve their accuracy. DFT[21] is an alternative method that models many-body electron correlation using electron density rather than orbitals, offering better results compared to HF methods at a similar computational cost.

2.1.3 Density functional theory (DFT)

Density Functional Theory (DFT)[21] is a QM-based method used to describe the electronic structure of atoms and molecules. The idea is that the electronic density $\rho(\mathbf{r})$ contains all the information that is needed to determine the ground state (GS) energy and other properties of a system. Unlike Hartree-Fock (HF) method, which focuses on electron orbitals, DFT considers the electron density $\rho(\mathbf{r})$ as the fundamental variable.

In DFT, the total energy E of a system is expressed as a functional of the electron density $\rho(\mathbf{r})$:

$$E[\rho] = T_s[\rho] + V_{\text{ext}}[\rho] + J[\rho] + E_{\text{xc}}[\rho] \tag{2.3}$$

Here, $T_s[\rho]$ is the kinetic energy of a non-interacting electron gas with density ρ, $V_{\text{ext}}[\rho]$ is the external potential energy, $J[\rho]$ is the classical electrostatic interaction energy, and $E_{\text{xc}}[\rho]$ is the exchange-correlation energy, accounting for quantum-mechanical exchange and correlation effects. The key to the success of DFT lies in the exchange-correlation functional $E_{\text{xc}}[\rho]$. Even the relatively simple approximation to the exchange-correlation functional can give favorable results. Common approximations include the local density approximation (LDA), also known as local spin density approximation (LSDA). LDA is based on the assumption that the exchange-correlation energy at a given point in space depends only on the electron density at that point. Mathematically, this is expressed as:

$$E_{\text{xc}}^{\text{LDA}}[\rho] = \int \epsilon_{\text{xc}}^{\text{LDA}}(\rho(\mathbf{r}))\rho(\mathbf{r})\, d\mathbf{r} \tag{2.4}$$

Here, $\epsilon_{\text{xc}}^{\text{LDA}}(\rho(\mathbf{r}))$ is the exchange-correlation energy per particle at point $\mathbf{r}$ in the electron density. Alongside the Local Density Approximation (LDA), there are several other approximations specific to DFT, including the Generalized Gradient Approximation (GGA), meta-GGA, hybrid functionals, and semi-empirical corrections. GGA[22] improves upon the LDA by incorporating information about the gradient of the electron density, which helps GGA capture some non-local effects that LDA misses, leading to more accurate predictions of molecular properties. Popular GGA functionals include PBE,[22] BLYP,[23, 24, 25] OLYP, and BP86. Meta-GGA[26] goes a step further by also considering the second derivative of the density or the kinetic energy density in addition to the electron density and its gradient. Meta-GGA functionals include M06-L,[27], TPSS,[26] revTPSS,[28] and SCAN. Hybrid functionals combine a fraction of the exact Hartree-Fock exchange with DFT exchange-correlation functionals. By including some

exact exchange, hybrid functionals can provide more accurate descriptions of both structural and energetic properties of molecules and materials compared to pure DFT functionals. Semi-empirical corrections are empirical adjustments made to DFT calculations to improve their accuracy. These parameters are often based on fitting to experimental data. Semi-empirical methods, such as AM1, MNDO, and PM3, can significantly improve the accuracy of DFT calculations while maintaining computational efficiency. Due to its balance between accuracy and computational efficiency, DFT has emerged as one of the most extensively employed methods in computational chemistry.[11, 12, 13, 14]

2.1.4 Basis set

In quantum chemistry, a basis set forms the foundation for describing molecular orbitals, which are essential for understanding the electronic structure of molecules.[29, 30, 31] These sets consist of one-particle functions that serve as building blocks for representing molecular orbitals. Typically, molecular orbitals are expressed as linear combinations of atomic orbitals, where the atomic orbitals can be either Slater-type orbitals (STO) or Gaussian-type orbitals (GTO). GTO are more commonly used in practice due to their computational efficiency. The choice of basis set is crucial, as it directly impacts the accuracy of quantum chemical calculations.

There are several types of basis sets used in quantum chemistry, each with its own characteristics and level of accuracy:

1. **Minimal Basis Sets:** Minimal basis sets, such as the minimal STO-3G (Slater-type orbital) basis, are simple and computationally efficient. They consist of a minimal number of basis functions per atom and are useful for quick calculations and preliminary studies.

2. **Double- and Triple- ζ Basis Sets:** Double- and triple- ζ basis sets, such as 6-31G(d) and 6-311+G(2d,2p), respectively, include additional basis functions to better describe the electron density around each atom. These basis sets provide more accurate results compared to minimal basis sets but are more computationally demanding.

3. **Polarization and Diffuse Basis Sets:** Polarization functions, denoted as 'p' or 'd', are added to basis sets to account for the electron density polarization that occurs in molecules. Diffuse functions, denoted as '++', are used to describe electrons that are further away from the nucleus than those described by standard basis functions. These basis sets are particularly handy for describing molecules with large dipole moments or with delocalized electrons.

4. **Correlation Consistent Basis Sets:** Correlation consistent basis sets, such as the cc-pVXZ series (where X is D, T, Q, etc.), are designed to accurately describe electron correlation effects. These basis sets include multiple levels of polarization and diffuse

functions and are used for high-accuracy calculations, especially in post-Hartree-Fock methods.

The choice of basis set is a critical consideration in quantum mechanical calculations, balancing computational efficiency with the need for accurate electronic structure predictions. Advanced basis sets can provide highly accurate results but require more computational time and memory compared to simpler basis sets. Researchers can effectively model molecular systems and gain valuable insights into their properties and behaviors by selecting an appropriate basis set.

Quantum mechanical (QM) calculations are computationally very expensive and hence cannot be used to study larger systems with tens of thousands of atoms for longer time scales. Hence, molecular dynamics simulations have been the method of choice when it comes to modeling biomolecular systems and processes. In the next section, we give a brief overview of molecular dynamics simulation and its application in studying biological processes.

2.2 Molecular Dynamics simulations

Molecular dynamics (MD)[32] simulation is a computational method that predicts how atoms move in a molecular system over time. The basic principle behind MD is to numerically solve Newton's equations of motion for a system of interacting particles. In MD simulations, atoms and molecules are represented as particles, and their positions and velocities are tracked as they evolve in time. As shown in Equation 2.5, the interactions between particles are described by a force field[33, 34], which contains terms for bonded interactions (bonds, angles, dihedrals) and non-bonded interactions. Bonded interactions includes bonds, angles and dihedrals whereas non-bonded interactions includes terms for van der Waals forces and electrostatic (coulombic) interactions[35, 36].

$$U(r) = \sum_{bonds} k_b(b - b_0)^2 + \sum_{angles} k_\theta(\theta - \theta_0)^2 + \sum_{dihedrals} k_\phi[1 + cos(n\phi + \delta)] + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\left[\frac{q_i q_j}{\varepsilon r_{ij}} + \varepsilon_{ij}\left(\left(\frac{R_{min,ij}}{r_{ij}}\right)^{12} - 2\left(\frac{R_{min,ij}}{r_{ij}}\right)^{6}\right)\right] \quad (2.5)$$

In the Equation 2.5,
k_b = the spring constant corresponding to the stretching of the bond,
b = the instantaneous bond length for a given bond,
b_0 = the equilibrium bond length,
k_θ = spring constant corresponding to the bending of the bond angle,
θ = the instantaneous angle between the bond for a given bond angle,

θ_0 = the equilibrium bond angle,
k_ϕ = the height of the potential energy corresponding to the dihedral,
n = the number of maxima/minima that occur as ψ varies from $-\pi$ to π,
ϕ = the dihedral angle for a given dihedral,
δ = the phase shift for ϕ,
q_i and q_j = the charge on particles i and j, respectively,
ε = the permittivity in a vacuum,
r_{ij} = the distance between atom i and atom j,
$R_{min,ij}$ = the distance between atom i and j where the Van der Waals energy is minimum
ε_{ij} = the energy at this distance.

MD simulations serve as a potent tool in computational chemistry and biology, allowing researchers to study the dynamic behavior of molecular systems over time. Molecular dynamics (MD) also enables researchers to gain atomic-level insights into various chemical and biological processes, such as DNA-repair mechanism, protein folding, effects of solvents on protein folding dynamics, lipid-protein interactions, lipid–drug interactions, and protein-ligand interactions aiding in drug design.[37, 38, 39, 40, 41, 42]

MD simulations employ integration algorithms like the Verlet[35, 43] or Leapfrog[44] algorithms to simulate molecular motion over time. These algorithms numerically solve Newton's equations of motion in order to update the positions and velocities of the atom at each time step based on the forces calculated using the potential energy function (force fields). We will discuss these integration algorithms in the next section.

2.2.1 Integration algorithms

The central idea in MD simulations is to numerically solve Newton's equations of motion to predict the trajectories of atoms and molecules.[44] Several integration algorithms exist that can numerically solve these equations efficiently and accurately.[35, 36, 43]

Integration algorithms in molecular dynamics typically employ the finite difference technique, often based on Taylor series expansion, to calculate particle positions, velocities, and accelerations at each time step. The initial coordinates can be obtained from experimental techniques (X-ray, NMR, and cryo-crystallography) or modeling protocols, and the initial velocities are randomly assigned according to a Maxwell-Boltzmann distribution, Where, $f(v_i)$ is the probability of an atom i of mass m_i to have a velocity v_i at a given temperature, T; k_b is the Boltzmann constant.

$$f(v_i) = \left(\frac{m_i}{2\pi k_b T}\right)^{1/2} e^{\left(\frac{m_i v_i^2}{2\pi k_b T}\right)} \tag{2.6}$$

After determining the initial positions and velocities of atoms, the forces acting on each atom are computed as the negative gradient of the potential energy function. The acceleration of each atom is then determined by dividing the force by the atom's mass. Subsequently, various integration algorithms, such as Verlet, Leapfrog, and Velocity Verlet,[35, 45] are used to numerically solve the equations of motion. Among these algorithms, the Verlet algorithm is particularly notable for its widespread use. We start with the Taylor series expansion for the position at time $t + \delta t$ and $t - \delta t$ around the current time t.[35, 44]

$$r(t + \delta t) = r(t) + v(t)\delta t + 0.5\delta t^2 at + ... \tag{2.7}$$

$$r(t - \delta t) = r(t) - v(t)\delta t + 0.5\delta t^2 at - ... \tag{2.8}$$

Addition of Equation 2.7 and Equation 2.8 gives us Equation 2.9.

$$r(t + \delta t) = 2r(t) - r(t - \delta t) + a(t)(\delta t)^2 \tag{2.9}$$

Similar to Equation 2.9, the subtraction of the Equation 2.7 and Equation 2.8 gives us Equation 2.10.

$$v(t) = \frac{r(t + \delta t) - r(t - \delta t)}{2\delta t} \tag{2.10}$$

These algorithms are widely used in MD simulations due to their efficiency and stability. In molecular dynamics (MD), we do not consider the molecule's electronic structure. Electrons are usually not explicitly modeled in MD simulations. Instead, their effects are incorporated indirectly through empirical force fields, which describe the interactions between atoms based on their positions. This assumption simplifies computations but results in inherent limitations compared to QM-based methods.

In recent years, efforts have been made to combine the strengths of quantum mechanics and classical mechanics. The goal is to create machine learning based methods that are as accurate as quantum mechanics based methods and as computationally fast as classical mechanics based methods. In the next section, we introduce machine learning methods and algorithms. This introduction lays the groundwork for a detailed exploration of the ML-based methods used and developed in this thesis.

2.3 Machine learning

In this section, we will explore key concepts in machine learning, such as artificial neural networks (ANNs), their various types, activation functions, reinforcement learning (RL), and advanced machine learning techniques. Understanding the different types of ANNs and their applications is crucial for building effective ML models. Additionally, activation functions play a

vital role in ANNs by introducing non-linearity, allowing the network to learn complex patterns and improve performance. RL is another important concept in machine learning, where an agent learns to make decisions through interaction with an environment, receiving feedback in the form of rewards or penalties. Lastly, we will explore advanced ML techniques and innovations that are shaping the future of machine learning, enhancing its capabilities and performance. This section sets the foundation and will help the reader better understand the ML methods and algorithms that are used and developed in this thesis.

2.3.1 Artificial neural network (ANN)

Artificial Neural Networks (ANNs) are ML algorithms inspired by the function and structure of a human brain.[46, 47] Similar to the human brain, ANNs consist of interconnected neurons/nodes that process the information and learn patterns from data. These interconnected neurons/nodes are organized as layers, such as an input layer, one or more hidden layers, and an output layer.[48] Each connection between neurons/nodes in an ANN has a weight that dictates how much influence one neuron has on another. The basic working principle of an Artificial Neural Network (ANN) involves passing input data through the network, which is processed through different layers to produce an output. Below is the list of layers in ANN and their function:

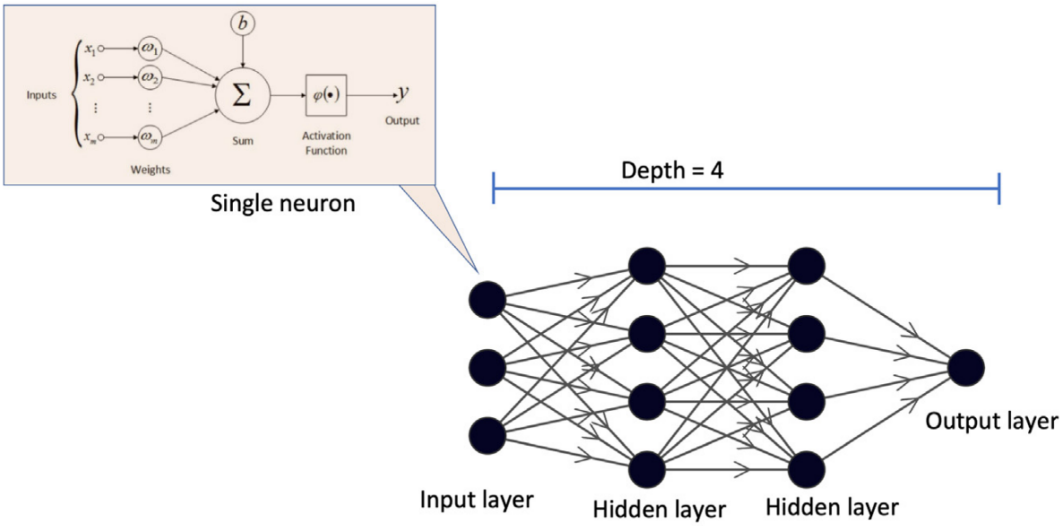


Figure 2.1: Neural network with 4 layers (1 input layer, 2 hidden layer and 1 output layer). Enlarged image of single neuron depicts the inputs (x), weights (w), bias (b), activation function (φ) and output (y). (Source - Nishant Kumar and Martin Raubal. Applications of deep learning in congestion detection, prediction and alleviation: A survey. Transp. Res. Part C Emerg. Technol., 133 (November):103432, 2021.)

1. **Input Layer:** The input layer accepts the input data (feature vector) and passes it to the neurons/nodes in the next layer, i.e., the Hidden layers.

2. **Hidden Layers:** The hidden layers perform mathematical operations on the input data using the weights associated with each connection. Each neuron in a hidden layer calculates a weighted sum of the inputs received from the previous layer before passing this sum through an activation function to introduce non-linearity. After passing through the activation function, the result is passed to the next layer of neurons/nodes, i.e., the output layer.

3. **Output Layer:** The final result or prediction is given by the output layer based on the processed input from the hidden layers. The type of problem being solved, such as classification or regression, decides the number of neurons in the output layer.

The mathematical representation of a neuron's output y in layer l given inputs $x_1^{(l)}, x_2^{(l)}, ..., x_n^{(l)}$ and weights $w_1^{(l)}, w_2^{(l)}, ..., w_n^{(l)}$ is:

$$y^{(l)} = f(\sum_{i=1}^{n} w_i^{(l)} x_i^{(l)} + b^{(l)}) \tag{2.11}$$

Here, f is the activation function (e.g., ReLU, Tanh), and $b^{(l)}$ is the bias term for layer l. The learnable parameters are the weights and biases shown in Equation 2.11. They are adjusted during the training process to minimize the error between the predicted output and the actual output (ground truth, label) using optimization techniques like gradient descent.

2.3.2 Different types of ANN

In the previous section, we saw how ANNs learn from data by adjusting their weights and biases through a process known as training, which allows them to make predictions or classify data based on patterns learned from the training data. In this section we will discuss some of the key variations and algorithms derived from ANNs.

1. **Recurrent Neural Networks (RNNs):** RNNs are designed to operate on sequential data, where the order of inputs matters. They have connections that form directed cycles, allowing them to exhibit dynamic temporal behavior. RNNs are frequently used in natural language processing (NLP) and speech recognition. Their applications extend to tasks like molecular sequence analysis, molecular design, and reaction prediction in computational chemistry.

2. **Convolutional Neural Networks (CNNs):** Convolutional Neural Networks (CNNs) are a class of ANN commonly used for analyzing images.[49] One key feature of CNNs is their ability to learn spatial hierarchies of features through convolutional layers. These layers apply a series of filters (kernels) to the input data, extracting important features at different spatial scales, which makes them particularly effective for tasks like image recognition, object detection, and image classification.

3. **Long Short-Term Memory (LSTM):** LSTM is a specialized type of RNN capable of learning long-term dependencies in sequantial data.[50] It introduces memory cells and various gating mechanisms to selectively remember or forget information over a long sequences, making it particularly effective for tasks requiring modeling of context over time.

4. **Generative Adversarial Networks (GANs):** GANs consist of two neural networks, a generator network and a discriminator network, which are trained simultaneously through adversarial training.[51] The generator learns to generate new data samples, such as images, audio, or text, that are indistinguishable from real data, while the discriminator learns to differentiate between real data and data generated by the generator. The two networks are trained in a minimax game, where the generator aims to fool the discriminator, and the discriminator aims to classify real and generated data correctly. This adversarial training process results in the generator producing increasingly realistic data samples. GANs are used to generate new content, such as images, music, and text.

5. **Autoencoders:** Autoencoders are variants of ANN used for unsupervised learning of representation (encoding) for a set of data.[52, 53, 54] The key idea behind autoencoders is to construct a network architecture that includes a bottleneck, forcing the model to compress the input data into a latent-space representation. This compression is achieved by learning the underlying structure and correlations present in the input data. By doing so, the autoencoder can effectively capture and represent the essential features of the data. The compressed representation can then be used for various tasks such as data denoising, dimensionality reduction, and even the generation of new data samples. Overall, autoencoders are powerful tools for learning meaningful representations of complex data.

6. **Graph Neural Networks (GNNs):** GNNs are designed to operate on graph-structured data. Graph Neural Networks (GNNs) are particularly relevant in computational chemistry as they excel in learning molecular representations. In this context, atoms are represented as nodes and chemical bonds as edges in a graph. GNNs can aggregate information from neighboring atoms (nodes) to learn complex molecular features, enabling tasks such as node classification (e.g., atom type prediction) and link prediction (e.g., chemical bond prediction) in molecular graphs.[55, 56, 57, 58]

7. **Mixture Density Networks (MDNs):** MDNs are a type of neural network architecture used in machine learning for modeling complex probability distributions, particularly when the data is multimodal (i.e., consists of multiple distinct peaks).[59, 60] MDNs predict not just a single output value but a set of parameters that define a mixture model, typically a Gaussian mixture model (GMM). These parameters include the means, variances, and mixture weights of the individual Gaussian components. By using MDNs,

one can model complex, non-linear relationships in the data and capture uncertainty in predictions, making them useful for tasks like generating diverse outputs in generative modeling or predicting multimodal outcomes in regression problems.

These are just a few examples of the many variations and algorithms that have been derived from the basic ANN architecture. Each variation is designed to address specific challenges or leverage specific characteristics of the data, making neural networks a versatile and powerful tool in machine learning and artificial intelligence.

2.3.3 Activation functions

Activation functions are a crucial component of ANNs, which are responsible for introducing non-linearities into the network and facilitating ANNs to learn complex patterns in data. Activation functions determine the output of a neuron/node, which is then passed on to the neuron/node present in the next layer of the network. In Figure 2.2, we illustrate the behavior of several activation functions with respect to the input (x) on the x-axis and their corresponding output on the y-axis.

Below, we give examples of some of these activation functions.

1. Rectified Linear Unit (ReLU) is a popular activation function used in neural networks. It is defined as $f(x) = \max(0, x)$, which means it outputs the input value if it is positive and zero otherwise. ReLU is favored for its simplicity and efficiency, as it introduces non-linearity to the network without computationally expensive operations. One of the key advantages of ReLU is that it helps mitigate the vanishing gradient problem, which can occur in deep networks. However, ReLU can suffer from the "dying ReLU" problem, where neurons can become inactive and stop learning if they consistently output zero. Despite this limitation, ReLU remains a widely used activation function in many state-of-the-art neural network architectures.

2. Continuously Differentiable Exponential Linear Unit (CELU) is an activation function that offers a smooth alternative to the ReLU family of functions. It is defined as $f(x) = \max(0, x) + \min(0, \alpha(\exp(x/\alpha) - 1))$, where α is a hyperparameter that controls the rate of decay for negative inputs. CELU provides a smooth gradient for all values of x, unlike ReLU, which has a gradient of zero for negative inputs. This property can be beneficial for training deep neural networks, as it helps prevent the "dying ReLU" problem and can lead to more stable convergence during training. Additionally, CELU retains the linear behavior for positive inputs, which can be advantageous for capturing linear relationships in the data.

3. Leaky ReLU It is an extension of the traditional ReLU. It is defined as $f(x) = \max(\alpha x, x)$, where α is a small constant, typically set to a small value like 0.01. Unlike the standard

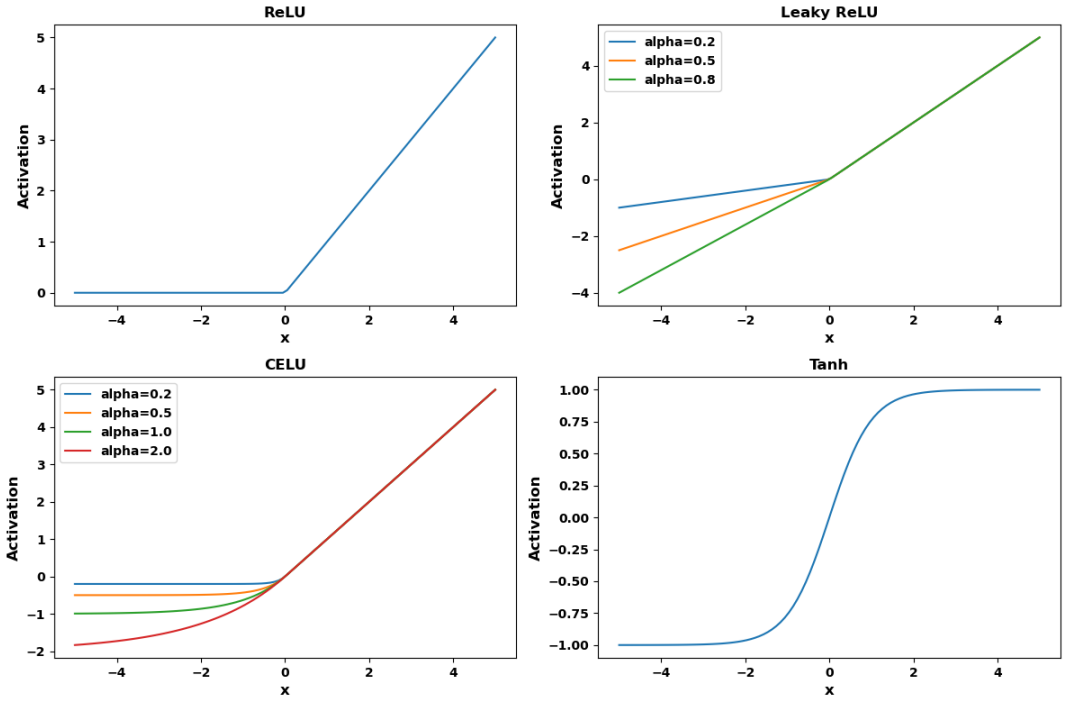


Figure 2.2: Comparison of activation functions: ReLU, Leaky ReLU, CELU, and Tanh.

ReLU, which sets all negative values to zero, Leaky ReLU allows a small, non-zero gradient for negative inputs. This small slope for negative inputs helps address the "dying ReLU" problem, where neurons can become inactive and stop learning during training.

4. Hyperbolic Tangent (Tanh) is another commonly used activation function in neural networks and is defined as $f(x) = \tanh(x)$. Tanh squashes the input values to the range [-1, 1], producing zero-centered outputs. This makes it useful for models where one needs normalized output between the ranges [-1, 1]. However, Tanh can suffer from the vanishing gradient problem for very large or very small inputs.

In general, the selection for activation function depends on the specific necessities of the neural network and the nature of the problem being solved. Various activation functions exhibit distinct properties that can influence the network's performance in terms of training speed and prediction accuracy.

2.3.4 Reinforcement learning (RL)

Reinforcement learning is the subfield of ML, where an agent learns how to make smart decisions by interacting with its environment.[61] It learns through the trial and error method and receives feedback as rewards or penalties. Consider RL as training a dog, as illustrated in Figure 2.3. In this analogy, the dog represents the RL agent, and the human represents the

Figure 2.3: The cartoon illustrates the intuition behind reinforcement learning. The Human can be considered as the environment, the dog as an agent and command, gestures etc are observations (state) in the RL setting.

environment. The human issues commands or gestures, which serve as observations or states. The dog (agent) observes these states and takes actions to receive a treat (reward) from the human (environment). The human rewards correct actions and penalizes incorrect ones. The goal of the dog (agent) is to maximize its reward by executing the correct actions. Through this process, the dog learns to perform the correct action based on the command or gesture to receive a reward.

The general formalism of RL can be described as follows:

1. **Agent:** The entity that interacts with the environment and makes decisions based on the feedback received from the environment.

2. **Environment:** The external system with which the agent interacts, providing feedback based on the agent's actions.

3. **State (S):** The representation of the environment at a specific point in time, crucial for the agent's decision-making process.

4. **Action (A):** The choices available to the agent, which affect the environment's state.

5. **Reward (R):** The feedback from the environment to the agent after each action, indicating the immediate benefit of that action.

6. **Policy (π):** The strategy or set of rules that the agent follows to select actions based on the current state.

7. **Value Function (V or Q):** An estimate of the expected cumulative reward that an agent can achieve from a given state (V) or state-action pair (Q).

The goal of the agent is to learn an optimal policy ($\pi*$) that maximizes the expected cumulative reward over time. This process usually involves iteratively engaging with the environment, observing states, taking actions, receiving rewards, and adjusting the policy based on these interactions.

2.3.5 Machine Learning: Advanced techniques and innovations

Machine learning (ML) techniques have advanced significantly over the years, resulting in the development of various sophisticated models and techniques. Below, we highlight some prominent techniques, although the list is by no means exhaustive.

1. **Residual Connections:** A residual connection is a learnable mapping that runs in parallel with a skip connection to form a residual block.[62] Along with skip connections, residue connections are used in architectures like ResNet[62] for image recognition tasks, allowing the training of very deep networks with hundreds of layers. As shown in Figure 2.4, $F(x)$ represents the residual mapping to be learned and identity x is the skip connection.

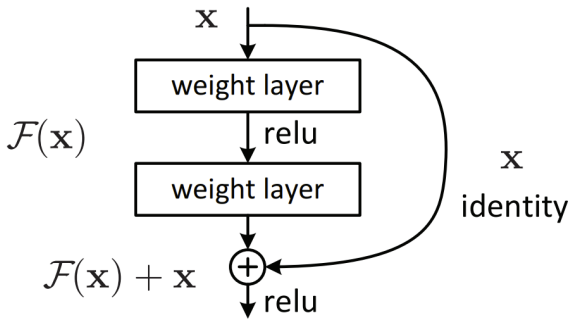


Figure 2.4: Residual block $F(x)$ along with identity as skip connection. (Source - Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., 770–778, 2016.

2. **Skip Connections:** Skip connections enable information to bypass specific layers within a neural network, mitigating the vanishing gradient problem by creating a shortcut for the gradient to propagate through the network.[62, 63] They are typically used in deep neural networks, such as ResNet, for image classification and object detection tasks.

3. **Attention Mechanism:** Attention mechanisms empower neural networks to concentrate on particular segments of the input sequence by assigning varying weights to different parts of the input.[64] This helps the model to selectively concentrate on relevant information, rendering them suitable for NLP duties like machine translation and text summarization, where focusing on different input sequence parts is context-dependent.

4. **Batch Normalization:** Batch normalization is a method employed to normalize the input of each layer by modifying and scaling the activations. This technique helps reduce internal covariate shift, accelerating the training process. It is used in various deep-learning models to improve training speed, stability, and generalization.

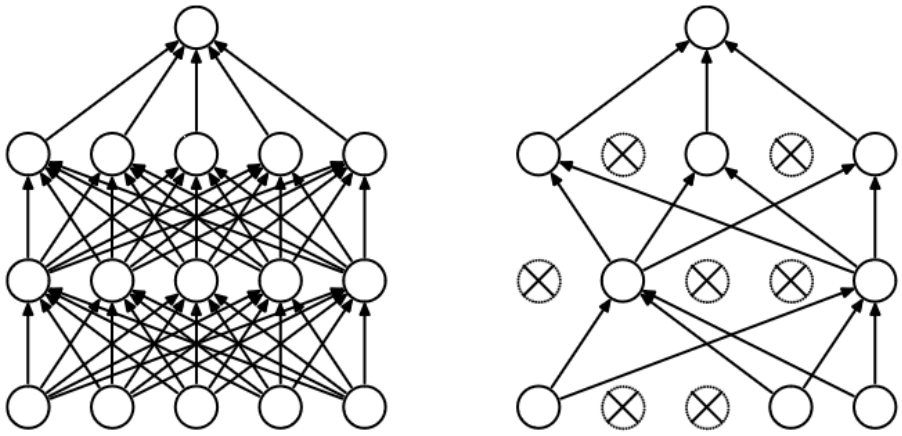


Figure 2.5: On the left is a standard neural net with 2 hidden layers. On the right the crossed units have been dropped. (Source - Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15:1929–1958, 2014.)

5. **Dropout:** Dropout is a regularization method that involves randomly dropping out or deactivating neurons during training, as shown in Figure 2.5. This process helps reduce overfitting and enhances the model's generalization ability. Dropout is widely used in deep neural networks, particularly in fully connected layers, to prevent overfitting and improve the model's robustness.[65, 66]

6. **Data Augmentation:** Data augmentation is a technique used to artificially expand the size of a dataset by creating modified versions of data samples, improving the robustness of models and reducing overfitting. It is widely used in computer vision tasks, such as image classification and object detection, where generating additional training data can improve the performance of the mode.[67]

7. **Transfer Learning:** Transfer learning is a method that involves using a pre-trained model as a starting point for a new task, which can help accelerate the training process and enhance the model's performance, especially when the new task has limited data. This approach is frequently employed in image recognition applications, where models like VGG, Inception, and ResNet are fine-tuned on specific datasets to achieve improved performance. In computational chemistry, ANI1-ccx model is one such example of transfer learning.[68] We use ANI1-ccx model in chapter 6 and chapter 7 to predict energy and forces.

2.4 Molecular geometry optimization

This section introduces three optimization algorithms, i.e., FIRE, MDMin, and BFGS. These optimizers have been used in chapter 6 and chapter 7, where we compare our RL-based optimizer with these three optimizers.

2.4.1 Fast Inertial Relaxation Engine (FIRE)

The Fast Inertial Relaxation Engine (FIRE) algorithm is a method that is based on molecular dynamics to efficiently relax a molecular system to a stable configuration. Any common MD integrator can be used as the basis for the propagation. At the beginning of the algorithm, the given atomic positions are used, and the initial velocity of the system is set to zero. It then calculates the forces and velocities acting on each atom based on the system's potential energy. Next, it adjusts the velocities of the atoms to reduce these forces, preventing the atoms from overshooting their equilibrium positions. To maintain stability, the algorithm normalizes the velocities if they are too high. Subsequently, it updates the positions of the atoms based on their velocities, gradually moving the system towards a more stable state. This process is repeated iteratively until the system reaches a stable configuration, as determined by a convergence check.[69]

2.4.2 MDMin

The MDMin algorithm is a variation of the velocity-Verlet molecular dynamics algorithm used to simulate molecular systems. It involves solving Newton's second law numerically but includes an additional check after each time step. This check examines the dot product between the forces acting on the atoms and their momenta. If this dot product is zero, it indicates that the system has passed through a local minimum in the potential energy surface. At this point, the algorithm sets the momentum to zero. Unlike traditional molecular dynamics, where atom masses are considered, in MDMin, all masses are set to one.

There are two versions of the MDMin algorithm. The first version tests and stops each atom individually. In contrast, the second version treats all coordinates as a single vector and sets all momenta to zero if the dot product between the momentum vector and the force vector is zero. The latter version is implemented in the ASE package used in the thesis. Although the MDMin algorithm is simple, it performs well because it leverages the physics of the system. When the system is close enough to a minimum that the potential energy surface is approximately quadratic, switching to a minimization method with quadratic convergence, such as Conjugate Gradient[70] or Quasi-Newton, becomes advantageous for further refinement.[71]

2.4.3 Broyden–Fletcher–Goldfarb–Shanno (BFGS)

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is a popular optimization method used to find the minimum of an objective function. It belongs to the class of quasi-Newton methods, which aim to find the minimum of a function by iteratively improving an approximation of the inverse Hessian matrix, which represents the local curvature of the objective function. At each iteration, the algorithm adjusts the search direction based on the gradient of the objective function and the current estimate of the inverse Hessian matrix. By combining information from previous iterations, the BFGS algorithm dynamically adapts its search direction to navigate the optimization landscape efficiently. This adaptive nature allows it to converge rapidly to the minimum of the objective function while avoiding unnecessary oscillations. Unlike traditional gradient descent methods, BFGS typically requires fewer iterations to converge and is less sensitive to the choice of the initial guess. As a result, the BFGS algorithm provides a robust and efficient approach for optimization problems, making it widely used in various fields such as machine learning, numerical optimization, and scientific computing.[72, 73, 74, 75]

This chapter has equipped us with the following: 1. the fundamental understanding of the principles of QM and MM, 2. a basic understanding of various ML models and techniques, and 3. a basic understanding of optimization algorithms. Equipped with this knowledge, we can dive deep into the ML-based potentials known as neural network potential (NNPs). In the next chapter, we will discuss four NNPs and perform a benchmark study on NNPs.

Chapter 3

Benchmark study on Deep Neural Network Potentials for Small Organic Molecules

3.1 Introduction

Accurate modeling of biological and chemical processes requires precise estimation of energies and physio-chemical properties of molecules. The density functional theory (DFT) method, even though is very accurate and has a central role in computational chemistry, are computationally prohibitive in high-throughput applications. On the other hand, ML algorithms, trained on known examples, can be used to predict the properties of new molecules at much reduced computational cost with comparable accuracy as DFT.[76, 77] ML algorithms such as Generative Adversarial Networks (GAN),[51] coupled with reinforced learning,[78, 79] have succeeded in finding new molecules with the desired drug-like properties.[80, 81, 82] ML models have been successfully trained and used as predictive models for interatomic potential energy surfaces,[83, 84, 85, 86, 87, 88] molecular force field,[89, 90] electron densities,[91] density functionals,[92] protein structure prediction,[93, 94, 95] protein-protein interactions,[96] material property prediction,[97, 98, 99] retrosynthesis,[100] and drug discovery.[80, 81, 101, 102]

Researchers use molecular modeling methods to understand biological and chemical processes. Biological and chemical molecular modeling requires an accurate estimation of the energies and physio-chemical properties of the molecules. DFT methods are popular quantum mechanical (QM) methods of choice for calculating accurate molecular energies and physio-chemical properties. They are computationally expensive and are impractical for much larger systems,[103] especially at a large scale. Hence, to model larger systems and to perform high throughput exercises, classical force fields (FF) are used with significantly reduced computational cost but with reduced accuracy compared to QM. FFs simplify the description of inter-atomic interactions by summing components of the bonded, angular, dihedral, and non-bonded contributions fitted to a simple analytical form. Except few, most FFs dont account

for polarizability. Further more, FFs have to be parameterized for modeling various biological and chemical processes. [3, 4, 5, 104, 105]

In recent years, ML methods have been successful in circumventing the need to solve Schrödinger equation (SE). ML methods learn the high dimensional function (HDF), f, by training on large amounts of data (eq., energy). The ability to learn the HDF (f), which can then be used to map function $f(Z_i, r_i) \rightarrow E$, which returns the energy (E). Hence, using learned HDF, one can estimate the properties of unknown compounds or structures using nuclear charges Z_i and atomic positions r_i.[2]

A host of supervised ML algorithms are available to train and optimize HDF, f, to approximate the output value for a given input. One such class of algorithms are known as the Artificial Neural Networks (ANN).[106, 107] These techniques have been used to tackle various complex problems in natural language processing[108] and computer vision.[66] Feed-forward neural networks (FFNN) are a class of ANN that have been proven to be general function approximators,[109] as they have the ability to transform the input into a new feature (latent) space, in which it becomes correlated with the output. The transformation is done sequentially through several layers and is typically highly non-linear. Hence feed-forward NNs are suitable for learning $f(Z_i, r_i) \rightarrow E$, which approximates PES.

Previously, NNs have been used to approximate PES for small molecules with the idea of many-body expansion.[110] While being accurate, these methods scale poorly due to a large number of individual NN involved, typically, one for each term in many-body expansion. Various feature vectors have been developed recently, such as the smooth overlap of atomic positions (SOAP), which was introduced by Bartók et al.[86] Nuclear charges (Z) and a matrix of interatomic distances were used as input to the model for prediction of the energy of the molecule by Schütt et al.[111, 112] List of bonds, angles, non-bonded, and dihedrals (BAND) is used as a feature vector in BAND-NN by Siddhartha et al.[113] Atomic environment vector (AEV) was developed by Smith et al., [84] which is a modified version of Behler-Parrinello symmetry function[85] (BPSF).

$$E_{tot} = \sum_{i=1}^{N} E_i \tag{3.1}$$

As it was later realized that energy is an extensive property and can be decomposed into atomic contributions. As shown in Equation 3.1, where E_i is the energy contribution of an atom i of the molecule with a total number of N atoms. Hence emerged high dimensional atomic NN (HDNN), which allowed one to use the same size network for different size molecules.[84, 85, 111, 112] HDNN can be broadly classified into two types, one is "descriptor-based," and the other is the "message-passing" variant.

In the descriptor-based variant, the environment information about an atom is encoded in a hand-crafted feature vector, e.g., BAND, AEV, and BPSF. In feature vectors such as BAND, an atom's local environment is encoded using a list of bonds, angles, non-bonded, and dihedral

interactions. For feature vectors such as BPSF and AEV, the local environment of atom i is encoded as an array of values. These values are calculated using symmetry functions which mathematically combine distances and/or angles between an atom of interest i and all other atoms in its neighborhood. This kind of atom-wise feature vector formulation makes these feature vectors invariant to translation, rotational, and permutation of equivalent atoms. The energy of the molecules is calculated as shown in Equation 3.1 where E_i is the energy contribution of the atom of interest. These invariances are desired property for feature vectors because NNs are numerical algorithms which will produce different output values if input changes due to such transformation, which in principle, does not change the molecule's conformation. Examples of models which use hand-crafted feature vector are BAND-NN, ANI, and TensorMol. ANI and TensorMol approaches are the extensions of the original approach proposed by Behler and Parrinello[85] or the Deep Potential Molecular Dynamics (DPMD) model.[114]

The second type is the "message-passing" variant, which takes nuclear charges and cartesian coordinates as input, and tries to learn a meaningful representation of the chemical environment by exchanging information between individual atoms using a deep neural network, also called as Deep learning (DL). This approach was first introduced by the DTNN and has since been refined in other DNN architectures, for example, SchNet,[112] HIP-NN,[115] and PhysNet.[116]

It is clear that novel machine learning methods have the potential to provide efficient means for predicting the properties of molecules. However, this potential has been limited by the lack of standard comparative evaluations. Recently, Folmsbee et al. have compared the performance of NNP for representing PES and geometry optimization[117] whereas Gastegger et al. compared the accuracy of NNP on *trans* alkanes.[118] Algorithmic papers often benchmark proposed methods on disjoint dataset collections and highlight a different kind of applicability, making it a challenge to gauge whether a proposed technique does, in fact, improve performance and applicability. Hence, in this study, we compare four NNP models viz. ANI, PhysNet, BAND-NN, and SchNet to evaluate their performance using a common dataset across different test cases: (i) Comparison of the goodness of fit and transferability among these NNPs. It checks how well these models perform when trained on N atom molecules and are tested on molecules with more than N atoms; (ii) Their performance on structural and geometric isomers; (iii) Their ability to produce smooth, physically meaningful surface with respect to bond, angles, and torsional angle changes; (iv) The applicability of these models for geometry optimization and evaluates their performance using various optimization algorithms.

3.2 Methods

3.2.1 Data selection

The ANI-1 dataset was chosen for the experiments, as they cover a large conformational space so that the models, when trained, are not confined to equilibrium structures only. This makes it possible to evaluate ML models for calculating energies of non-equilibrium structures and, subsequently, geometry optimization. The ANI-1 dataset was built from an exhaustive sampling of a subset of the GDB-11 database containing molecules upto 8 heavy atoms and limiting the atomic species to C, N, O and H, which gives 57,947 molecules. These 57,947 molecules were DFT optimized with ωB97X density functional and 6-31G(d) basis set, which gives 57,462 DFT optimized starting structures. Smith and coworker then generated tens of thousands of non-equilibrium structures for each 57,462 equilibrium structure, using normal-mode sampling. For this work, we chose a subset of the ANI-1 dataset that includes all the equilibrium structures and non-equilibrium structures whose relative energies with respect to the corresponding minimum energy structure is less than 30 kcal/mol.[7, 113] The justification for choosing this subset is that software such as GaussView[119] and RDKit[120] can produce initial geometries near the energy minima. Also, in most of the cases, the aim of drug design/bio-molecular simulations is to model processes which are near to energy minima. Hence, the chosen subset of the dataset is deemed sufficient.

3.2.2 Training

All four models were trained from scratch on the subset of the ANI-1 dataset described in the previous section. A train:test:validation data split in the ratio of 80:10:10 was used. This results in $\sim$ 7 million data points in the training set and $\sim$ 885k data points each in the test and validation sets. All the hyper-parameters were taken as default as stated in original papers of ANI-1,[84] PhysNet,[116] BAND-NN [113] and SchNet.[112] The training code of all the models is obtained from the code repositories provided by respective authors.

$$L = |E - E_{ref}| + L_{nh} \tag{3.2}$$

$$L_{nh} = \frac{\lambda_{nh}}{N} \sum_{i=1}^{N} \sum_{m=2}^{N_{modules}} \frac{(E_i^m)^2}{(E_i^m)^2 + (E_i^{m-1})^2} \tag{3.3}$$

All the models were trained on atomization energy with criteria being minimization of the Mean Squared Error loss function. In PhysNet, a different loss function is used as shown in equations 3.2 and 3.3 where E is predicted energy, E_{ref} is DFT reference energy, L_{nh} is "nonhierarchicality penalty", λ_{nh} is regularization hyperparameter set to 10^{-2}, E_i^m is energy

contribution of atom i from module m, the total number of modules used in the PhysNet model is $N_{modules}$ and N is total number of atoms in a molecule.[116]

For ANI, SchNet, and BAND model's ADAM, [121] optimizer is used to minimize the loss function, whereas, for PhysNet, we used ASMGrad.[122] The batch size for ANI is 1024, and 32 for BAND-NN, SchNet, and PhysNet. The learning rate of 0.01 was used for BAND, 0.0001 for SchNet, and 0.001 for ANI and PhysNet. Learning rate decay of 0.5, 0.1, 0.1, and exponential decay of 0.96 was used in ANI, BAND, PhysNet, and SchNet, respectively.

These four models incorporate different type of feature vectors. ANI uses AEVs, which are modified Behler and Parrinello symmetry functions (BPSF).[85] SchNet and PhysNet take nuclear charges and positions as input, and BAND-NN uses feature vectors inspired from classical force field equation. While ANI and BAND-NN use simple multilayered neural network, SchNet and PhysNet have modular architectures to allow for calculation of interactions between atoms and learn atomic features in a hierarchical manner. These four descriptors demonstrate the evolution of models from simple hand-engineered descriptor and multilayer NN to complex message-passing models with embedding and cfconv[123] layers. These varied sophistications within these models piqued our interest to see how well they perform when compared to each other.

3.2.2.1 ANI

The feature vectors used in ANI are called atomic environment vectors (AEVs). They represent the atomic environment around each atom. AEVs are constructed from 'symmetry functions' which probe the radial and angular environment for each atom. These are modified versions of the original Behler and Parrinello symmetry function (BPSF).[85] These AEV's are then fed into a multi-layered feed-forward neural network (FFNN), and the weights of the neural network are optimized during the training process. The energy is calculated as the sum of contributions E_i from each atom of a molecule, as shown in Equation 3.1. The architecture of ANI-model is described in Figure 3.1.

3.2.2.2 BAND-NN

BAND-NN is inspired from the force-field equation, where the energies are calculated as sum of the contributions from Bonds, Angles, Non-bonds and Dihedrals as described in Equation 3.4. The feature vectors are thus created from the list of bonds, angles, non-bonds and dihedrals (BAND) of a molecule denoting the atoms that make up these terms, distances, angles and dihedral angles. BAND-NN uses four different neural networks for each of the bonds, angles, non-bonds and dihedrals. The BAND-NN architecture is described in Figure 3.2.

$$E_{total} = \sum E_{bonds} + \sum E_{angles} + \sum E_{non-bonds} + \sum E_{dihedrals} \tag{3.4}$$

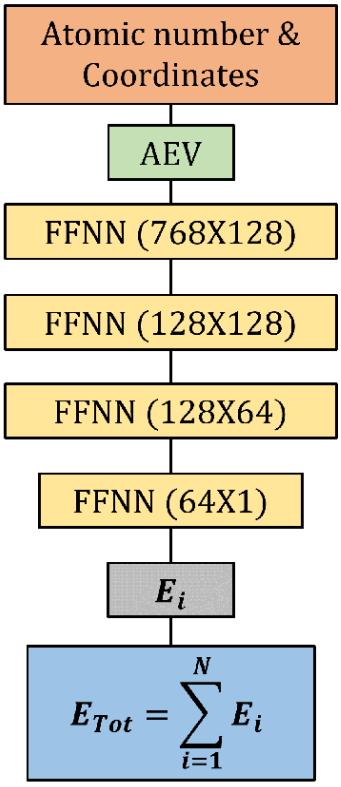


Figure 3.1: Overview of ANI architecture showing coordinates (orange box), atomic environment vector (AEV in green box). Feed Forward Neural Network (FFNN) (yellow boxes) with hidden layers specified in round brackets, E_i (grey box) is the energy contribution of atom i and E_{Tot} (blue box) gives the total energy of the molecule calculated as shown in Equation 3.1 (J. Smith et. al.[84] 2017)

3.2.2.3 SchNet

SchNet is a variant of Deep Tensor Neural Network (DTNNs).[111] This allows SchNet to learn representations for molecules that are invariant to rotation, translation and atom indexing. SchNet uses continuous-filter convolutions with filter-generating networks[123, 124] to model the atomic interactions inside the interaction block (blue boxes). As shown in Figure 3.3, SchNet has a modular architecture, atom embedding (green box), interaction refinements (blue boxes) and atom-wise energy contributions E_i. At each layer, the atomic representation is refined to better model the atomic interaction with the surrounding environment. SchNet takes nuclear charges (Z) and positions (R) as input during training.

3.2.2.4 PhysNet

PhysNet[116] is inspired from SchNet[112] and hierarchically interacting particle neural network (HIP-NN).[115] The architecture is modular and is shown in the Figure 3.4. To model the atomic interactions, PhysNet uses learnable distance based attention masks that select differ-

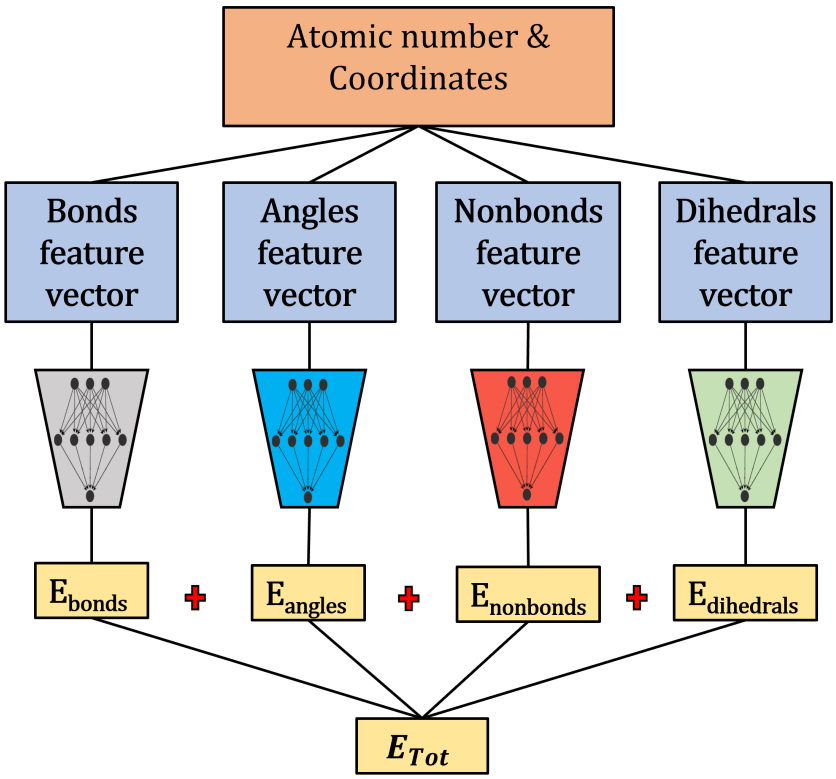


Figure 3.2: Overview of the BAND-NN architecture showing four different neural networks (trapezoids) for bonds (grey), angles (blue), non-bonds (red) and dihedral (green) inspired from force fields. (S. Laghuvarapu et. al.[113] 2019)

ent features based on the pairwise distance r_{ij} between atoms inside the interaction block. In addition, to circumvent the vanishing gradients problem arising due to vanishing of gradients in deeper neural nets, PhysNet uses pre-activation residual blocks, which skips one or several layers while training. In the first module, the input is an atom wise embedding vector. For other lower modules, the feature vector is obtained as output from the respective previous module. Feature vector gets updated as it moves down the five modules. As the feature vector passes through each module, it captures higher order interactions. Energy from each module is added to obtain the atomic energy E_i. These energies are further summed to calculate the total energy E_{Tot} of a molecule with "N" atoms.

3.3 Results and discussion

In this section, we present the comparative results of all four models viz. ANI, PhysNet, BAND-NN, and SchNet across four test cases. The first test case evaluates the goodness of fit, transferability, and overall performance of the models. For this, we tested the models using 885k molecules test-set. We also used one more test-set called random GDB, which has 1617

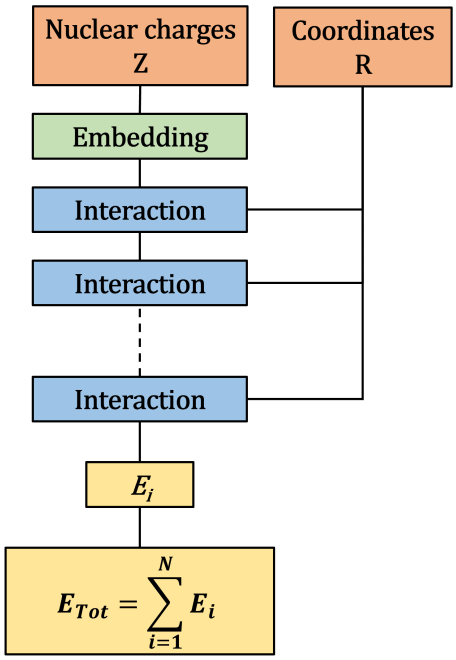


Figure 3.3: Overview of SchNet architecture with nuclear charge Z and coordinates R as input to embedding (green box) and interaction block (blue box) respectively. There are total of six interaction blocks, represented as dotted line. E_i is the atomic energy contribution of atom i and E_{Tot} is the total energy of a molecule with N atoms. (K. Schütt et. al.[112] 2018)

molecules with ten heavy atoms. The atom counts for random GDB test systems are ten heavy atoms, and total atom count ranges from 15 to 32 atoms.[84] In the second test case, we show the accuracy of the models in predicting the relative energies of DFT energy minimized $C_{10}H_{20}$ isomers with respect to the lowest energy isomer. The third test case is to evaluate the ability of the models to produce a smooth potential energy surface(PES). Hence, potential energy surface scans for bond stretch, angle bend, and two dihedral rotations on relatively large molecules are carried out using NNPs and are compared with reference DFT results. In the fourth test case, we perform geometry optimization on the following four molecules viz. methamphetamine (8 conformers), decane (8 conformers), fentanyl (8 conformers), and retinol (8 conformers) using various optimization methods, 32 conformer in total and use NNPs to predict energy during the optimization.

3.3.1 Accuracy and transferability

As mentioned in the *Data selection* section, all the conformers that were under 30 kcal/mol in the ANI-1 data set from the corresponding minimum energy structure were chosen for this study. We did 80:10:10 split; this resulted in 7 million data points in the training set and 885k data points each in the test and validation sets. We have summarized the performance of all four models on test-set in Table 3.1. A summary of the performances of all four models on

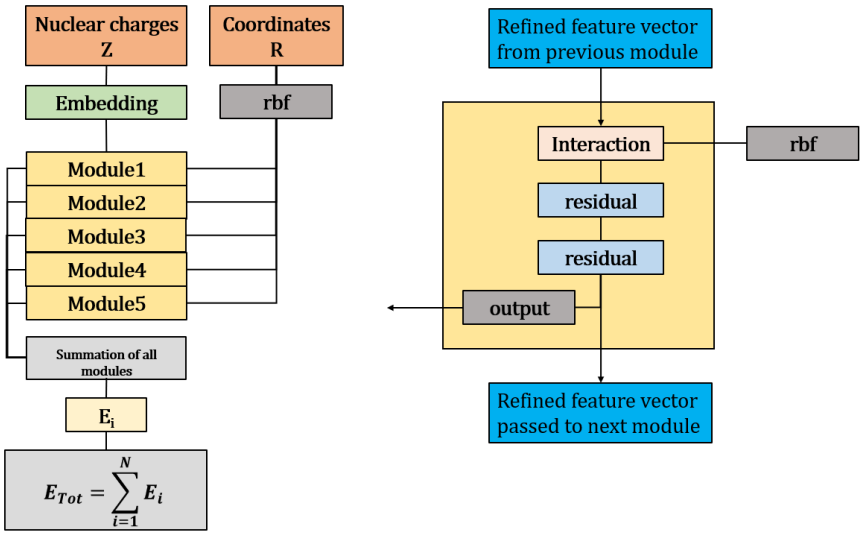


Figure 3.4: Overview of the PhysNet architecture (left). Embedding block, five module blocks (yellow boxes). The schema of module block (right). Atom-wise energy contribution E_i and their sum gives total energy E_{Tot} of a molecule with "N" atoms. (O. Unke et. al.[116] 2019)

a random GDB test set, which has 1617 molecules with ten heavy atoms, is given in Table 3.2. This shows the overall accuracy and transferability of these models as they are tested on larger molecules as compared to molecules in the training set. It is also important to know the performance of these models from their respective papers. ANI reported a mean absolute error (MAE) of 0.83 kcal/mol, and root means squared error (RMSE) of 1.17 kcal/mol on GDB-10 test set with structures that are 30 kcal/mol away from the energy minima and RMSE of 1.91 kcal/mol on all of GDB-10 test set.[84] Unke and Meuwly, in 2019, proposed PhysNet, which achieved an MAE of 0.19 kcal/mol on QM9 dataset.[116] SchNet model by K. Schütt et al. achieved MAE of 0.31 kcal/mol on QM9 dataset.[112] Both PhysNet and SchNet were trained on the QM9 dataset, which has only $\sim$ 134k small organic molecules, which is small as compared to the 7 million training set which has been used in this work. BAND-NN reported an MAE of 1.45 kcal/mol on the test-set and MAE of 2.1 kcal/mol on the GDB-10 dataset.

ANI, which uses a hand-crafted feature vector, achieves MAE of 0.39 kcal/mol and RMSE of 0.55 kcal/mol on test set. In contrast, PhysNet, which learns the feature vector directly from the data, achieves MAE of 0.40 kcal/mol and RMSE of 0.62 kcal/mol on the test set with the same size molecules as in the training set, i.e., molecules with eight heavy atoms. So when it concerns only accuracy, ANI and PhysNet have similar accuracy. But, when it concerns accuracy and transferability to bigger molecules, ANI achieves the best performance out of all four NNP's with MAE of 0.83 kcal/mol and RMSE of 1.17 kcal/mol on a random GDB test set which has larger molecules with ten heavy atoms in contrast to the training set which has molecules up to 8 heavy atoms.

Model	MAE	RMSE
ANI	0.39	0.55
PhysNet	0.40	0.62
BANDNN	1.45	1.82
SchNet	0.48	0.60

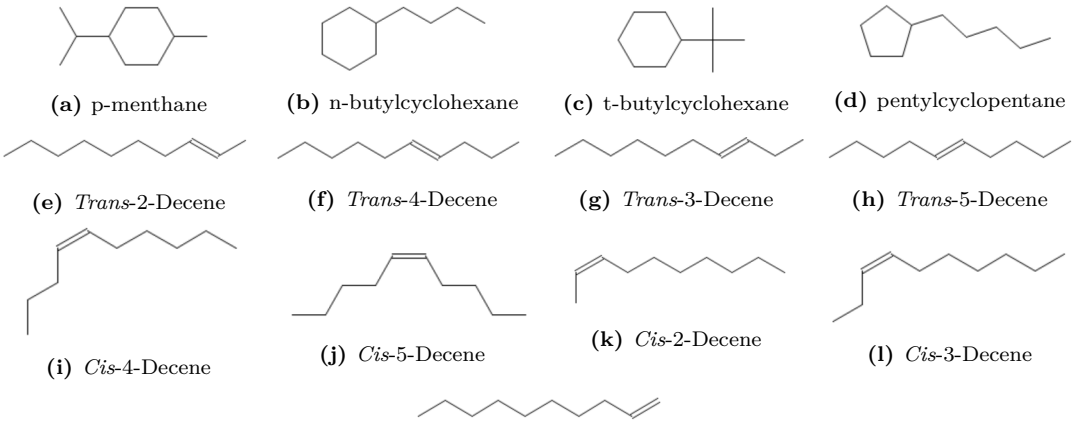
Table 3.1: Summary of performance of the models on Test Set (kcal/mol).

Model	MAE	RMSE
ANI	0.83	1.17
PhysNet	1.26	1.65
BANDNN	2.1	2.68
SchNet	1.51	1.89

Table 3.2: Summary of performance of the models on Test Set from GDB-10 dataset (kcal/mol).

3.3.2 Structural and geometric isomers

In this section, the models ability to accurately predict energies of geometric isomers with the empirical formula $C_{10}H_{20}$ is evaluated. Thirteen geometric isomers spanning diverse structural and geometric space are chosen, including linear chains, *cis-trans*, and ring containing isomers viz p-menthane, n-butylcyclohexane, t-butylcyclohexane, pentycyclopentane, *trans*-2-decene, *trans*-4-decene, *trans*-3-decene, *trans*-5-decene, *cis*-4-decene, *cis*-5-decene, *cis*-2-decene, *cis*-3-decene, and dec-1-ene. The structures of all thirteen isomers are shown in Figure 3.5. All isomers were optimized using the ωB97X/6-31G(d) level of theory using Gaussian 09 software.[125] The accuracy of these models in predicting the energies of the DFT optimized $C_{10}H_{20}$ isomers with respect to the lowest energy isomer are compared. Figure 3.5 shows that all the NNPs accurately predict the minimum energy structure and continue to accurately order the energies across ring containing structures, linear alkenes structures, and linear alkanes structures. ANI achieves the best performance with an RMSE of 0.29 kcal/mol, followed by PhysNet, with an RMSE of 0.52 kcal/mol. This experiment is indicative of the ability of these models to capture higher-order atomic interactions and differentiate between the isomers.

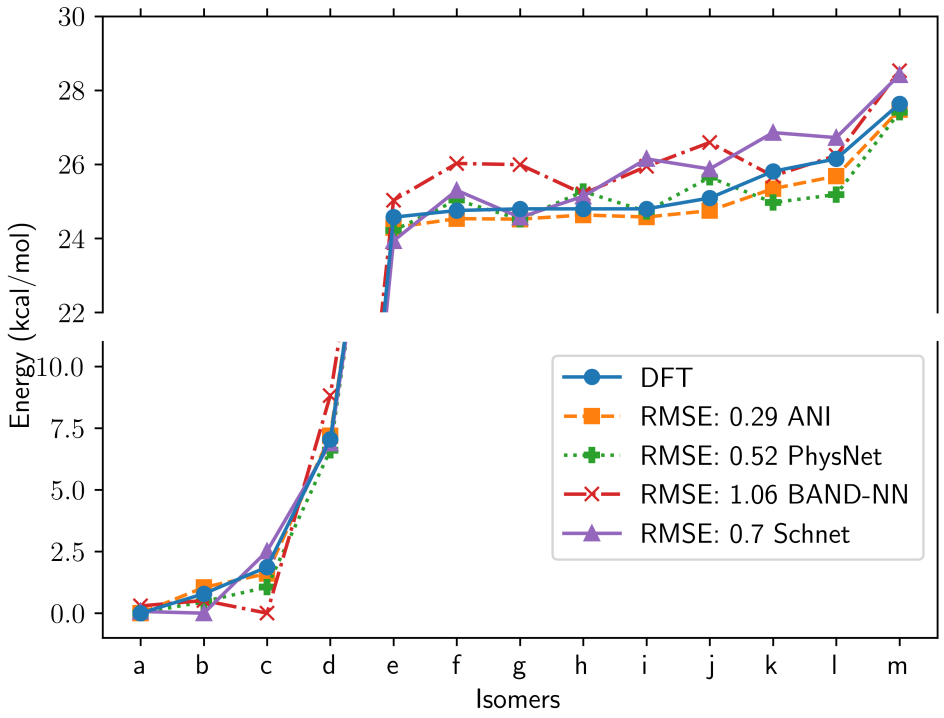


(a) p-menthane **(b)** n-butylcyclohexane **(c)** t-butylcyclohexane **(d)** pentylcyclopentane

(e) *Trans*-2-Decene **(f)** *Trans*-4-Decene **(g)** *Trans*-3-Decene **(h)** *Trans*-5-Decene

(i) *Cis*-4-Decene **(j)** *Cis*-5-Decene **(k)** *Cis*-2-Decene **(l)** *Cis*-3-Decene

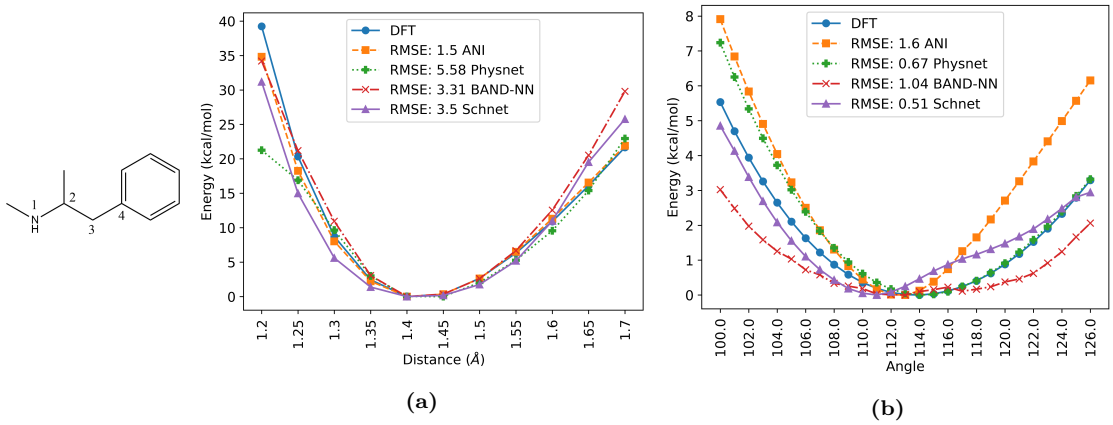
(m) Dec-1-ene



(n) Relative energies of $C_{10}H_{20}$ isomers

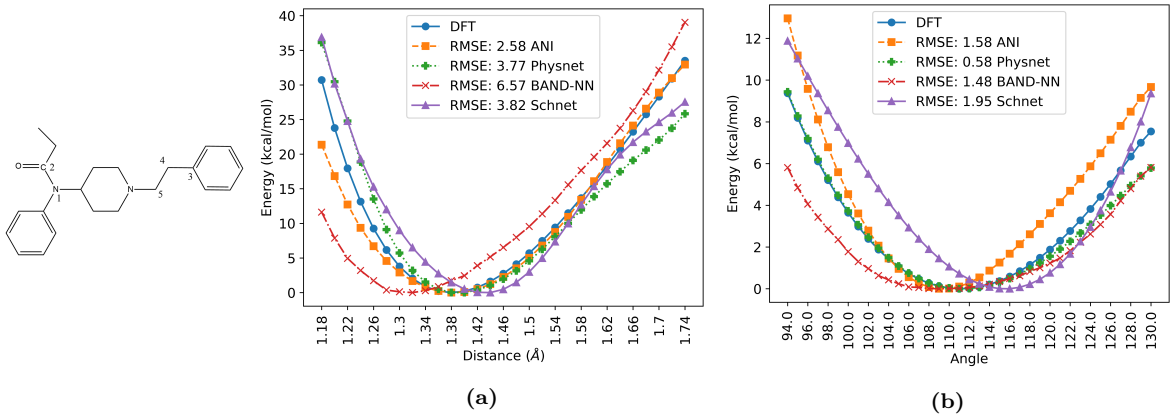
Figure 3.5: (a-m) Are all the structural and geometric isomers used to generate the data for the isomer case study and (n) Plot of relative energies of structures given in (a-m).

3.3.3 Potential energy surface



(a) (b)

Figure 3.6: Methamphetamine structure and specific atoms used for PES are shown (left). PES scan of (a) N1-C2 bond stretch and (b) C2-C3-C4 angle bend are shown.



(a) (b)

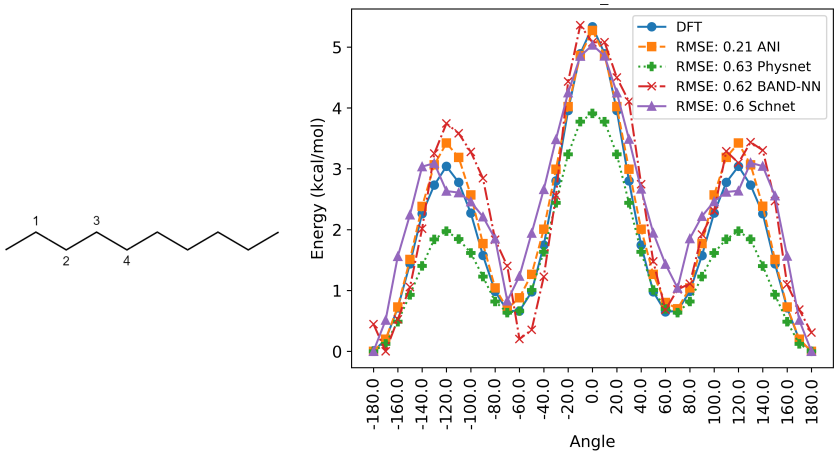
Figure 3.7: Fentanyl structure and specific atoms used for PES are shown (left). PES scan of (a) N1-C2 bond stretch and (b) C3-C4-C5 angle bend are shown.

In this section, the models ability to generate smooth and accurate potential energy surfaces (PES) is evaluated. From the experiments reported above, it is clear that these models can accurately predict the atomization energy of small organic molecules. However, it is important that these models not only accurately predict atomization energy but also be able to produce meaningful PES for a given molecule. Such behavior is necessary for these models to be applicable in energy minimization and force calculations in molecular dynamics simulations.

To examine how well these models produce PES, we performed PES scans of various molecules that are significantly larger than those in the training set. We generated PES profiles for N1-C2 bond stretch and C2-C3-C4 angle bend in methamphetamine, shown in Figure 3.6, N1-C2 bond stretch and C3-C4-C5 angle bend in fentanyl, shown in Figure 3.7, C1-C2-C3-C4 dihedral scan in decane and 4-cyclohexyl butanol, shown in Figure 3.8 and 3.9 respectively. C1-C2 bond

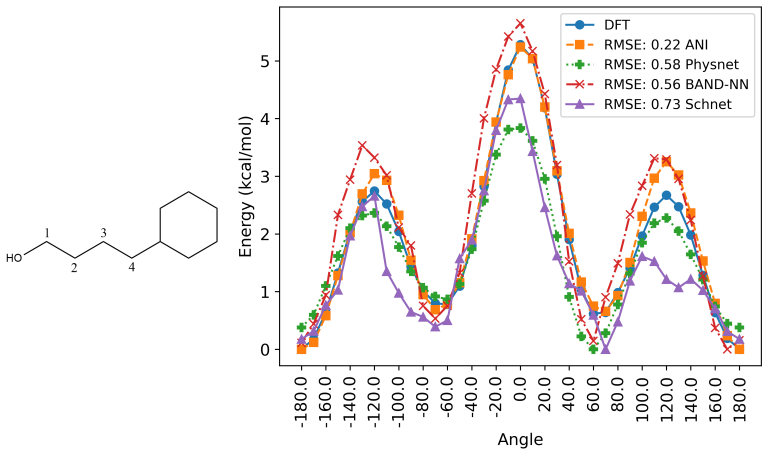
stretch, C2-C3-C4 angle bend, and C1-C2-C3-C4 torsion angle in pentadecane shown in Figure 3.10.

As evident from the Figures 3.6a, 3.7a and 3.10a for 3 C-N, C-N and C-C bond stretch scan respectively, ANI and PhysNet produce smoother curves and have predicted accurate bond equilibrium distances when compared to others. SchNet misses the bond equilibrium distance by 0.04 Å in Figure 3.7a. For PES scan along angle bend, as seen in Figures 3.6b, 3.7b and 3.10b, all the models have good agreement with DFT curve, and PhysNet has the best performance in all the 3 cases. For torsion angle, as shown in Figures 3.8a, 3.9a and 3.10c, ANI does better than the other models and is able to get an accurate estimate of dihedral rotation barrier.



(a)

Figure 3.8: Decane structure and specific atoms used for PES are shown (left). PES scan of (a) C1-C2-C3-C4 dihedral angle is shown.



(a)

Figure 3.9: 4-cyclohexylbutanol structure and specific atoms used for PES are shown (left). PES scan of (a) C1-C2-C3-C4 dihedral angle is shown.

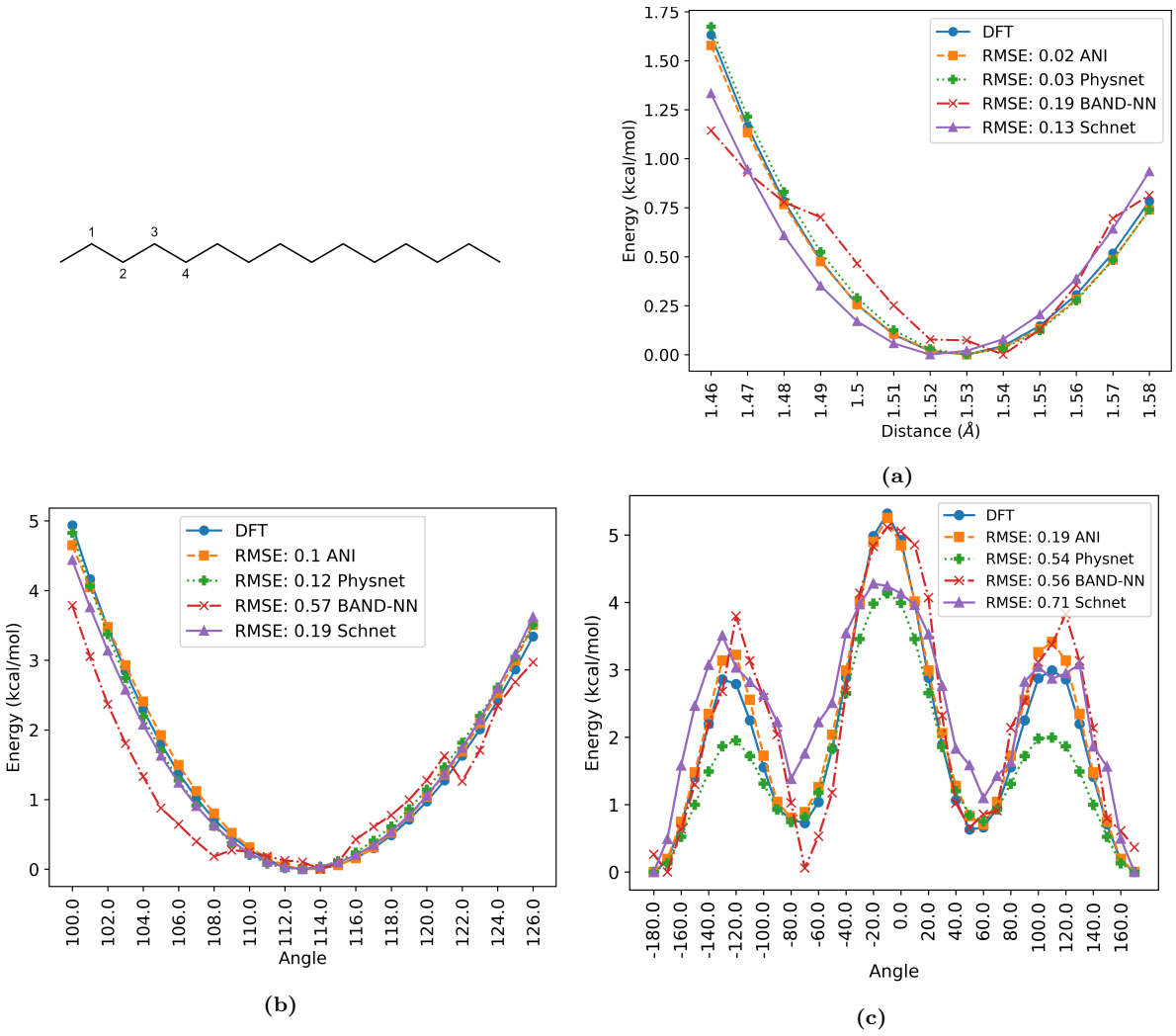


Figure 3.10: Pentadecane structure and specific atoms used for PES are shown (Top left). PES scan of (a) C1-C2 bond stretch, (b) C2-C3-C4 angle bend and (c) C1-C2-C3-C4 torsion angle are shown.

3.3.4 Geometry optimization

Along with property prediction and accurate representation of PES, it is also desirable for these models to be used with optimization algorithms to perform geometry optimization of non-optimized structures, circumventing the need for expensive DFT-based optimization. The combination of NNPs with optimization algorithms will not only allow us to accelerate the optimization process, but also give us access to optimize larger molecules which cannot be optimized using DFT. Four molecules, which are larger than the molecules in the training set, were considered for optimization, viz. decane ($C_{10}H_{22}$), fentanyl ($C_{22}H_{28}N_2O$), retinol ($C_{20}H_{30}O$) and methamphetamine ($C_{10}H_{15}N$). Eight conformers for each of the four molecules were created using the RDKit[120] package. These were then optimized to obtain their respective DFT energy minima. Eight different conformers were generated for each molecule so as to have a different starting point for each optimization and hence probe more of the conformational space.

We used fourteen different optimization algorithms spanning across different paradigms of gradient-free optimization techniques. To name a few, we have used the BFGS optimizer, a quasi-Newton method that searches for the stationary point on the optimization landscape using first-order gradients or gradients computed approximately. The Nelder-Mead algorithm is based on the iterative computation of non-linear simplices. The OnePlusOne is a subclass of evolutionary algorithms. The Covariance matrix adaptation (CMA) belongs to a class of evolutionary strategies (ES) using Gaussian sampling. Powell is a line search method built using iterative computation of search vectors. Cobyla works through iterative linear approximations of the objective function. For this work, the BFGS optimizer implementation in the scipy python package was used. Other non-gradient based methods are implemented in the Nevergrad[126] python package. The results of optimization have been summarized in the Appendix A Tables A.1-A.55. Each table in Appendix A represents data of one optimization method for a single molecule out of four molecules.

We have 14 optimization methods and four molecules, which make up a total of 56 tables (See Appendix A). One such set of optimization results are summarized in Table 3.3, where column number "1 to 8" represent eight conformers of a single molecule, and models are present in rows. The values in each cell represents energy ΔE_{opt} in kcal/mol calculated as shown in Equation 3.5, where E_{opt}^{Model} is the model optimized energy and E_{opt}^{DFT} is DFT optimized energy (ground truth). ΔE_{opt} indicates how far away is the model optimized conformer from the DFT optimized conformer. Hence smaller values of ΔE_{opt} are better as they indicate the ability of the models to predict optimized structures in very close agreement to that of the DFT optimized structures (ground truth). The last column, called "Best count", represents the number of counts out of 8 conformers for which the model got the best ΔE_{opt} value (smaller is better).

Table 3.3 shows optimization results for decane molecules using CMandAS3 method. As ANI gets the best ΔE_{opt} values (smaller is better) for conformers 2, 4, and 6 as compared to other models, we put "Best count" as 3 for ANI. Similarly, PhysNet gets the best values for five

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.43	**0.56**	0.62	**0.64**	0.44	**0.56**	0.96	0.58	3
PhysNet	**0.42**	0.57	**0.59**	0.66	**0.44**	0.57	**0.95**	**0.57**	5
SchNet	0.5	0.63	0.67	0.64	0.48	0.64	1.12	0.62	0
BAND-NN	0.57	0.68	0.79	0.83	0.57	0.65	1.14	0.73	0

Table 3.3: Optimization of 8 decane structures using CMandAS3 optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol).

conformers out of 8; hence we give 5 to PhysNet in the "Best count" column. Similarly, for all the 55 tables in Appendix A.

$$\Delta E_{opt} = E_{opt}^{Model} - E_{opt}^{DFT} \tag{3.5}$$

The experiments are performed on four molecules viz. methamphetamine (8 conformers), decane (8 conformers), fentanyl (8 conformers), and retinol (8 conformers) and fourteen different optimization methods. Altogether, there are 4*8*14 = 448 different optimization procedures. For decane, ANI achieves 55 best-optimized conformers out of 112 as compared to all other models, whereas PhysNet achieves 48 best-optimized conformers out of 112 (Table 3.4). For the three molecules viz., decane, fentanyl, and retinol, the ANI model has the highest number of best-optimized conformers count, whereas, for methamphetamine, PhysNet has a higher number of best-optimized conformers count of 56 out of 112. ANI performs better than the other three models as it gets the best optimization for 245 times out of 448. PhysNet gets 137 best-optimized conformers out of 448. Although results for all the conformers are not perfect, this could be further improved using appropriate molecular representations. We have also used gradient-based optimization BFGS, which gives the best-optimized conformer for decane, fentanyl, retinol, and methamphetamine, which are 0.49 (ANI), 4.30(ANI), 2.93(SchNet), and 0.95(PhysNet) kcal/mol away from DFT energy minima respectively. Overall, ANI performs better than other three models.

Molecules	Decane	Fentanyl	Retinol	Methamphetamine
ANI	55	66	85	39
PhysNet	48	15	18	56
SchNet	4	19	9	5
BAND-NN	5	12	0	12
Total	112	112	112	112

Table 3.4: Count of the best optimized conformers.

3.4 Conclusions

In this work, we evaluate and compare four models, i.e., ANI, PhysNet, SchNet, and BAND-NN, on their accuracy in energy prediction, transferability to larger molecules, ability to produce accurate PES, and applicability in geometry optimization. These models were originally trained and tested on different data sets, which makes their applicability and comparison difficult. This work provides a standard comparative evaluation of these models by training and testing them on the same sets. For accuracy and transferability, we report that the ANI model performs best with RMSE of 0.55 kcal/mol and 1.17 kcal/mol on $\sim$ 885k molecules and on the GDB-10 test set, which has 1617 molecules with ten heavy atoms randomly selected from the GDB-11 database. We also notice that both ANI (descriptor-based) and PhysNet (message-passing) produce smooth and meaningful surfaces and are potentially applicable in molecular dynamics simulations. All these models were also able to accurately differentiate different isomers of the same empirical formula $C_{10}H_{20}$. ANI and PhysNet achieve an RMSE of 0.29 kcal/mol and 0.52 kcal/mol, respectively. All these models are trained on DFT energies, but can also be extended to higher-level ab initio QM methods and larger basis sets. These models also show their potential for geometry optimization. ANI outperformed all models by optimizing 245 structures more accurately than others in 448 test structures. PhysNet stood next with 137 best-optimized conformers.

ANI is a variant of descriptor-based HDNNs where features are manually constructed. On the contrary, PhysNet is an end-to-end data-driven model. It is clear that ANI outperforms all the models in most of the experiments. The PhysNet model, which takes only distances and atom types as input without requiring any additional hand engineering, is not far behind ANI. ANI and PhysNet show promising results in producing smooth and accurate PES and can perform geometry optimization. While deep neural network have demonstrated great potential in achieving accurate energies, systematic benchmarking of these are necessary for wider applicability and transferability.

NNPs can accelerate the process of property prediction. The question arises can they also help in accelerating material design and discovery. In the next chapter i.e. chapter 4, we develop an energy predictor known as DART. DART is inspired by NNPs, we show how DART can be used to predict the energy of gallium clusters and thereby accelerate the work flow of identifying unique low energy isomer of gallium clusters. In chapter 5, we show how one can accelerate the process of generating unique low energy isomer of gallium cluster using RL-based model MeGen. Our MeGen model uses DART as a reward function in the RL setting.

Chapter 4

DART: Deep Learning Enabled Topological Interaction Model for Energy Prediction of Metal Clusters and its Application in Identifying Unique Low Energy Isomers

4.1 Introduction

Clusters are a bridge between two very well-understood extremes, i.e., the atoms and bulk. [127, 128] Clusters have been researched to develop a fundamental understanding of systems at a length scale where addition or subtraction of even one atom affects their physicochemical properties. [129, 130, 131, 132, 133] Computationally, structure generation techniques like random sampling, [134] genetic algorithm, [135, 136] minima hopping, [137, 138, 139] etc., are combined with geometry optimization DFT algorithms to obtain low energy/stable structures. The hunt for such stable structures is greatly limited by the computation expense involved. And hence, combining traditional approaches of energy prediction with machine learning is one such recent trend. [140, 141, 142, 143, 144, 145, 146]

Machine Learning techniques are finding increasing application to problems of materials science, right from exploring suitable materials and structures for a desired property dependent application to digging out hidden patterns in the ML datasets.[147, 148, 149, 150, 151] The power of ML to assist domain experts with insights from vast datasets has proven to be of immense promise. [145, 152, 153, 154] An important factor that lies at the heart of any ML problem is accurate "representation of the data". And hence, design of accurate "descriptors" or "features" for various class of materials/compounds is still an active area of research. Various approaches have been developed for engineering features that give a fair representation of cluster structures.[146, 149, 155, 156, 157, 158, 159, 160] The most direct and crude form of descriptor to represent a cluster structure are the Cartesian coordinates. But their direct application for energy prediction is severely restricted due to lack of rotational and translational invariance.

Symmetry functions that depend on interatomic distances and angles are yet another successful class of descriptors applied to various systems like bulk, clusters, and molecules for

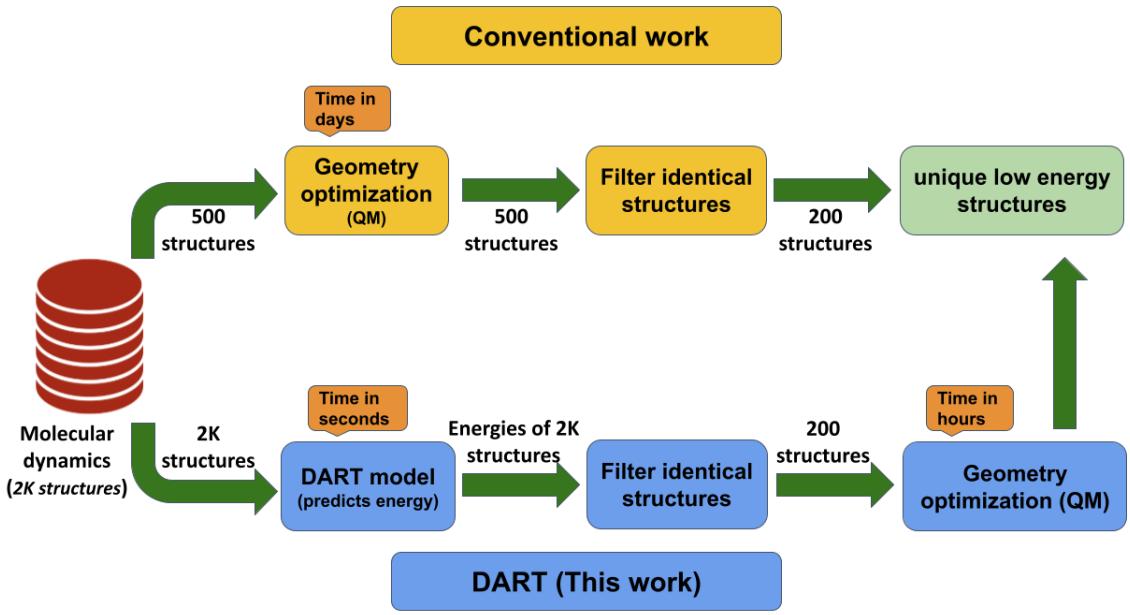


Figure 4.1: Work flow to identify ground state Gallium clusters using molecular dynamics simulations and Quantum Mechanics geometry optimization.

prediction of formation energies.[161, 162, 163, 164, 165, 166] These approaches mainly involve accurate mapping of local atomic environment to describe the structure and hence predict their energies. Descriptors like the Coulomb Matrix (CM), [164] Smooth Overlap of Atomic Positions (SOAP), [165] Atom Centered Symmetry Function (ACSF), [161] Partial Radial Distribution Function (PRDF), [167] and so on have been developed and applied to problems of finding structural similarity and also structure energy predictions. These functions, while they describe the structure to a very reasonable extent and are accurate for energy predictions, are also complex in nature, require fitting of multiple parameters and need energy minimized structures for predicting energies.[168]

Recently, graph-based representations for molecules and materials is also finding applications in property/energy prediction. [169, 170] Interatomic bonds are represented as edges and atoms as nodes of a graph. Jensen et al., developed graph based features for atomic clusters for prediction of various molecular properties.[169] They showed that they could predict octanol solubility of molecules with as low as 200 training examples using appropriate molecular representation and MAE of 0.4 $\log_{10}$ molar units. Further, the author also acknowledges the lack of precise 3D positional information that usually graph based descriptors encounter and hence limiting their prediction capacity. While, graph based representations are very simple to understand, they come at the cost of prediction accuracy as they miss out on structural details which may be of interest during energy/property predictions. [169, 171] In another study by Wolverton et. al., they demonstrate representation of crystalline compounds derived from

Voronoi tessellations of the structure. [170] Their model outperformed CM and PRDF based methods with an MAE of 2.03 kcal/mol (88 meV/atom), at a training set size of 30,000 entries. They design descriptors in a way such that they are insensitive to the choice of unit cell or even the volume of unit cell. Their descriptors are a step ahead in the direction of reducing independence of descriptors on atomic coordinates of a structure. Hence, developing descriptors that are easy to design/understand yet accurate for energy predictions is a topic that is still under research.[172]

In this work, we propose a novel workflow for energy prediction of metallic clusters as shown in Figure 4.1. For this purpose, we introduce, DART (Deep learning-enabled topologicAl inteRacTion) model, which is designed based on the principle that energy is a function of atomic interactions. To model these interactions, we developed an interaction block which is inspired by SchNet.[112] Contrary to other models that use structural information, DART uses features that capture topological/connectivity information to predict metallic clusters energies. As the descriptor consists of just the atom counts in each shell, it captures topological information. Other models such as High-Dimensional Neural Network (HDNN)[85] uses Behler-Parinello Symmetry Function (BPSF) as a feature vector that encodes structural information. DART with only topological information is able to reach Mean Absolute Error (MAE), which is very close to HDNN, which requires structural information. Our model is able to learn physically meaningful atomic interactions as well as distinguish between core and surface atoms. It is important to realise that even though we have compared our model with a neural network potential such as BPSF-HDNN, our model is not designed with the intention to learn PES but with the sole purpose of accelerating the workflow of finding low energy clusters (local minima) and increasing the possibility of finding ground state structure (global minima). If exploration of PES is the goal, one needs much more fine-grained feature vector. Such a feature vector should be able to encode minute structural changes in the cluster and the model should be able to learn energy as function of atomic interactions for exploration of PES. Further the model needs to be trained on large amount of data set which has sufficient exploration of chemical and configurational space.

In summary, our main contributions are as follows:

1. We introduce a new dataset called **G**allium **N**eutral **C**lusters, GNC_31-70 dataset. The dataset comprises of optimized, unique low energy isomers of gallium clusters with size ranging from 31 atoms to 70 atoms and their binding energies.

2. We introduce a novel feature vector called Topological Atomic Descriptor (TAD). TAD is designed to capture topological information to identify low energy clusters.

3. We propose DART model, to accelerate the workflow of identifying low energy structures and increase the probability of finding unique low energy structures.

4.2 Methods

This section is divided into four sub-sections. In the sub-section 4.2.1 we describe the generation of the new dataset called **G**allium **N**eutral **C**lusters, GNC_31-70 dataset. We introduce a novel feature vector, namely Topological Atomic Descriptor (TAD) in sub-section 4.2.2. TAD encodes the atomic environment information of a particular atom using the atom count information of its neighboring atoms. In sub-section 4.2.3, the building blocks of DART and its complete architecture which consists of five Multi-layered perceptron namely MLP_f, MLP_J, MLP_K, MLP_L, and MLP_{int} is described. DART uses TAD to predict the energy of gallium clusters. In sub-section 4.2.4 we describe the training procedure and mention other computational details. The code for this work, along with examples, is available at https://github.com/devalab/DART.

4.2.1 Dataset

The dataset comprises of optimized, unique low energy isomers of gallium clusters with size ranging from 31 atoms to 70 atoms and their binding energies. All the calculations were carried out within the Kohn-Sham formulation of DFT. Projector Augmented Wave potential [173, 174] was used, with Perdew–Burke–Ehrzenhof (PBE) approximation [175, 176]for the exchange-correlation and generalized gradient approximation, as implemented in planewave, pseudopotential based code, *VASP*.[177, 178, 179] We begin by optimizing the previously published geometries for neutral Ga_n clusters (size n = 31 to 70). [180, 181]

The dataset has total of 6851 structures and are shared as a Supplementary file and is also available at `https://figshare.com/s/9808d756f107e8fd6c69`. See Figure 4.2 for distribution of number of isomers across cluster size ranging from 31 to 69 atoms. We create train, validation and test dataset by randomly choosing structures from the cluster size from range 31 to 69. Train:valid:test split of 80:10:10 was used which gives 5265 structures in train set, 656 structures each in validation and test sets. We also created another data set called Ga-70 data set which has 285 structures all having 70 Gallium atom. Ga-70 dataset was created to demonstrate our models transferability in terms of predicting energy for Ga-70 clusters by training the model on Ga clusters of size 31 to 69.

4.2.2 Descriptor

We use TAD as the feature vector. As the name suggests, TAD encodes topological information of the cluster. TAD is designed to capture topological information and hence does not require exact structural information to identify low energy clusters; we show this in the results section. As seen in Figure 4.3, TAD encodes the atomic environment information of a particular i^{th} atom A by dividing the neighboring atoms into three different shells viz. J-shell, K-shell, and L-shell. These shells around i^{th} atom A are created using distance cutoffs. Here, we have

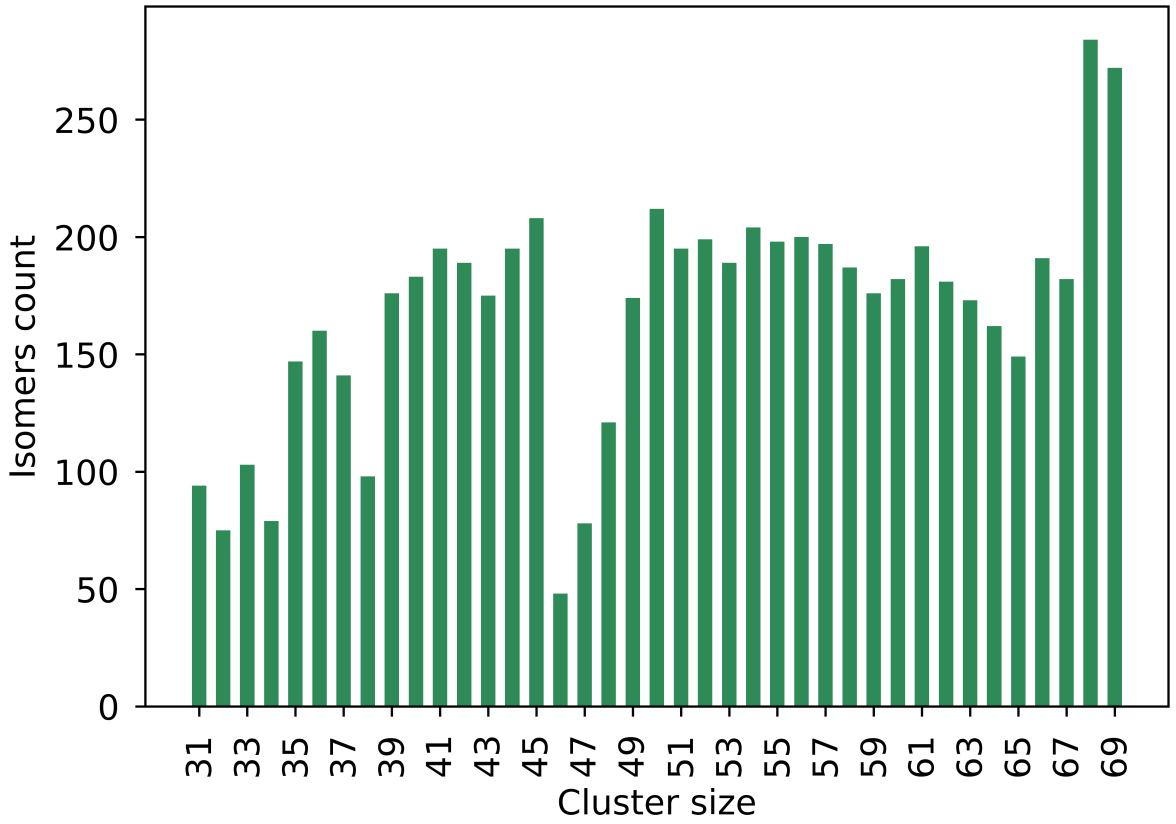


Figure 4.2: Shows distribution of number of isomer w.r.t Gallium cluster size.

used distance cutoff less than 3.49 Å for J-shell, 3.49 Å to 6.3 Å for K-shell, and beyond 6.3 Å is L-shell. The rationale behind choosing the distance cutoffs is based on the Ga-Ga pairwise distance distribution as shown in Figure 4.4. We count the number of atoms in each of these shells to create TAD. Hence, TAD is a very simple and elegant descriptor which in some sense tries to encode structural information by dividing the connectivity information using distance cutoffs.

4.2.3 DART (Deep Learning Enabled Topological Interaction)

The architecture of DART is shown in Figure 4.5, it uses TAD as the feature vector to predict the energy of a single i^{th} atom. It sums the energy of all the atom's to obtain clusters energy as shown in Eq 4.1, where E_i is i^{th} atom energy and $i = 1..N$ where N is the total number of atoms in the cluster. Figure 4.5 shows TAD representation of i_1 atom in twelve atom cluster. The remaining eleven atoms belong to either of the three shells based on the distance cutoff from the focal atom (i_1 in this case) whose energy is to be predicted. DART consists of five multilayered perceptrons (MLP), four for atomwise feature refinement, and one

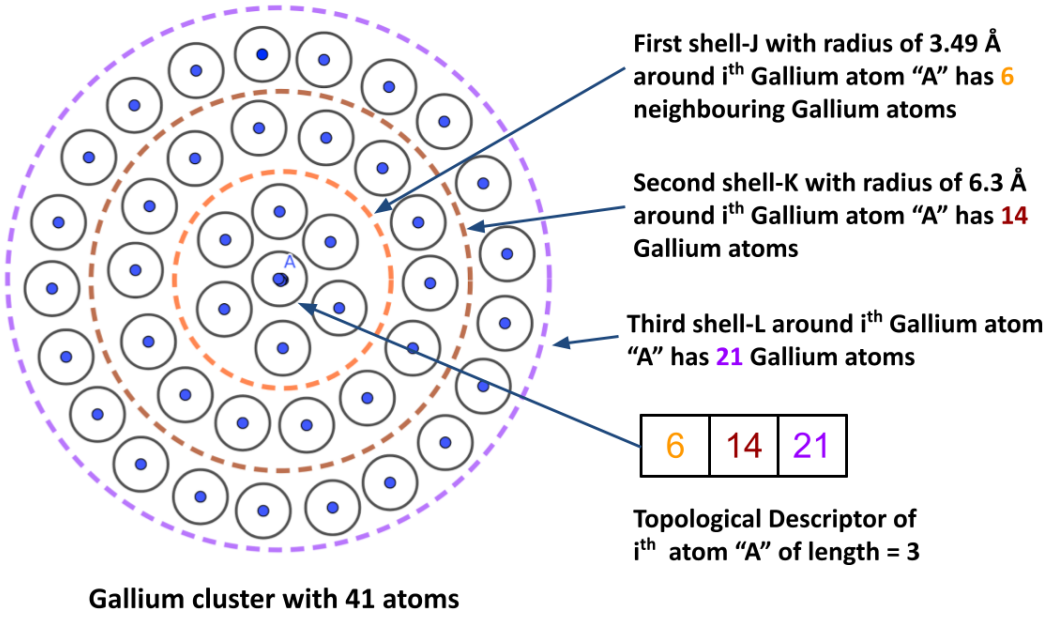


Figure 4.3: Shows topological descriptor of atom A in cluster of 51 Gallium atoms.

MLP is to model/refine atomwise interactions. As in equation 4.2 where W are the weights and b is the bias of MLP, the feature vector x_f of the focal atom i_1 is passed through MLP_f.

$$E_{Tot} = \sum_{i}^{N} E_i \tag{4.1}$$

$$Y_f = W_f x_f + b_f \tag{4.2}$$

The feature vector of each atom in the J^{th}-shell is fed to MLP_J as shown in equation 4.3 and element-wise summation of all the atoms provides us with J^{th}shell features. Similarly, we get K^{th}-shell and L^{th}-shell features as given in equation 4.4 and 4.5.

$$Y_J = \sum_{j} W_J x_j + b_J \tag{4.3}$$

$$Y_K = \sum_{k} W_K x_k + b_K \tag{4.4}$$

$$Y_L = \sum_{l} W_L x_l + b_L \tag{4.5}$$

Further, we do an element-wise summation of features of the focal atom with the rest of the shells to get $Y_{f_{Total}}$ as in equation 4.6.

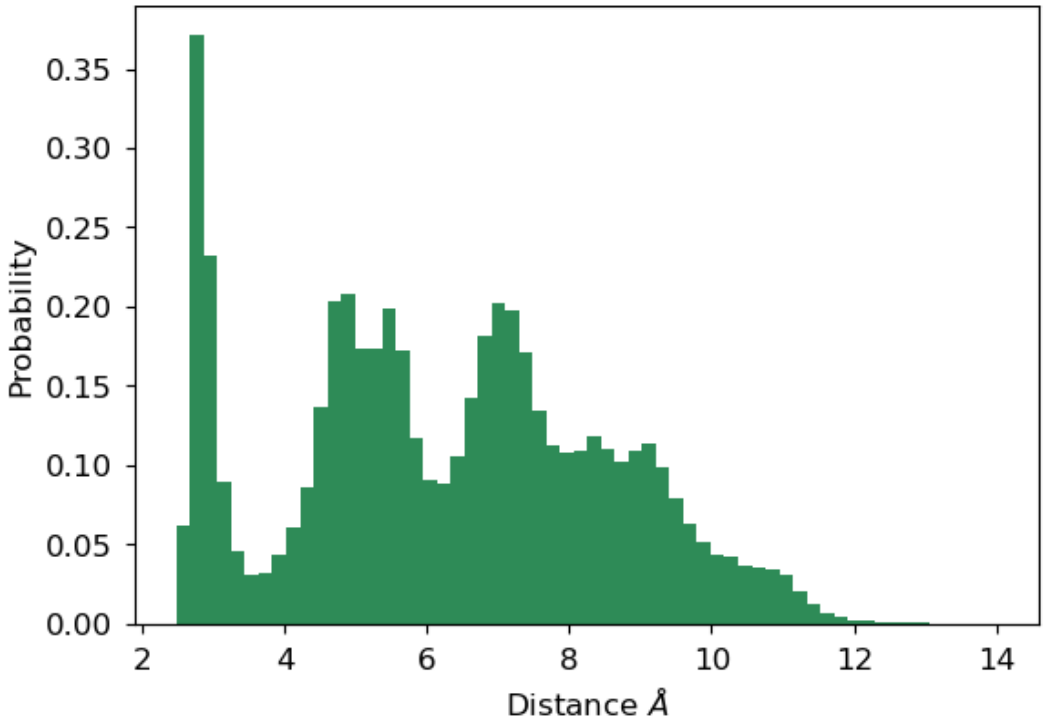


Figure 4.4: Shows Ga-Ga pairwise distance distribution.

$$Y_{f_{Total}} = Y_f + Y_J + Y_K + Y_L \tag{4.6}$$

Which is then feed to MLP_{int}, where int stands for interaction, which further refines these features to model atomic interactions. The final output of the MLP_{int} is the focal atom's energy, see equation 4.7.

$$E_f = WY_{f_{Total}} + b \tag{4.7}$$

4.2.4 Training the models

DART model is implemented in PyTorch.[182] Before training the model, all the learnable parameters are initialized using Kaiming initialization.[183] We have 2-hidden layers in MLP_f, MLP_J, MLP_K, and MLP_L and 4-hidden layers in the interaction block MLP_{int}. The network architecture plays a major role in the performance of the DART model. Too small of a network has reduced flexibility which causes poor performance; on the other hand, larger networks tend to overfit the data leading to bad generalization, especially on small datasets. Table 4.1 gives the dimensions of DART model. After initialization, we train the DART model using a batch size of 32 to predict energy by minimizing the Mean Absolute Error (MAE) between predicted and actual energies using Adaptive Moment Estimation (ADAM) optimizer with an

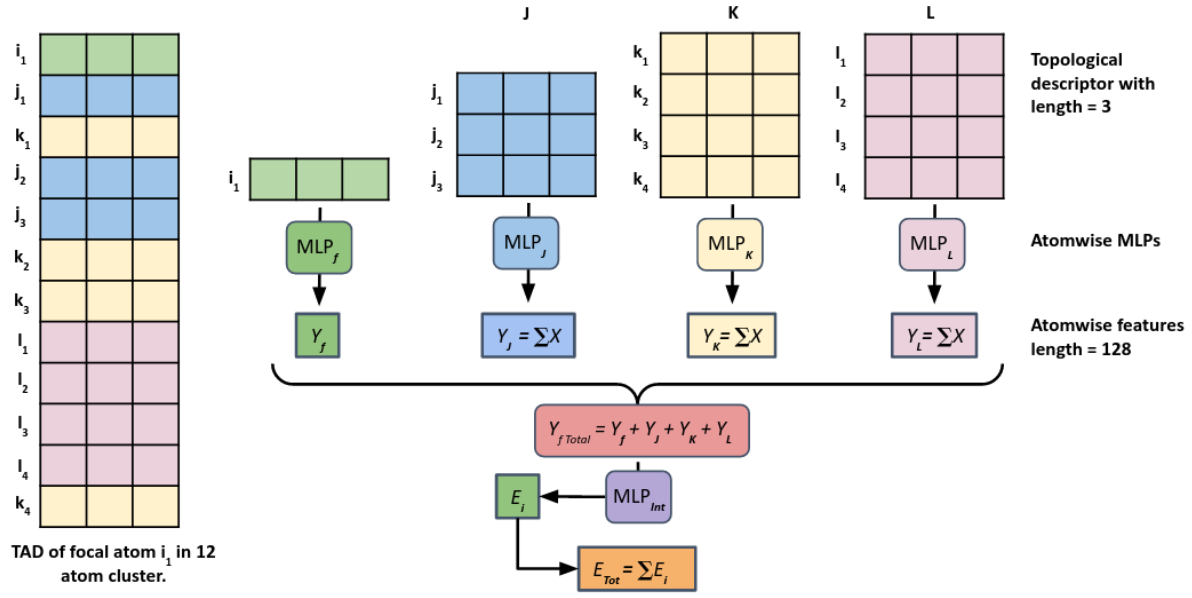


Figure 4.5: On the left is topological descriptor of 12 atom dummy cluster. On right is DART architecture, where energy of the i^{th} atom of interest is predicted, with their corresponding atoms in J, K, and L shells, for i = 1..N respectively, where N is total number of atoms in the cluster. There are five different types of multilayered perceptron one each for focal atom, J-shell, K-shell, L-shell and interaction block are MLP_f, MLP_J, MLP_K, MLP_L, and MLP_{int} respectively. It should be noted that the number of MLP_J depends on number of atoms in J-shell.

initial learning rate of 0.001, other ADAM parameters set to $\beta_1 = 0.9$, $\beta_2 = 0.999$. Learning rate is multiplied by 0.1 after reaching a plateau with the patience of 25 epochs and eps = 10^{-9}. All the intermediate layers were activated using the Continuously Differentiable Exponential Linear Units (CeLU) activation function. We stop training by setting the early stopping learning rate to 10^{-8} to avoid overfitting. Training of HDNN-BPSF is similar to DART model with a batch size of 32, $L1$ loss function except weights are updated using ADAMW (Adaptive Moment Estimation algorithms wiith decoupled weight decay regularization), and bias is updated using Stochastic gradient descent (SGD). All hyperparameter values are obtained after hyperparameter optimization for both models.

4.3 Results

4.3.1 BPSF-HDNN

Behler-Parrinello Symmetry Function (BPSF) descriptor encodes the atomic environment using 3D geometry. In contrast, we developed a topological descriptor that encodes the atomic environments using atom counts as described in section 4.2.2. The topological descriptor captures the topological information as compared to BPSF, which captures structural information. Hence, BPSF with the HDNN model is considered a hard limit that a model using

Module	Dimensions
MLP_f	3:128:128
MLP_J	3:128:128
MLP_K	3:128:128
MLP_L	3:128:128
MLP_{int}	128:256:128:32:1

Table 4.1: Number of hidden layers input size, and output size of each module is given. MLP_f is multi-layered perceptron of focal atom f, similarly MLP_J is for J-shell, MLP_K is for K-shell, MLP_L is for L-shell, and MLP_{int} is for interaction block. All these values are obtained after hyperparameter optimization.

topological descriptors cannot out perform. We compare the DART model, which uses TAD, with the BPSF-HDNN model, which uses 3D structures. BPSF-HDNN has an architecture of 640:128:128:64:1, and we used a radial cutoff of 15 Å and angular cutoff of 6.5 Å to generate atomic environment vector (AEV). The radial and angular cutoffs dictate the distance neighboring environment should be probed. We used $\eta_r = 16.0$, $\zeta = 32.0$ and $\eta_a = 8.0$. Sixty-four evenly spaced radial shifting parameters were used for the radial part, and a total of twenty-four radial and twenty-four angular shifting parameters were used for the angular part. The total length of AEV is $64(radial)+24*24(angular) = 640$.[84, 85] We get MAE of 2.4 kcal/mol, Root Mean Squared Error (RMSE) of 3.03 kcal/mol on the test set. As expected, BPSF performs very well in predicting the binding energy of Gallium clusters.

4.3.2 Accuracy and Transferability of DART

Model	Testset (kcal/mol)	
	MAE	RMSE
BPSF-HDNN (structural)	2.4	3.03
DART (topological)	3.59	4.55
DART without MLP_K	4.06	5.16
DART without MLP_L	3.93	5.05
DART without MLP_K & MLP_L	4.29	5.45
DART without MLP_{int}	12.33	16.40

Table 4.2: Behler Parinello Symmetry Functions (BPSF) - HDNN uses structural information whereas DART uses topological information. Test set includes 655 clusters.

DART model has been applied on Gallium clusters dataset introduced in section 4.2.1 to predict binding energies by using topological information. After training DART on 80 % of data .i.e. 5256 clusters, we achieve MAE of 3.59 kcal/mol and RMSE of 4.55 kcal/mol on a test set that has 656 clusters with size ranging from 31 to 69 atom clusters. The largest Gallium clusters in the train set contain 69 atoms; hence we also test the model on the Ga-70 cluster to check its transferability to larger clusters that the model has not seen during training. When

trained on $N = 31$ to $N = 69$ atom clusters and tested on $N + 1 = 70$ atom clusters we get MAE of 4.71 kcal/mol and RMSE of 5.96 kcal/mol.

In another experiment, we validate our model on three test cases each having different test-set to show the robustness of our model on different cluster sizes. In first test case we train our model on Ga-31 to Ga-70 excluding Ga-46, 57 and 60. These sizes (46, 57 and 60) of Ga clusters will be used as test-set. In second test case our model is trained on Ga-31 to Ga-70 excluding Ga-46, 57, 66 and excluded sizes (46, 57, 66) are considered as test-set. Similarly for third test case we train model on Ga-31 to 70 excluding sizes (57, 60, 66) of Ga clusters. These excluded sizes (57, 60, 66) of Ga cluster are used as test-set. Performance of DART model on each of these test cases is summarized in Table 4.4, it can be seen that DART models performance is consistent across all the test cases. TAD and DART can be extended to other metallic clusters.

Results for all the test cases for each Gallium cluster individually is given in Appendix B Table B.1, B.2, and B.3. As it is evident from the results that the model is robust, the RMSE values for Gallium-57, 60 and 66 are consistent across three experiments. There are small structural changes in the core-shell of Ga-46 cluster which leads to large change in energy whereas these small structural changes are not fully captured by our descriptor (TAD) hence we observe higher values of RMSE for Ga-46 which is a magic cluster (Table B.1 and B.2).

We have also tried various flavors of our DART model, and their performance on the test set is summarized in Table 4.3. Where MLP_{ijlk} represents four different multi-layered perceptrons each for i, j, k, and l. MLP_{int} represents interaction block. From the results in Table 4.3 it is evident that interaction block is very important, and reducing the number of hidden layers in the interaction block increases the MAE to 4.28 kcal/mol. Also, the results in Table 4.3 show that decreasing the number of hidden layers does not have a drastic impact on the performance of the DART model. When compared with out best model (MAE=3.59 kcal/mol), the increase in MAE is less than 1 kcal/mol when we tried with reduced hidden layers ($MLP_{ijlk} = 3 : 64 : 64$ and $MLP_{int} = 64 : 128 : 128 : 32 : 1$). Hence having a simple feature vector does not always necessitate the need to have many hidden layers in the model.

4.3.3 Importance of each shell and interaction block

To further show the importance of each of the modules in the DART model, we perform a few experiments (reported in Table 4.2) . In these experiments, we train our model by excluding or "switching off" each of these modules individually or in combination to see how the absence of each of these modules affects the performance of the DART model. It should be noted that all these experiments are compared with the DART model (topological), which has all-shells and interaction block "switched on". In the first experiment, the MLP, which provides information about the L-shell, i.e., MLP_f, is switched off; we get MAE of 3.93 kcal/mol, RMSE of 5.05 kcal/mol on the test-set. For the second experiment, the MLP that provides information about the K-shell is switched off, which gives MAE of 4.06 kcal/mol, RMSE of 5.16 kcal/mol on the

DART Model	Testset (kcal/mol)		Time
	MAE	RMSE	Mins
$MLP_{ijkl} = 3:64:64$, $MLP_{int} = 64:128:128:32:1$	4.28	5.51	15
$MLP_{ijkl} = 3:64:64$, $MLP_{int} = 64:256:128:32:1$	3.75	4.86	16
$MLP_{ijkl} = 3:64:96$, $MLP_{int} = 96:256:128:32:1$	4.02	5.04	16
$MLP_{ijkl} = 3:64:128$, $MLP_{int} = 128:256:128:32:1$	3.99	5.13	18
$MLP_{ijkl} = 3:96:128$, $MLP_{int} = 128:256:128:32:1$	3.92	5.04	23

Table 4.3: DART performance with varying number of hidden layers. MLP_{ijlk} represents four different multi-layered perceptron each for i, j, k and l. MLP_{int} represents interaction block. Test set includes 655 clusters. Total time required to train the model in minutes on Nvidia GeForce RTX 2080 Ti GPU.

Model	DART (values in kcal/mol)		
	MAE	RMSE	# structures
Test case 1 - Test set: (Ga-46, 57, and 60)	4.77	6.16	427
Test case 2 - Test set: (Ga-46, 57, and 66)	4.76	6.02	436
Test case 3 - Test set (Ga-57, 60, and 66)	4.34	5.58	570

Table 4.4: Performance of DART model on three different test cases each having different test-set to show the robustness of our model. Last column gives the number of structures present in the test-set. All values are in (kcal/mol).

test-set. For the third experiment, the MLP_K and MLP_L, which provide information about the K-shell and L-shell, respectively, are switched off, which gives MAE of 4.29 kcal/mol and RMSE of 5.45 kcal/mol on test-set. Finally, in the fourth experiment, the interaction module MLP_{int}, which learns the physically meaningful interaction between the atoms, is switched off, which gives MAE of 12.33 kcal/mol, RMSE of 16.40 kcal/mol on test-set. It is evident that when compared to DART (topological), which achieves MAE of 3.59 kcal/mol and RMSE of 4.55 kcal/mol on test-set that switching off MLP_K gives a larger MAE value than the absence of MLP_L hence K_{th}-shell is more critical than L_{th}-shell. Switching off the interaction module has the most significant effect on the DART model's performance; MAE values go as high as 12.33 kcal/mol. Hence, we conclude that the interaction module MLP_{int} is critical in the DART model, which helps learn the physically meaningful atomic interactions.

4.3.4 Model learns to distinguish between core and surface atoms

This exercise shows our model's ability to distinguish between the Gallium clusters' core and surface atom. The model learns this distinction from our descriptor TAD, which encodes the chemical environment information of each atom in the cluster. Which further proves that our descriptor TAD encodes the chemical environment information of each atom within the cluster, which is sufficient for the model to distinguish between core and surface atoms. It should be noted that not all atoms contribute equally towards the binding energy of the cluster. The core atoms will have more contribution towards binding energy as compared to the surface atoms. In Ga-70 cluster, atoms within 3.49 Å from the center of mass (CoM) of the cluster are considered as core atoms, and atoms beyond 6.3 Å from CoM are considered as surface atoms. We extract the features from the third hidden layer of interaction block MLP_{int}. The third hidden layer of the interaction module MLP_{int} is just before MLP_{int} outputs the predicted atomic energy. We perform t-Distributed Stochastic Neighbor Embedding (t-SNE) on the extracted features. The t-SNE plot of the previously extracted feature in Figure 4.6 shows a clear distinction of core and surface atoms.

4.3.5 Identification of stable isomers from molecular dynamics data

This section demonstrates the model's ability to identify/filter unique low energy isomers from unseen structures taken from molecular dynamics simulation trajectories. Three randomly selected sizes (46, 57, 60) of Ga clusters were taken from molecular dynamics simulation. They were used as a test set for the DART model trained on geometry optimized data from size 31 to 70, excluding size (46, 57, and 60) of Ga cluster. Molecular dynamics test set consists of 1919, 1051, and 2596 structures of Ga-46, Ga-57, and Ga-60, respectively. The model was able to predict energy values for all 5566 structures within seconds. Firstly, the predicted energies were sorted and 100 lowest energy structures were picked for DFT optimisation. Upon optimisation it was observed that the DART model had correctly identified the ground state geometry from the vast dataset on which it was tested. And hence, greatly reducing the load on DFT calculations. Further, the DART model was also used to identify the different low energy isomers for each size. Conventionally, about 25-30 % structures picked for each size from molecular dynamics would be chosen to perform further DFT optimizations, which would in turn yield roughly 100-150 unique structures. This search, i.e., the number of DFT optimizations, could be significantly reduced with the application of the DART model. Since the model was trained on optimization data, energy predictions were closer to optimized geometry values than MD. Moreover, it was possible to significantly reduce the number of DFT optimizations while maximizing the probability of finding unique low energy structures. There were two possible approaches to identify unique isomers out of the existing MD data; i.e. structures could either be segregated into bins of varying descriptors in the feature space or bins of

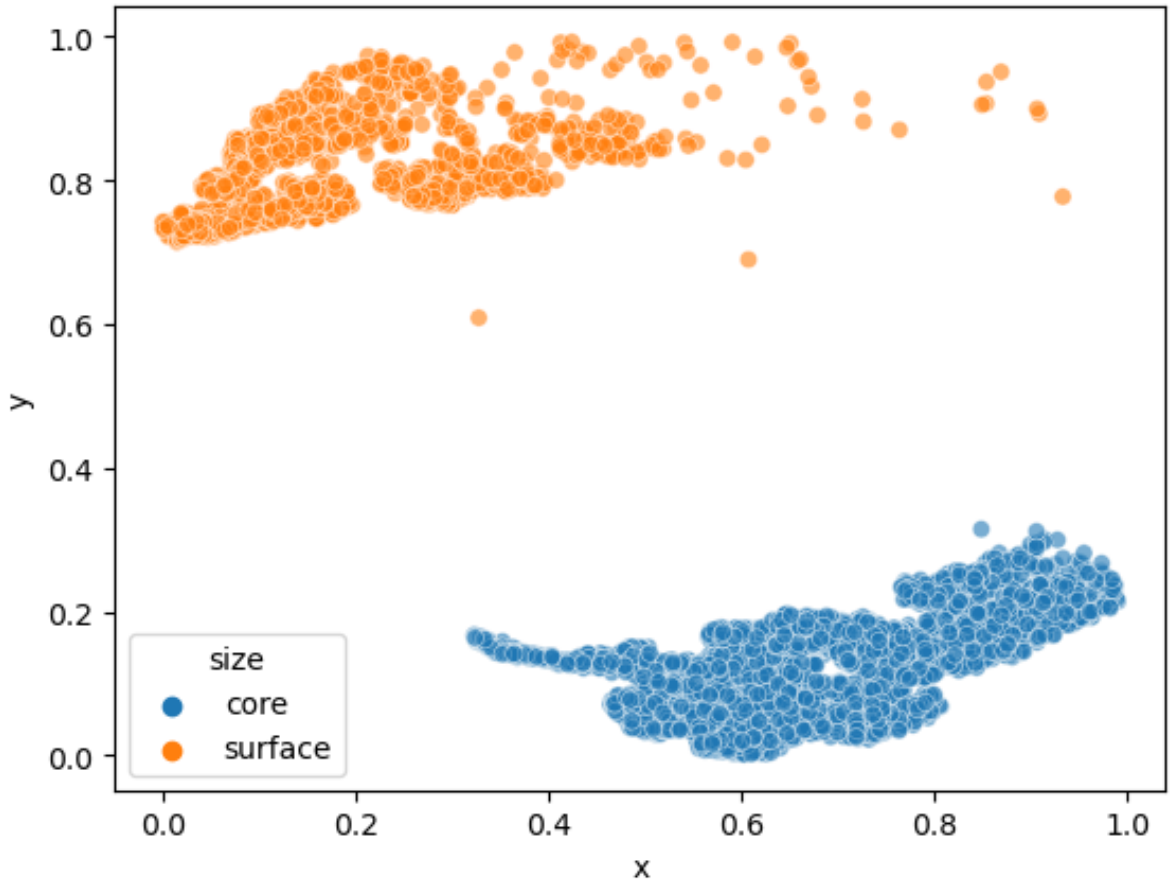


Figure 4.6: t-Distributed Stochastic Neighbor Embedding (t-SNE) of the feature extracted from the 3rd hidden layer of interaction module during testing of Ga-70 dataset.

varying ML predicted energy values. Optimisation of one structure from each bin would in turn identify the unique isomers. However, given that TAD counts the coordination of each atom in its neighborhood within three shells, classification based on descriptors would lead to very broad bins of structures. This is turn would lead to classifying similar structures of clusters belonging to different local minima's into one bin and missing out on many unique isomers. Hence, structures were segregated into bins of varying ML predicted energy values. About 100 structures were chosen (one from each energy bin) and optimized using DFT to identify unique low energy geometries. It was noted that out of 93 DFT optimizations performed for each of the three cluster sizes, nearly 97 % of the structures turned out to be unique. Hence, we note that DART not only predicts energy but also could be employed to identify unique low energy isomers from a large number of structures generated from molecular dynamics simulations.

4.4 Conclusion

In this study, a deep learning model, namely DART, developed using interaction block, capable of modeling atomic interactions using topological descriptor (TAD) is reported. DART is significantly more efficient than the conventional work-flow for identifying ground state isomer in terms of time and computational resources. A novel feature vector called Topological Atomic Descriptor (TAD), which effectively encodes the clusters topology, is used to train the DART model. A deep neural network DART model exhibits its ability to predict energy of Gallium cluster, demonstrating the trained network's robustness and TAD's adequacy as a feature vector compared to symmetry functions. The DART model generalizes reasonably well during extrapolation on larger gallium cluster size, but we see that the error increases as we increase the cluster size. The interaction module is crucial to mimic atomic interactions. Examining the features extracted from this module indicates that the model has learned the underlying chemistry by its ability to differentiate between core and surface atoms. Furthermore, DART model can be extended to any metal cluster and can be modified to be used on nanoalloys.[184] Also, DART model can identify unique low energy structures from a corpus of structures obtained from MD simulations, thereby reducing the number of QM calculations manifolds. This method successfully predicts clusters energy and can identify unique low energy isomers. DART can possibly be used with methods like CALYPSO[185] or stochastic kicking[186] to accelerate the process of finding the low energy isomer by predicting multiple low-energy candidates by means of a computationally cheap model (DART), and the fine refinement of the lowest energy ones by means of an accurate but computationally expensive method (DFT).

Next objective is to develop a model which can learn and generate three-dimensional structures of the clusters rather than just identifying the low energy isomers. In the next chapter i.e. chapter 5, we develop an RL-based model known as MeGen which can generate $n + 1$ atom gallium cluster given a n atom cluster. We show that MeGen, not only accelerates the workflow of generating low energy isomers of gallium clusters but also hits the ground state (GS) structure of this $n + 1$ atom cluster.

Chapter 5

MeGen - Generation of Gallium Metal Clusters Using Reinforcement Learning

5.1 Introduction

Metal clusters are aggregates of atoms, and they exhibit substantially different properties from their corresponding bulk metals.[187, 188] They also differ from the nano-particles, and their extreme size sensitivity reflects in almost all their properties like melting point, reactivity, etc.[189, 190, 191] Clusters, as opposed to their bulk counterpart, have numerous local minima, and the number of such minima increases exponentially with the size of the cluster. It has been demonstrated that many properties of these clusters depend on their Ground State (GS) structure. Hence, there has been a constant drive to develop better algorithms for the search of GS and energetically low-lying structures.[192] For example, random search provides an unbiased configuration sampling[193] while genetic algorithm combines and propagates useful structural markers.[194, 195] Basin-hopping[196, 197] represents an efficient technique for escaping from local minima and mapping the PES and particle swarm optimization[198, 199] relies on the population information for navigating the energy landscape. However, when these techniques are combined with ab-initio methods for the description of inter-atomic interaction, they become computationally expensive and result in an insufficient sampling of PES.[200, 201] On the other hand, classical potentials fall short of describing the problem with enough accuracy. Further, the search of atomic structure will be biased with the range or form of the potential used.[202] Hence faster and more reliable models need to be developed for searching the GS and low-lying isomers of metallic clusters.

Recent machine learning applications have accelerated the search for new molecules with specific desired properties. Much progress has been made in developing deep generative models for 2D molecule generation. Such as Simonovsky and Komodakis [53], Kusner et al. [54], Gómez-Bombarelli et al. [203], Jin et al. [204], Dai et al. [205] used VAE and its variants for generation of molecules, Guimaraes et al. [206], De Cao and Kipf [207] used GAN, Segler et al. [208]

used RNN, Popova et al. [209] used RL-RNN and Maziarka et al. [210] developed Mol-cycle-GAN for generation of molecules. Most of these studies use SMILES string representation or graph representation of molecules. SMILES have various limitations, such as not capturing molecular similarity; for example, two similar chemical structures may have very different SMILES representations, hindering VAEs from learning smooth molecular embeddings. Furthermore, methods that use SMILES to generate SMILES strings are prone to generate strings that do not correspond to a valid molecule. Hence Simonovsky and Komodakis [53], Jin et al. [204], Maziarka et al. [210] have shown that operating directly on graphs helps generate more valid molecules.

The domain of 3D structure generation is lacking in comparison to 2D structure generation. Moreover, most of the work in 2D structure generation is SMILES string or graph-based, where an atom is considered as a node, and the bond between the atom is the edge. One of the major disadvantages of string or graph-based representation is that they do not contain the inter-atomic distance or 3D information about the molecule. In contrast, chemical properties depend on the 3D structure of the molecule. Recently efforts have been made to develop a model to generate 3D structures and bias the structure generation based on chemical property. One such attempt would be to combine the method described by Mansimov et al. [55] to sample 3D conformers given a molecule with graph-based generative models to generate 3D structures in a two-step procedure. Gebauer et al. [211] developed an autoregressive deep generative model based on SchNet[111] which is capable of generating a variety of $C_7O_2H_{10}$ 3D structures which are close to equilibrium. Gebauer et al. [212] made improvements in the previous model to develop G-SchNet, which can deal with an arbitrary number of atom species. In inorganic chemistry, the 3D structure generation model is even less common than it is in organic chemistry. Notable mentions are iMatGen by Noh et al. [213], which learns latent space of inorganic structures using 3D images as input to generate crystal structures. Kim et al. [214], who is also one of the authors of iMatGen proposed a new generative model to generate crystal structures using GAN; they used a coordinate-based inversion-free crystal representation inspired by point clouds.[215]. Nouira et al. [216] developed CrystalGAN to produce stable ternary structures by using GAN. Court et al. [217] introduced a deep-representation learning autoencoder-based generative pipeline for geometrically optimized 3D crystal structures, which also predicts the values of target properties.

In this work, we propose a 3D structure generator model named as MeGen (**Me**tal cluster structure **Gen**eration). MeGen generates energetically low-lying 3D structures of Ga clusters in cartesian coordinates using Reinforcement Learning. Our model exploits the rotationally covariant state representation for 3D structure generation. We integrate this state representation into an actor-critic neural network architecture with a rotationally covariant auto-regressive policy, where the position of the atom to be placed is modeled through a flexible distribution based on spherical harmonics. The reward function is based on a fundamental physical prop-

erty (energy). We use the previously developed ML model DART[218] to calculate the energies of various conformers/isomers generated during training. Contrary to other models that use structural information, our DART model uses features that capture topological/connectivity information to predict metallic clusters energies. As the descriptor consists of just the atom counts in each shell, it captures topological information and is known as Topological Atomic Descriptor (TAD). DART model provides accuracy comparable to DFT at minimal computational cost.

Our main contributions are as follows:

1. We introduce a new dataset called as **G**allium **N**eutral **C**lusters, GNC version 2.0. The dataset comprises of optimized, low energy isomers of gallium clusters with size ranging from 2 atoms to 70 atoms and their corresponding energies.

2. We propose a model based on reinforcement learning for generation of structures in Cartesian coordinates.

3. We accelerate the workflow of generating low energy isomers of gallium clusters and in the process we also hit the GS.

4. We demonstrate that this method generates novel structural motifs along with previously reported ones and increases the probability of finding new unique low-lying isomers for clusters.

5.2 Theory and Methods

This section describes various components of the proposed framework as shown in Figure 5.1 to generate 3D structures in Cartesian coordinates for gallium clusters.

5.2.1 Dataset

In our previous work, DART is trained over GNC_31–70 data-set to predict energy for a give Ga cluster. The GNC_31–70 comprised of low energy isomers and their corresponding energies with sizes ranging from 31 to 70 atoms.[218] In this work, we added new optimized geometries of neutral Ga_n clusters (size n = 2 to 31) to data-set GNC_31–70. The new data-set called GNC version 2.0., comprises of optimize low energy isomers of gallium clusters with size ranging from 2 to 70 atoms. The dataset has total of 8166 structures of gallium cluster. This new data set GNC 2.0. is used to train the DART which is integrated as a reward function in MeGen model. All the calculations were carried out within the Kohn-Sham formulation of DFT. Projector Augmented Wave potential [173, 174] was used, with Perdew–Burke–Ernzerhof (PBE) approximation [175, 176] for the exchange-correlation and generalized gradient approximation, as implemented in planewave, pseudopotential based code, VASP.[177, 178, 179]

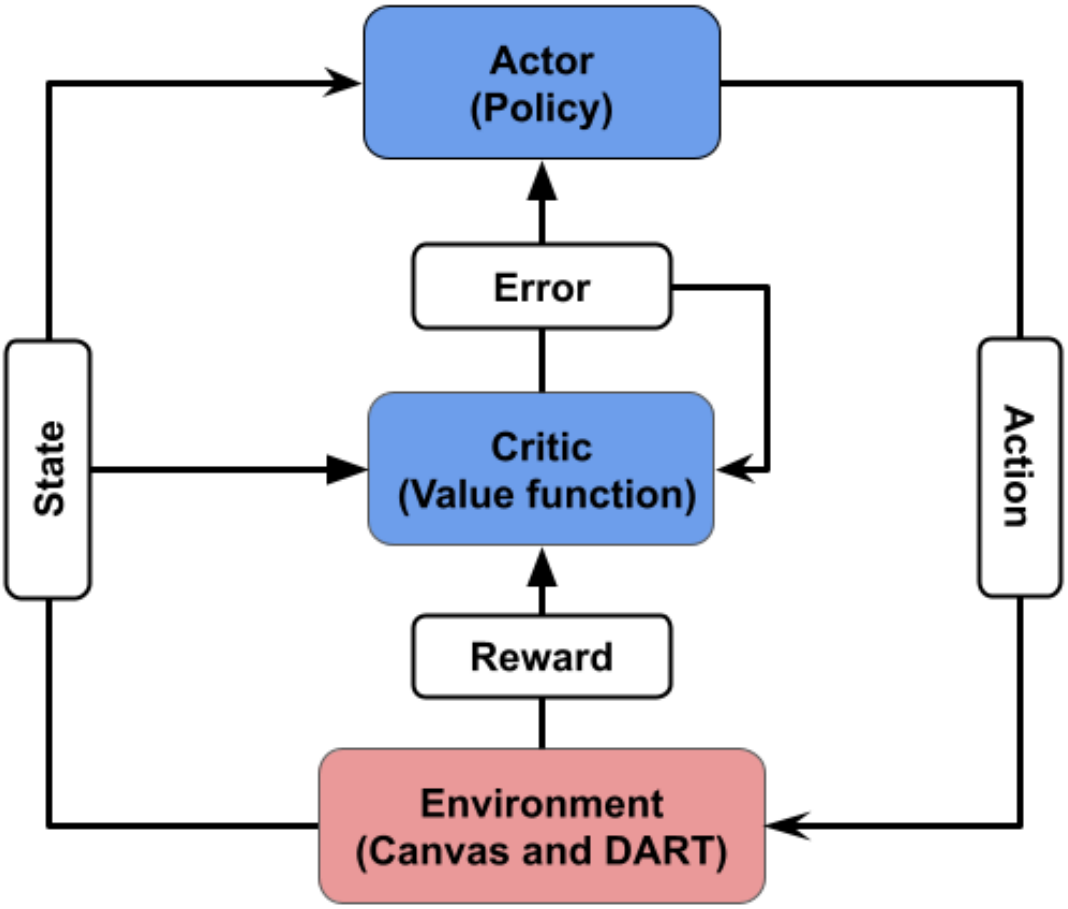


Figure 5.1: Our reinforcement learning setting to generate GS/low-energy gallium clusters. This actor-critic formulation can be used iteratively to generate $(N, N+1, N+2, ..., N+M)$ atom gallium clusters.

5.2.2 Reinforcement Learning formulation

Reinforcement learning (RL) formulations, in general, have an agent which interacts with the environment to maximize its reward. In principle, the formulation can be described as a Markov decision process (MDP).

Definition : MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}_a(s, s'), \mathcal{R}_a(s, s') \rangle$ where:

- State $s \in \mathcal{S}$, $\mathcal{S}$ is set of all possible states.
- Action $a \in \mathcal{A}$, $\mathcal{A}$ is the set of possible actions.
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow$ is the (stochastic) transition function.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$ is the reward function.

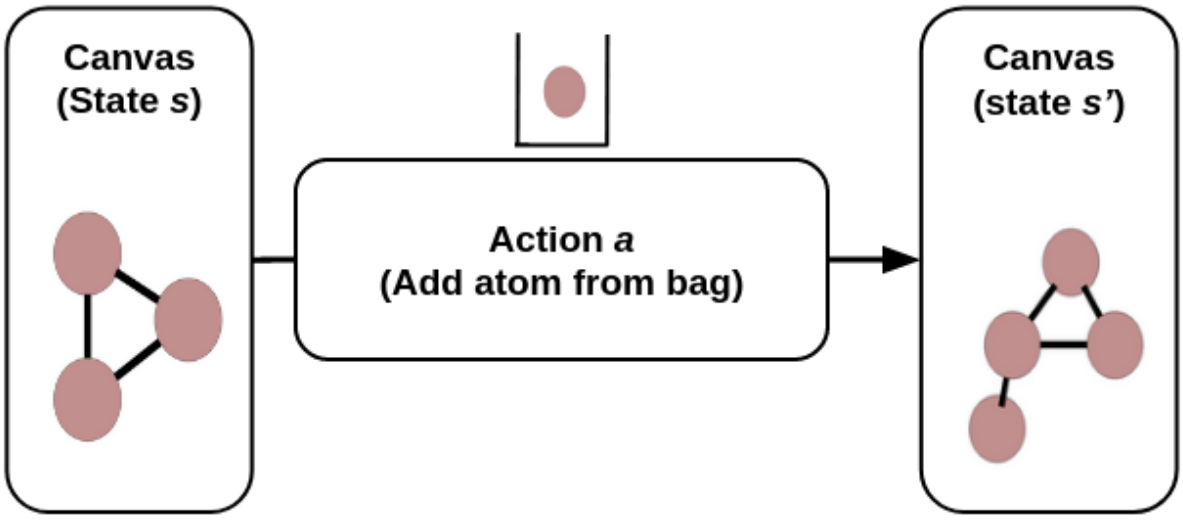


Figure 5.2: Schematic representation of Ga_4 structure generation from Ga_3 seed structure on canvas.

By solving the RL problem, we aim to learn a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{R}$, which is a conditional probability density over actions given the current state, such that the expected cumulative reward is maximized. In other words, we aim to find a mapping from states to action, called a policy (denoted as π), with maximum achievable total reward. Most often, the policy π is parameterized using a neural network.

Similar to Simm et al. [59], we begin with formulating 3D structure generation as a reinforcement learning problem. As shown in Figure 5.2, We generate gallium clusters by adding a single atom from the bag B to the existing seed structure of the gallium cluster present on the 3D canvas C. In principle, we are generating N atoms cluster from $N-1$ atoms seed cluster already present on canvas C. The idea is to allow the model to learn the process of addition of a single atom to $N-1$ atom cluster to generate N atom cluster. The reward function $R_a(s, s')$ is given in Equation 5.4 as the negative of the energy calculated using DART model.[218] This allows the the model to learn the most efficient ways of adding atoms in the presence of different structural motifs across different sizes/classes of gallium clusters to get low-energy gallium clusters. This will also allow the model to access the structural motifs, which are hard to find in some classes but easily found in others. As introduced by Simm et al. [59] to model action a using policy π, we need to model sub-action such as given a state s choose focal atom f to which new atom type e will be added at a particular distance d and orientation $\hat{x}$ as given in Equation 5.1.

$$\pi(a/s) = \pi(\hat{x}, d, e, f|s) = p(\hat{x}|d, e, f, s)\ p(d|e, f, s)\ p(e|f, s)\ p(f|s) \tag{5.1}$$

In our case Equation 5.1 gets modified to Equation 5.2. This is because we randomly choose focal atom f and atom type e of next atom is always gallium.

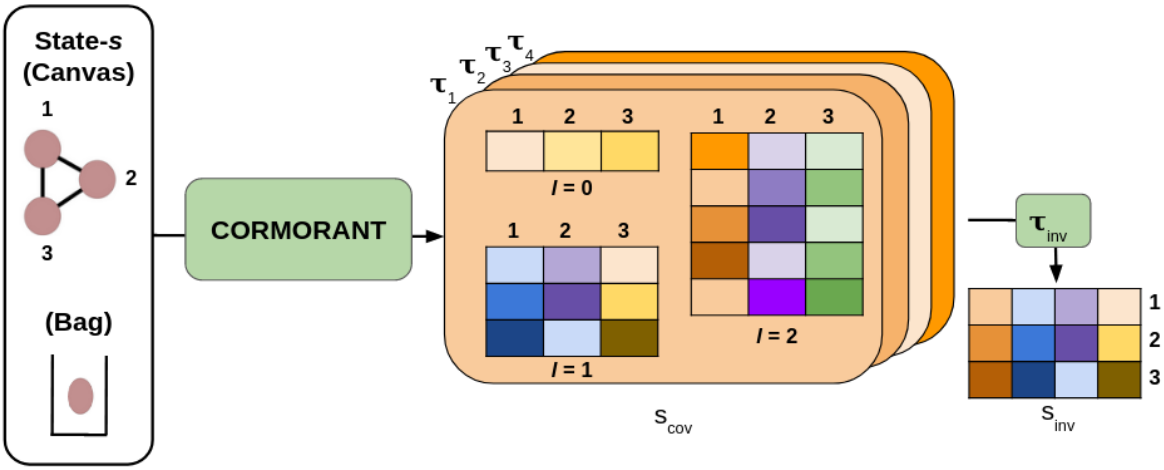


Figure 5.3: Starting from seed structures, we choose one seed structure at random. We use CORMORANT to obtain rotation-covariant (s_{cov}) and invariant(s_{inv}) state representations. $\tau_1, \tau_2, \tau_3, \tau_4$ are four channels and τ_{inv} is the combination of learnable transformation taken from Anderson et al. [219] to obtain invariants from covariant features.

$$\pi(a/s) = \pi(\hat{x}, d, f|s) = p(\hat{x}|d, f, s)\ p(d|f, s) \tag{5.2}$$

5.2.3 State representation

As shown in Figure 5.3, we concatenate a vectorized representation of the bag B with each atom on the canvas C. This concatenated representation is fed to the state embedding network CORMORANT[219], which then transforms bag B and canvas C to get a rotationally covariant representation s_{cov}. We further obtain s_{inv} from s_{cov} by applying a combination of transformations known as τ_{inv} from Anderson et al. [219]. An invariant representation s_{inv} is essential as the choice of distance d between the focal atom and the new atom must be invariant to rotation and translation. The position of the new atom w.r.t the focal atom needs to be covariant under rotation and dependents on distance d. Therefore we condition s_{cov} on distance d to preserve rotational covariance (see Figure 5.4). The number of channel τ is set to four $(\tau_1, \tau_2, \tau_3, \tau_4)$ in our work.

5.2.4 Actor-Critic Model

As shown in Figure 5.4, the choice of focal atom f is random such that there is enough exploration and an increased number of gallium clusters generated with different structural motifs. Since the bag B contains only gallium atoms, there is not much choice in choosing the atom type for the next atom e to be placed on canvas C. The choice of distance d between

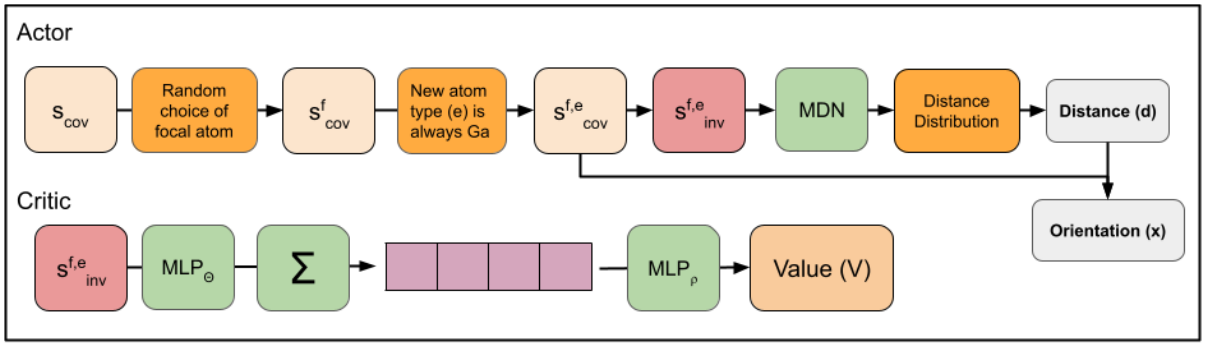


Figure 5.4: Figure of the actor-critic networks. We have the actor-network which samples the different actions based on the current policy and state. The critic-network takes the invariant representation ($s_{cov}^{f,e}$) to compute a value V.

the focal atom and the new atom should be invariant to rotation and translation; hence to achieve this, we use $s_{inv}^{f,e}$ and a mixture density network (MDN). Distance distribution between the focal atom and the next atom to be placed is modeled using Gaussian mixtures as shown in Equation 5.3, where π_m is the mixing coefficient of the m^{th} Gaussian $\mathcal{N}(\mu_m, \sigma_m^2)$. The mixing coefficients π_m, and means μ_m are predicted using a mixture density network (MDN) as shown in Figure 5.4. The standard deviations σ_m are global parameters. We clip values below zero to ensure that the sampled distances are positive.

$$p(d|f,s) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\mu_m, \sigma_m^2) \tag{5.3}$$

As shown in Figure 5.4 the orientation x of the atom depends on the chosen distance d hence we condition $s_{f,e}^{cov}$ on distance d to obtain orientation x. The critic needs to compute a value V for the state s that is invariant under translation and rotation.

5.2.5 Reward calculator - DART Model

We use DART model[218], which predicts the energy of gallium clusters as reward function $R_a(s,s') = -E_{DART}$

$$R_a(s,s') = -E_{DART} \tag{5.4}$$

We use PPO to learn the optimal policy. To increase the exploration, we randomly choose a focal atom. We further have distance-based restrictions on the placement of new atom. We calculate the three nearest neighbors (nn) of the new atom. The first nearest neighbor should be between distance 2.3 to 3.1 Å. The second nearest neighbor should be between 2.5 to 3.1 Å, and the third nearest neighbor should be between 2.5 to 5.5 Å. The rationale behind choosing the distance cutoffs is based on the Ga-Ga pairwise distance and the min-max observed for each of these nearest neighbors. For example, the first nearest neighbors min-max was 2.3 to 3.1 Å.

The Ga-Ga pairwise distances were calculated over a range of Ga clusters. If these distance restrictions are violated, we give a penalty of -10.

Removal of duplicate structures from MeGen generated structures in a non-trivial task. To capture the relative positions and orientations of the atoms in a structure we calculate bond-lengths (pairwise-distance) and angles. On the basis of these bond-lengths and angles we remove duplicate structures. Removal of duplicate structures is performed as follows:

1. Calculate all pairwise distances. We get $upd = \frac{N \times (N-1)}{2}$, number of unique pairwise-distance for N atom cluster.

2. Calculate bond cutoff as shown in Equation 5.5 where b_1^i and b_2^i are the i^{th} bonds of structure 1 and 2, respectively, if $B_c <= 0$ we consider the structures to be similar and place them in same group and pick only one structure from each group thereby removing the duplicate structure.

3. Further we remove duplicates using angles.

4. Calculate the angle (x) between the atom of interest (AOI) and its two nearest neighbors.

5. Do step 4 for all the N atoms in the cluster.

6. Sort the angles (x) and calculate cutoff angle (A_c) as shown in Equation 5.6 where x_1^i and x_2^i are the i^{th} angles of structure 1 and 2, respectively.

7. For an N atom cluster if $A_c < N$ Å we consider the structures to be similar and place them in same group and pick only one structure from each group thereby removing the duplicate structure.

$$B_c = \sum_{i=0}^{upd} (b_1^i - b_2^i) \tag{5.5}$$

$$A_c = \sum_{i=0}^{N} (x_1^i - x_2^i) \tag{5.6}$$

5.3 Results

In our previous work, we trained a DART model to predict energy for a given structure. In this work, we have developed a model which predicts structures of (N) atom clusters based on information about ($N-1$) atom clusters. Our MeGen model could be employed in two ways. Isomers for an (N) atom cluster could be generated using isomers of ($N-1$) atom cluster as seed. Alternatively, isomers for N atom clusters can be generated from scratch, i.e., starting from two atom clusters generating three atom clusters, which would be used as the

seed structures to generate four atom clusters and so on and so forth. In what follows, we demonstrate both these approaches.

Ga seed ⇒ Ga target structure	Structures from DFT based method	Structures from MeGen model
$Ga_7 \Rightarrow Ga_8$	3	5
$Ga_{17} \Rightarrow Ga_{18}$	27	90
$Ga_{29} \Rightarrow Ga_{30}$	25	76
$Ga_{56} \Rightarrow Ga_{57}$	127	161

Table 5.1: Comparative analysis of unique structure obtained using the MeGen model against the DFT-based method.

To demonstrate structure generation using seed structure, MeGen is employed to predict structures of four different sizes $Ga_{[8,18,30,57]}$ from seed structures of $Ga_{[7,17,29,56]}$ respectively. The number of unique structures generated from DFT and MeGen are tabulated in Table 5.1. The number of structures generated from our model is significantly more than the one generated through DFT based search and MeGen has always hit the GS for all sizes. Further in Figure 5.5 we compare structural motifs of isomers generated through DFT vs that of MeGen model for Ga_8. As can be seen in both approaches GS is identical. Even for a small size like Ga_8, MeGen model predicts some structural motifs which are missed out by DFT based search. Further the high energy isomers of Ga_8 are observed as building blocks of larger clusters with more than 30 atoms.[180]

Ga seed ⇒ Ga target structure	Structures from DFT based method	Structures from MeGen model
$Ga_{10} \Rightarrow Ga_{11}$	10	13
$Ga_{11} \Rightarrow Ga_{12}$	10	20
$Ga_{12} \Rightarrow Ga_{13}$	15	35
$Ga_{13} \Rightarrow Ga_{14}$	13	70
$Ga_{14} \Rightarrow Ga_{15}$	37	120

Table 5.2: Iterative generation of Ga_n cluster structures (size n = 11 to 15).

In our second experiment we generated structures of Ga_{11} from seed structures of Ga_{10}. However to generate isomers of Ga_{12} instead of using DFT generated structures we have used MeGen predicted optimize structures of Ga_{11} as seed. This was repeated till Ga_{15}, where seed structures were also generated from MeGen. Table 5.2 demonstrate the number of unique isomers generated through DFT vs MeGen model. As was the previous case number of generated low energy isomers through MeGen is much more than that of DFT.

Further, we also investigated distinct structural families present in these unique structures. Our analysis clearly demonstrates that our model has captured structural motifs which were

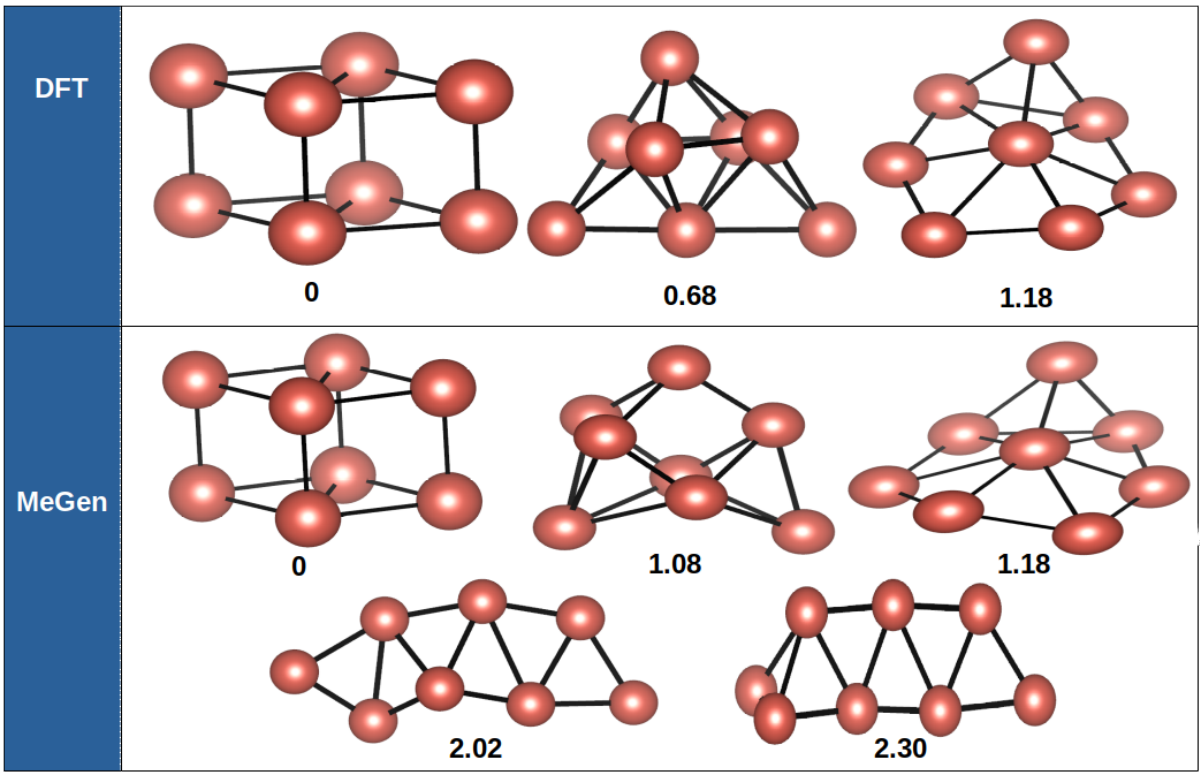


Figure 5.5: Unique structural motifs of Ga_8 obtained from DFT based search method vs MeGen generated structure. The number at the bottom of each structure represents the energy difference between the isomer and the GS.

not captured previously through DFT simulations. For example, DFT based search has predicted only two distinct structural motifs for Ga_{11} where as MeGen has predicted one more. For the next three sizes from Ga_{12} to Ga_{14}, only one structural motif was captured in DFT based searches, whereas MeGen has predicted many more distinct structural motifs. All these structural motifs, along with their energies with reference to their respective GS, are presented in Figure 5.6. The second experiment is the proof of concept that we can generate isomers along with GS from scratch. And the model is successful in capturing various structural motifs. A finer/detailed analysis of these isomer families also brings out the growth pattern in this size range.

5.4 Conclusion

In this study, a reinforcement learning model, namely MeGen, was developed to generate 3D low-energy isomers of Ga clusters. MeGen is significantly more efficient than the conventional workflow for generating ground state geometries as well as low lying isomers in terms of time and computational resources. Integration of the DART model as a reward function to calculate

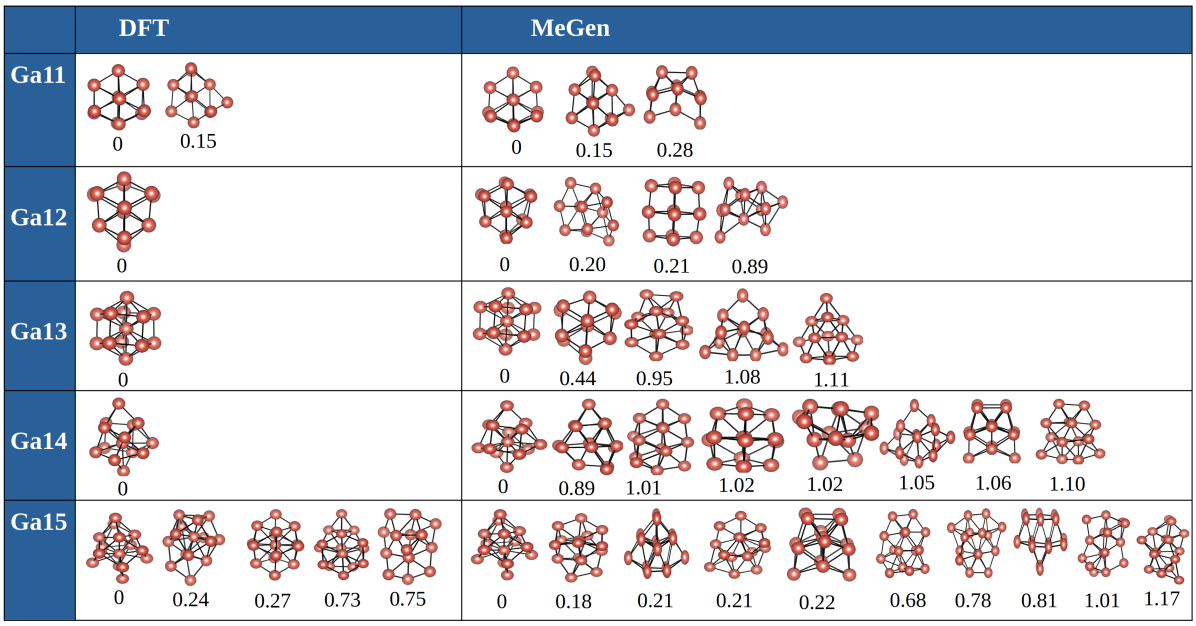


Figure 5.6: Different structural motifs obtained from MeGen and DFT based search methods. Structures labeled by 0 are the GS. The number at the bottom of other structures represents energy difference between isomer and the GS.

energy allows us to efficiently learn the process of addition of a single atom to $(N-1)$ atom cluster to generate (N) atom low energy cluster. Investigating the structure generated using MeGen indicates that the model has learned the underlying cluster growth pattern by its ability to identify valid low energy structures incorporated by the reward function of the RL model. Furthermore, the MeGen model can be extended to any metal cluster.

Essentially MeGen is an efficient search algorithm which generates $n+1$ atom gallium cluster given an n atom cluster. Similarly, molecular geometry optimization is an search algorithms which aims to search for the nearest minima on the PES given a molecular 3D structure. Most optimization problems require that the user selects an algorithm and, to some extent, also tunes it for better performance. Although intuition and knowledge about the problem can speed up these selection and fine-tuning process, users often use trial-and-error methodologies, which can be time-consuming and inefficient. With all of that in mind and much more, the concept of "Learned optimizers," "learning to learn," and "meta-learning" has been gathering attention in recent years. In the next chapter i.e. chapter 6, we propose MolOpt that uses Multi-Agent Reinforcement Learning (MARL) for autonomous molecular geometry optimization (MGO).

Chapter 6

MolOpt: Autonomous Molecular Geometry Optimization using Multi-Agent Reinforcement Learning

6.1 Introduction

Neural network potentials (NNPs) learn to approximate potential energy surface (PES) as a high dimensional function (HDF) f by learning from existing reference data. Once trained NNPs can successfully circumvent the need to solve the electronic Schrödinger equation explicitly as it has learned the mapping $f(Z_i, r_i) \rightarrow E$, where Z_i are the nuclear charges and r_i are the atomic positions.[85, 111, 112, 113, 116, 218, 220] Machine learning (ML) methods in general have been successful in improving computational chemistry algorithms leading to accelerated property prediction and chemical space exploration.[221] Recently, much emphasis has been on developing efficient ML-based search algorithms to explore chemical space,[60, 67, 209, 222] but the same is not the case for conformational space. There are very few attempts to develop an efficient ML-based search algorithm that can explore the conformational space, i.e., probe the potential energy surface (PES).[223, 224, 225, 226, 227] These ML-based search algorithms have applications in 3D structure generation[59, 228] and molecular geometry optimization (MGO). MGO aims to find the nearest/local molecular conformation with minimum potential energy on PES, starting from a given initial 3D conformation. MGO is an essential part of computational chemistry because any studies related to equilibrium geometries demand searches for minima on PES. Over the past several decades, there have been a variety of well-established optimization methods, such as conjugate gradient (CG), steepest descent(SD), Newton-Raphson, and quasi-Newton methods, to solve this task of MGO.[36, 69, 229, 230, 231]

These geometry optimization methods involve using the PES's first-order and/or second-order derivatives. The examples of first-order optimization algorithms are the steepest descent (SD)[232, 233] and conjugate gradient (CG)[70]. These first-order optimization algorithms use gradient information to perform optimization. The steepest descent method takes the next step by searching for the steepest direction to minimize the function's value, given the current

point. This is done by taking a step in the direction of the negative gradient, and step size is calculated using line search. Conversely, the conjugate gradient method involves the use of gradient information to compute n-conjugate (A-orthogonal) directions and takes a gradient direction descent step in these n-conjugate directions at each iteration, thereby reaching the minima in n number of steps. One can see that using the n-conjugate directions, the CG method avoids moving in the zig-zag fashion during optimization, i.e., CG takes a step along these n-conjugate directions only once. SD and CG require gradient calculation to decide the update direction and have very slow convergence (more number of steps) as compared to second-order optimization methods.[229]

Second-order optimization methods make use of the Hessian of the PES for optimization. The benefit of second-order optimization methods is that the use of hessian and gradient provides a much better update step than the step taken with only gradient information. An example of second-order optimization algorithms is the Newton−Raphson method. At each iteration of the Newton-Raphson method, the PES is approximated as a quadratic function locally, and the optimization step is computed as the step towards the minima of this local approximation. Even though these sophisticated, second-order optimization methods converge in fewer steps, they still need to compute the Hessian and its inverse at each iteration, which is computationally expensive. Hence each iteration of second-order optimization methods is computationally expensive compared to first-order optimization methods but takes fewer steps to converge. On the other hand, quasi-Newton methods like BFGS[72, 73, 74, 75] achieve performance similar to the second-order optimization algorithms by circumventing the need to calculate the Hessian and its inverse at each iteration explicitly. These methods instead make a lower-rank approximation to the Hessian using the displacement vectors and gradients and then take a Newton-type update step based on this approximation.

Developing these algorithms is a laborious process, one that needs to be formulated and validated iteratively.[234] Lately, the focus has been on devising new methods to machine-learn molecular features resulting in robust representations. Just as deep learning (DL) has been successful in automating feature engineering, automating algorithm design could open new avenues and change how we design algorithms. Automating algorithm design and learning a "learned optimizer" may outperform current hand-designed optimizers.[234, 235, 236]

Lately, there has been some progress in developing customized optimizers using DL to handle varied optimization problems. For instance, Egidio et al. introduced a "step-size policy" that predicts the step size for the L-BFGS algorithm using the local information at the current position.[235] Andrychowicz et al. developed learning optimization algorithms using a supervised learning technique using long short-term memory networks (LSTMs). They showed that their learning optimization algorithms could solve simple convex optimization problems and were able to optimize neural networks.[237] Metz et al. have discussed the difficulties in training the learned optimizers, and by analyzing the trained optimizer, they desire to acquire

wisdom that may transfer back to hand-designed optimizers.[236] Using RL, Li and Malik developed learned optimizers for different classes of convex and non-convex objective functions and showed that the autonomous optimizers converge in fewer steps and/or reach better optima than hand-designed optimizers.[234] Ahuja et al. have designed an RL-based optimizer that adds a corrective term to the BFGS algorithm.[238]

Motivated by the aforementioned methods, we introduce MolOpt - a novel approach for autonomous molecular geometry optimization (MGO) that utilizes multi-agent reinforcement learning (MARL). By defining the input as an atomic environment vector (AEV) and forces on each atom, we are able to develop an RL-based model known as MolOpt, which outputs displacements of each atom in the molecule. These displacements are used to update the Cartesian coordinates of the atoms in the molecule. MolOpt is a MARL-based model where each atom is treated as an agent. This formulation allows us to use AEVs as input; this mitigates several problems, viz. 1. Due to MARL formulation, the policy is defined for atoms; hence MolOpt can handle the different sizes of molecules and is permutationally invariant, 2. As we have used AEV as input, we have a fixed-size vector incorporating rotational and translational invariance into the model. All these above properties enable us to output action as displacement, which can be used to update the molecular structure directly in the Cartesian coordinate. Our model MolOpt is novel because it is a de-novo learned optimizer for MGO. MolOpt is independent of other optimizers as it does not require other optimizers for training or testing. In the methods section, we briefly introduced MARL and described the MGO problem as MARL formulation, followed by the dataset used for training and testing and "training and implementation details". In the results section, we demonstrated the effect of various input "feature vectors" or state representation s_t and architectural differences on the performance of MolOpt. We show the ability of our learned optimizer, MolOpt, to perform MGO on different classes of alkanes, demonstrating the transferability of our model to different molecules. We also compare MolOpt with other optimizers such as MDMin, FIRE, and BFGS. MolOpt is the first of its kind to apply reinforcement learning to MGO without any dependence on other hand-designed optimizers. Our work serves as a proof-of-concept for the potential of MARL in this domain, opening up new research directions for MGO. The main contributions of the paper are as follows:

- Formulation of molecular geometry optimization as Multi-agent RL (MARL) problem. Where each atom is an agent, thus allowing us to have the same policy across different molecular sizes and mitigate the problem of permutation transformation with the molecules.

- We have developed a *"learned optimizer"* which is in contrast to the hand-designed optimizers available for MGO.

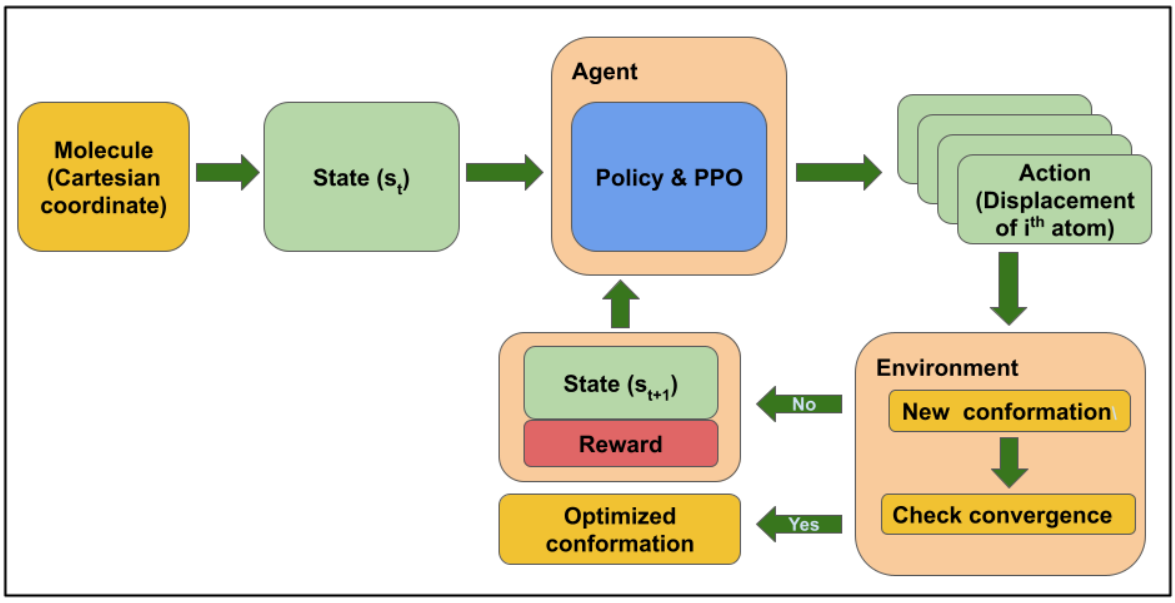


Figure 6.1: The figure shows the workflow of MolOpt model. The molecule's structure is in Cartesian coordinates, which are used to compute rotational and translational invariant state representation. State representation consists of atomic environment vector (AEV), one-hot encoding of atom type, and unit forces. The shared policy network receives atomic state representation as an observation and predicts actions based on those observations. These actions are modeled as displacements of each atom of the molecule in the Cartesian coordinate system. As intended, the actions produce displacement of each atom, resulting in a new conformation. We then calculate the new conformation's energy and forces to check if optimization has reached convergence and calculate reward to check the goodness and badness of the action. If convergence is not achieved, we compute the invariant state representation of the new conformation and repeat the process.

- Our MolOpt model is an non-history based *"learned optimizer"* which needs only a single previous state to predict actions. Which is in contrast to other models which need multiple previous states as observation to predict next action.

- MolOpt's learned policy incorporates the principles of chemistry to optimize molecular geometry with a gradient-based local optimization approach.

- Transferability - Our optimizer trained on small molecules i.e. Ethane and Butane (includes 2 isomers) can be used to optimize larger molecules such as heptane (includes 9 isomers) and octane (includes 18 isomers).

6.2 Method

6.2.1 Preliminaries

In reinforcement learning, the agent chooses actions $a_t \in \mathcal{A}$ at each timestep t, thus changing the state $s_t \in \mathcal{S}$ of the environment in a random manner, and gets feedback based on the outcome of the action. The feedback is commonly provided as a reward or cost $r_t \in \mathcal{R}$. The agent's goal is to take appropriate actions based on the observation/state s_t that maximizes the cumulative reward or minimizes cumulative cost over all time steps.

To this effect, we formulate MGO as an RL problem by defining the potential energy surface (PES) as a game-like environment for repeated exploration of conformation space. An essential aspect of solving an RL problem is by learning a policy using a neural network that can predict appropriate actions by observing different states/points along the surface of the objective function, PES, in this case. Finally, we train our model using a popular RL algorithm known as proximal policy optimization (PPO) to learn the optimal policy. In the following subsections, we will formulate geometry optimization as an RL problem.

6.2.2 Multi-agent Reinforcement learning (MARL)

A reinforcement learning problem is generally represented as a Markov decision process (MDP). We define finite horizon MDP with continuous state and action space as a tuple $(\mathcal{S}, \mathcal{A}, p_o, p, \mathcal{R}, \gamma)$, where a set of states $\mathcal{S}$ encodes the information or knowledge about the environment at various moments of time; a set of actions or desicion $\mathcal{A}$ that helps to move from one state to another; $p_o : \mathcal{S} \longrightarrow \mathcal{R}^+$ is the probability density over initial states, a transition probability function $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow \mathcal{R}^+$ that defines the probability of moving from one state to another on taking a particular action; and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$ that defines the goodness or badness of taking a particular action at some given state and $\gamma \in (0, 1]$ is the discount factor.

By solving the RL problem, we aim to learn a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{R}^+$, which is a conditional probability density over actions given the current state, such that the expected cumulative reward is maximized. In other words, we aim to find a mapping from states to action, called a policy (denoted as π), with maximum achievable total reward. The policy π is often parameterized using a neural network.

To formulate geometry optimization as an MDP, let us consider N atom molecule having cartesian coordinates $x \in \mathbb{R}^{3N}$. Here we define each atom as an agent making this formulation a Multi-agent MDP[239] (MMDP). However, MMDP's presume all agents get same reward. Shapley, in 1953 introduced stochastic Games (aka Markov Games), in which he allowed a unique reward function for each agent.[240]

Partially-Observable Stochastic Games ("POSG") (Lowe et al., 2017)[241], defined below, is an extension of Stochastic Games to the situation in MDP where we have only a partially observable state (similar to a partially-observable MDP)[239], and is the model we use throughout this paper.

Definition : POSG is a tuple $\langle \mathcal{S}, \mathcal{N}, \{\mathcal{A}_i\}, \mathcal{P}, \{\mathcal{R}_i\}, \{\Omega_i\}, \{\mathcal{O}_i\} \rangle$ where:

- $\mathcal{S}$ is set of all possible states.
- $\mathcal{N}$ is number of agents. The set of agents in $[\mathcal{N}]$.
- $\mathcal{A}_i$ is the set of possible actions for agent i.
- $\mathcal{P} : \mathcal{S} \times \prod_{i \in [\mathcal{N}]} \mathcal{A}_i \times \mathcal{S} \longrightarrow [0, 1]$ is the (stochastic) transition function.
- $\mathcal{R}_i : \mathcal{S} \times \prod_{i \in [\mathcal{N}]} \mathcal{A}_i \longrightarrow \mathbb{R}$ is the reward function for agent i.
- Ω_i is the set of possible observations for agent i.
- $\mathcal{O}_i : \mathcal{A}_i, \times \mathcal{S} \times \Omega_i$ is the observations function

6.2.2.1 Parameter Sharing

"Nonstationarity" is a fundamental problem in cooperative MARL.[242] Each agent's policy evolves during learning while it is also part of the environment from the perspective of other agents. This is known as the ringing effect, in which the information oscillates between agents during learning, significantly slowing the convergence. Increasing centralization during learning can mitigate the problem of slow convergence due to nonstationarity.[243] One of the centralized cases of learning is parameter sharing. The concept of parameter sharing is quite common throughout deep learning. In MARL, parameter sharing[244, 245] refers to a learning algorithm that acts on behalf of every agent while using and making updates to a collectively shared policy.[244]

6.2.3 Atomic environment vector (AEV)

The atomic environment vector (AEV) captures the atomic environment around each atom. AEVs are constructed from 'symmetry functions,' which encode each atom's radial and angular environment. As described by Smith et al. in [84], we have used a modified version of the original Behler and Parrinello symmetry function (BPSF).[85] AEV comprises a radial part and an angular part that encode the radial and angular environment around the atom, respectively. As summarized in Table 6.4, we have five variants of the MolOpt model, each employing different state representations and architectures. A detailed discussion of these five variants can be found in the results section. This section explicitly highlights the AEV component of the state

representation used in these variants. For variants 1 and 2, we utilized 32 evenly spaced radial shifting parameters for the radial part and eight radial and eight angular shifting parameters for the angular part. Given that there are two atom types (C and H), this results in a 256-length AEV, where radial and angular parts are of 64 and 192 lengths, respectively. In contrast, for variants 4 and 5, we employed 16 evenly spaced radial shifting parameters for the radial part and eight radial and four angular shifting parameters for the angular part, which results in a 128-length AEV, where radial and angular parts are of 32 and 96 lengths, respectively.

6.2.4 Formulation

Coming back to geometry optimization as an MDP. let us consider N atom molecule having cartesian coordinates $x \in \mathbb{R}^{3N}$. We define each atom as an agent making this formulation an Multi-agent MDP[239] (MMDP), number of agents $\mathcal{N}$ depends on the number of atoms in the molecule. As seen in Figure 5.1 we compute rotationally and translationally invariant state representation s_t at timestep t. State representation s_t encodes the 3D structure of the molecule by the virtue of AEV, atom type and unit forces F_x, F_y, F_z. Each of these entities of state s_t are for single atom in the molecule i.e. dimensions of $AEV = N \times 128$, Atom-type one-hot vector= $N\times$ number of atomic species (2 in our case) and unit forces = $N \times 3$, hence dimension of $s_t = N \times 133$. It should be noted that the state representation s_t is different for different variants, see Table 6.4.

- State, s_t = [AEV, Atom-type (one-hot vector), F_x^t, F_y^t, F_z^t] where F_x^t, F_y^t, F_z^t are the unit forces in x,y,z direction at time t.
- Action, $a_t = [D_x, D_y, D_z]$ (Displacement of atom in x,y,z direction).
- Reward, r_t = Total reward (see equation 6.3), where F_r is the resultant force (eV/Å) on each atom.

$$Atomic\ reward = \begin{cases} -1 & \text{if } F_r > 10 \\ 0 & \text{if } F_r < 10 \text{ or } F_r >= 1 \\ 1 & \text{if } F_r < 1 \text{ or } F_r >= 0.1 \\ 20 & \text{if } F_r < 0.1 \text{ or } F_r >= 0.01 \\ 500 & \text{if } F_r < 0.01 \\ 2000 & \text{if } F_r^{max} < 0.01(converged) \end{cases} \tag{6.1}$$

$$Team\ reward = mean(-log(F_r)) \tag{6.2}$$

$$Total\ reward = Atomic\ reward + Team\ reward \tag{6.3}$$

The atomic agents take in the state representations as observations and uses policy (π) to predict atomic actions (a_t). The next conformation in the optimization trajectory is obtained by using the actions (atomic displacements) and previous conformation as follows $x_{t+1} = x_t + a_t$. As we can compute state representation s_t from x_t (Cartesian coordinates) we can write $s_{t+1} = s_t + a_t$. Now that we have a new molecular conformation in the optimization trajectory, we check for optimization convergence, if convergence is reached, we stop the optimization, if convergence is not reached, we again compute the state representation of new conformation from Cartesian coordinates, pass it to the agent and this loop continues as shown in Figure 5.1. We use the proximal policy optimization (PPO) algorithm to train our policy network. PPO can be categorized as the policy-based RL method that aims to learn the optimal policy (π^*). A policy is a mapping function that predicts appropriate action given a state that results in the maximum possible cumulative reward.

We have designed a custom reward function (see equation 6.3), which is a combination of the atomic reward and team reward (molecular reward). Now, why is this custom reward function important? One must realize that in our multi-agent RL (MARL) formulation, we predict atomwise actions, which leads to the "non-stationarity" problem. There are various ways to mitigate the "non-stationarity" problem; in our work, we use the reward function as an in-direct communication method to overcome the "non-stationarity" problem. Using the team reward function component, we provide the agent with information about what is happening with other agents (atoms). As shown in equation 6.3, the total reward is the summation of atomic and team rewards. The atomic reward is given to each atom based on the resultant force (F_r) on that particular atom (see equation 6.1). Atomic reward aims to reduce the forces on an individual atom. In molecular geometry optimization, the displacement of a single atom leads to changes in the forces of multiple other atoms. Hence, we introduce team reward to prevent actions that will cause the F_r on other atoms to increase dramatically. Team reward, equation 6.2 aims to reduce forces on all atoms without drastically increasing forces on other atoms. In equation 6.1, F_r^{max} is the maximum resultant force on a particular atom in a molecule. If F_r^{max}, which represents the largest resultant atomic force in the molecule, is below 0.01 eV/Å, we conclude that the optimization has converged and terminate the episode.

6.2.5 Dataset

We generated a dataset containing initial conformers of alkanes known as ALINCO. ALINCO dataset contains geometries of Ethane, n-Butane, and Isobutane. Below are the steps followed to generate the ALINCO dataset.

1. From SMILES generate "10 X n atoms" structures for each isomer using RDKit.[120] Therefore, we get a total of 360 structures consisting of 80, 140, and 140 of Ethane, Butane, and Isobutane, respectively.

Parameters	Value
Entropy coefficient	0.0001
KL coefficient	1.0
KL target	0.01
gamma	1.0
Clip parameter	0.3
vf clip parameter	10.0
Horizon	20
lr	5e-05
train batch size	2048

Table 6.1: Hyperparameter used during training and evaluation of MolOpt

2. We then add random noise with mean = 0 and std = 0.1 to the geometries generated in step 1. We repeat this step 5 times, generating 5X gives 1800 structures.

3. Combine the initial 360 and perturbed 1800 structures to get a total of 2160 structures.

4. Optimize these 2160 structures using the BFGS algorithm provided in Atomic Simulation Environment (ASE)[71] package. We use ANI1ccx[68] for energy and force calculation.

5. From optimization trajectory sample 0, 1, 2, 4, 6, 8 initial frames and also sample every ith frame from 10th till 10th last frame at the interval of 5.

In total, ALINCO has 42,262 structures of Ethane, n-Butane, and Isobutane.

6.2.6 Training and Implementation Details

We implemented MolOpt using RLLIB and PPO implemented in RLLIB to train the policy network. The objective function in equation 6.4 is optimized using PPO. In equation 6.4 the parameterized policy is given as $\pi_\theta(a_t|s_t)$ which predicts action a_t given the state s_t and $\pi_{\theta_{old}}(a_t|s_t)$ represents the older iteration of policy network. The clipping parameter ϵ guarantees that while updating the policy, we do not make excessively large updates, and we have set its value to 0.3. The advantage function $A(s_t, a_t)$ is defined as the advantage function that determines how good, or bad the action a_t is on an average for a given state s_t. Other hyperparameters used during training are given in Table 6.1.

$$L(\theta) = \mathbb{E}_t \left[min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}(a_t|s_t)}} A(s_t, a_t), clip \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A(s_t, a_t) \right) \right] \tag{6.4}$$

To train our model, we sample initial conformation from the ALINCO dataset. We then compute rotationally and translationally invariant state representation (s_t). Agent takes (s_t) as an input and predicts actions (a_t). We get new conformation at time ($t+1$) in the optimization

Molecule name	# Isomers × # structures	# Atoms
Propane	1 × 10	11
Pentane	3 × 10	17
Hexane	5 × 10	20
Heptane	9 × 10	23
Octane	18 × 10	26
Total	360	

Table 6.2: Test set that contains 360 structures of alkanes (C_nH_{2n+2}) where n=3,5,6,7 and 8.

trajectory. We then calculate the energy and forces of the new conformation using ANI1ccx.[68] For the agent to understand if the action a_t taken was good or bad, we calculate the reward as given in equation 6.3.

Once trained, we evaluate the model using a test set that contains alkanes (C_nH_{2n+2}) where n=3,5,6,7 and 8. We generate 10 structures each of 36 isomers across different alkanes (C_nH_{2n+2}) where n=3,5,6,7 and 8 using RDKit.[120] In total, we have 360 structures in our test set.

6.3 Results

In this section, we show the ability of our learned optimizer, MolOpt, to perform geometry optimization on different classes of alkanes. We compare five different variants of MolOpt based on their state representation and architectural differences. The performance evaluation metric for MolOpt is based on energy and all-atom RMSD. The all-atom RMSD is calculated by aligning the molecules to remove any translational and rotational transformations prior to computation. All model were trained on ALINCO dataset and evaluated on a test set containing 360 structures of alkanes, details about the number of isomers and structures in the test set is summarized in Table 6.2. The performance of these five variants has been summarised in Table 6.3. We can see that Variant 5, which has state representation s_t = [AEV, Atom-type, F_x, F_y, F_z, ΔF_x, ΔF_y, ΔF_z] achieves the best performance on the test set containing 360 structures with mean $\Delta E = -0.57$ kcal/mol and standard deviation $\Delta E = 0.67 kcal/mol$. In terms of RMSD, Variant 5 achieves an overall mean RMSD of 0.107±0.078 Å. We have further discussed each of these variants in the following subsections.

6.3.1 Flavours of MolOpt/Ablation study

6.3.1.1 Variant 1

We consider Variant 1 as the baseline. As seen in Table 6.4, the state representation (s_t) of variant 1 consists of atomic environment vectors (AEV) of length 256, atom-type as the one-hot

Variant	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
variant 1	360	−3.51	2.12	0.18	0.08
variant 2	360	−1.64	1.25	0.15	0.09
variant 3	360	−1.34	1.14	0.14	0.07
variant 4	360	−0.99	0.94	0.13	0.08
variant 5	360	−0.57	0.67	0.11	0.08

Table 6.3: Geometry optimization performance of different flavors of MolOpt on test set containing 360 structures. Mean $\Delta E = \frac{1}{N}\sum_{i=0}^{N} E^i_{BFGS} - E^i_{MolOpt}$ both E^i_{BFGS} and E^i_{MolOpt} are optimized energies and i runs over the structures in the test set i.e., $N = 360$. Similarly, STD ΔE represents the standard deviations within the ΔE. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure. The mean RMSD= $\frac{1}{N}\sum_{i=0}^{N} RMSD^i$ where i runs over the structures in the test set. The STD RMSD represents the standard deviations with the RMSD

vector of length 2, unit forces of length 3, and resultant force of length 1, hence the length of s_t is 262.

The idea is to provide the model with the local atomic environment of each atom using AEV. Unit forces and resultant force on each atom provide the model with the gradient information. The unit forces are defined as $F_x = \frac{dE}{dx}$, $F_y = \frac{dE}{dy}$, $F_z = \frac{dE}{dz}$ and resultant force is $F_r = \frac{dE}{dr}$, where E is the energy and r is the position. The policy network has four multi-layered perceptrons (MLP) known as MLP_{aev}, MLP_f, MLP_{int}, and MLP_v. MLP_{aev} and MLP_f produces refined aev and force features respectively. We add these features and pass them through Interaction MLP, MLP_{int}, which predicts actions. We use a separate MLP_v, which acts as a value function. The architecture of these MLPs is shown in Table 6.4. We use $LeakyReLU$ in MLP_{aev} and $tanh$ for all other MLPs as the activation function.

Performance of Variant 1 is summarized in Appendix C Table C.1 and C.2. We can see that the mean ΔE and mean RMSD values increase with the size of the alkanes. Variant 1 achieves the overall mean ΔE of -3.51±2.12 kcal/mol and RMSD of 0.18±0.08 Å. In Appendix C Table C.2, the overall Mean RMSD before optimization is 0.25±0.07 Å and after optimization mean RMSD is 0.18±0.08 Å. Considering that it is a baseline model, these results are very encouraging.

6.3.1.2 Variant 2

Variant 2 is similar to variant 1, except the s_t does not have resultant force F_r hence the length of s_t is 261. We have excluded the resultant force as its information is already present in unit forces, and the resultant force in itself does not add much to improve the accuracy of the MolOpt model. Performance of Variant 2 is summarized in Appendix C Table C.3 and C.4. Variant 2 achieves the overall mean ΔE of -1.63±1.24 kcal/mol and RMSD of 0.15±0.09 Å. In

Variants	State	Architecture
Variant 1	[Atom-type, AEV, F_x, F_y, F_z, F_r]	$MLP_{aev} = 262 \times [128]^3$ $MLP_f = 4 \times [128]^4$ $MLP_{int} = 128 \times 3$ $MLP_v = 262 \times 1$
Variant 2	[Atom-type, AEV, F_x, F_y, F_z]	$MLP_{aev} = 261 \times [128]^3$ $MLP_f = 5 \times [128]^4$ $MLP_{int} = [128]^3 \times 3$ $MLP_v = 261 \times [128]^2 \times 1$
Variant 3	[Atom-type, F_x, F_y, F_z]	$MLP_f = 5 \times [128]^5 \times 3$ $MLP_v = 5 \times [128]^3 \times 1$
Variant 4	[Atom-type, AEV, F_x, F_y, F_z]	$MLP_{aev} = 133 \times [128]^3$ $MLP_f = 5 \times [128]^4$ $MLP_{int} = [128]^3 \times 3$ $MLP_v = 133 \times [128]^2 \times 1$
Variant 5	[Atom-type,AEV, F_x, F_y, F_z, ΔF_x, ΔF_y, ΔF_z]	$MLP_{aev} = 136 \times [128]^4$ $MLP_f = 8 \times [128]^3$ $MLP_{int} = [128]^3 \times 3$ $MLP_v = 136 \times [128]^2 \times 1$

Table 6.4: We compare five different variants of MolOpt based on their state representation and architectural differences. We represent $128 \times 128 \times 128$ as $[128]^3$ and so forth in the architecture column. MLP stands for multi-layered perceptron. Subscripts aev stands for the atomic environment vector, f for forces, int for interaction, and v for the value function. F_x, F_y, F_z, ΔF_x, ΔF_y, ΔF_z are unit and delta unit forces in x,y,z direction respectively. F_r is the resultant force.

Appendix C Table C.4, the overall Mean RMSD before optimization is 0.25±0.07 Å and after optimization mean RMSD is 0.15±0.09 Å.

6.3.1.3 Variant 3

The idea is to evaluate the performance of the MolOpt model in the absence of local chemical environment information, hence variant 3 receives only information about the type of atoms and the forces on each atom; AEV is excluded from the state representation as shown in Table 6.4 therefore the length of s_t is 5. The architecture of the policy network is shown in Table 6.4. We use *tanh* as the activation function for all MLPs.

Performance of Variant 3 is summarized in Appendix C Table C.5 and C.6. Variant 3 achieves the overall mean ΔE of -1.33±1.14 kcal/mol and RMSD of 0.14±0.07 Å. In Appendix C Table C.6, the overall Mean RMSD before optimization is 0.25±0.07 Å and after optimization mean RMSD is 0.14±0.07 Å.

6.3.1.4 Variant 4

From variant 3, we see that there is a slight improvement in the performance after the removal of AEV from state representation. This necessitates changes in the AEV; therefore, variant 4 has an AEV of length 128, contrary to the 256 used in previous variants. The length of s_t is 133 due to reduction in length of AEV. Variant 4 is similar to variant 2, except AEV has been changed.

Performance of Variant 4 is summarized in Appendix C Table C.7 and C.8, Variant 4 outperforms all previous variants to achieve the overall mean ΔE of -0.98±0.95 kcal/mol and RMSD of 0.13±0.08 Å. In Appendix C Table C.8, the overall Mean RMSD after optimization mean RMSD is 0.13±0.08 Å.

6.3.1.5 Variant 5

Variant 5 is our best-performing variant. In variant 5, the state representation (s_t) consists of atomic environment vectors (AEV), atom-type as one-hot vector, unit forces and unit delta forces as shown in Table 6.4, the length of s_t is 136.

Here variant 5 not only receives information about the atomic environment, type of atoms, and unit forces but also information about changes in forces on each atom. Unit force $F = \frac{dE}{dr}$ and change in force $\Delta F = \frac{dF}{dr}$, where E is the energy and r is the position.

Performance of variant 5 is summarized in Appendix C Table C.9 and C.10. Extra added information about the change in forces in the state representation improves the performance of variant 5. Variant 5 achieves the overall mean ΔE of -0.57±0.67 kcal/mol and RMSD of 0.10±0.08 Å. In Appendix C Table C.10, the overall Mean RMSD after optimization mean RMSD is 0.10±0.08 Å. In Figure 6.2 we have shown how the RMSD values decrease after optimization using MolOpt model and how the RMSD values increase as the size of the alkanes increases. The RMSD values are calculated by using BFGS optimized structures as reference hence zero in Figure 6.2 indicates BFGS optimized structures.

As can be seen RMSD values about are very close to BFGS optimized structures. In next section we benchmark Variant 5 with few other optimizers viz. FIRE and MDMin.

6.3.2 MolOpt (Variant 5) Benchmark

Over the past several decades, extensive work has delivered many popular optimization methods, like steepest descent, conjugate gradient, Newton-Raphson, and BFGS, to name a few. These methods share one commonality: they are all hand-designed, i.e., human experts carefully ideate, design, and validate these algorithms hence developing these algorithms is a laborious process. Taking inspiration from DL, which was able to automate feature design, we have tried to automate algorithm design and learn a "learned optimizer" that may outperform current hand-designed optimizers. This section shows how our MolOpt model, an "learned

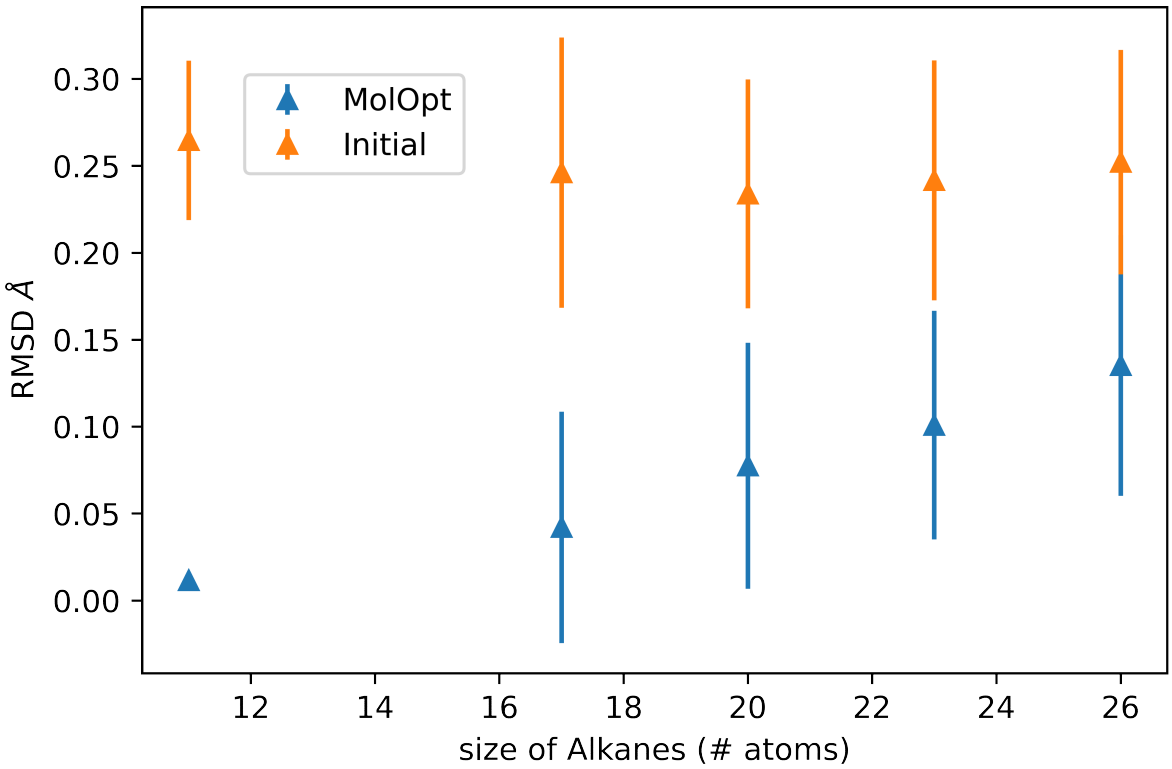


Figure 6.2: Geometry optimization of 360 structures using variant 5. The plot shows the difference in the RMSD values of the different classes of alkanes before and after optimization.

optimizer" compares with other optimizers. We have compared our model MolOpt with three other optimizers, viz. MDMin, FIRE[69], and BFGS.[72, 73, 74, 75] All three optimizers are provided in the ASE package.[71]

The MDMin is a modified version of the typical velocity-Verlet MD method, which numerically solves Newton's second law. However, the dot product between the momenta and the forces is evaluated at each timestep. If the dot product is zero, then it means that the system has just departed through a (local) minima on the PES; the kinetic energy is large and about to decrease again. At this juncture, the momentum is set to zero. Contrary to MD, all atomic masses are set to one. Fast inertial relaxation engine (FIRE)[69] is based on conventional MD with further velocity modifications and adaptive time steps. The Broyden, Fletcher, Goldfarb, and Shanno, or BFGS Algorithm, is a second-order optimization algorithm. BFGS is an example of Quasi-Newton methods in which an approximate Hessian is computed for optimization problems where the second derivative is very expensive to calculate and cannot be computed for all practical purposes. The BFGS is one of the most widely used second-order algorithms for numerical optimization.

As seen from the Appendix C Table C.11, we perform better than the MDMin optimizer by achieving an overall mean ΔE of 0.97±1.06 kcal/mol. The positive mean indicates that MolOpt,

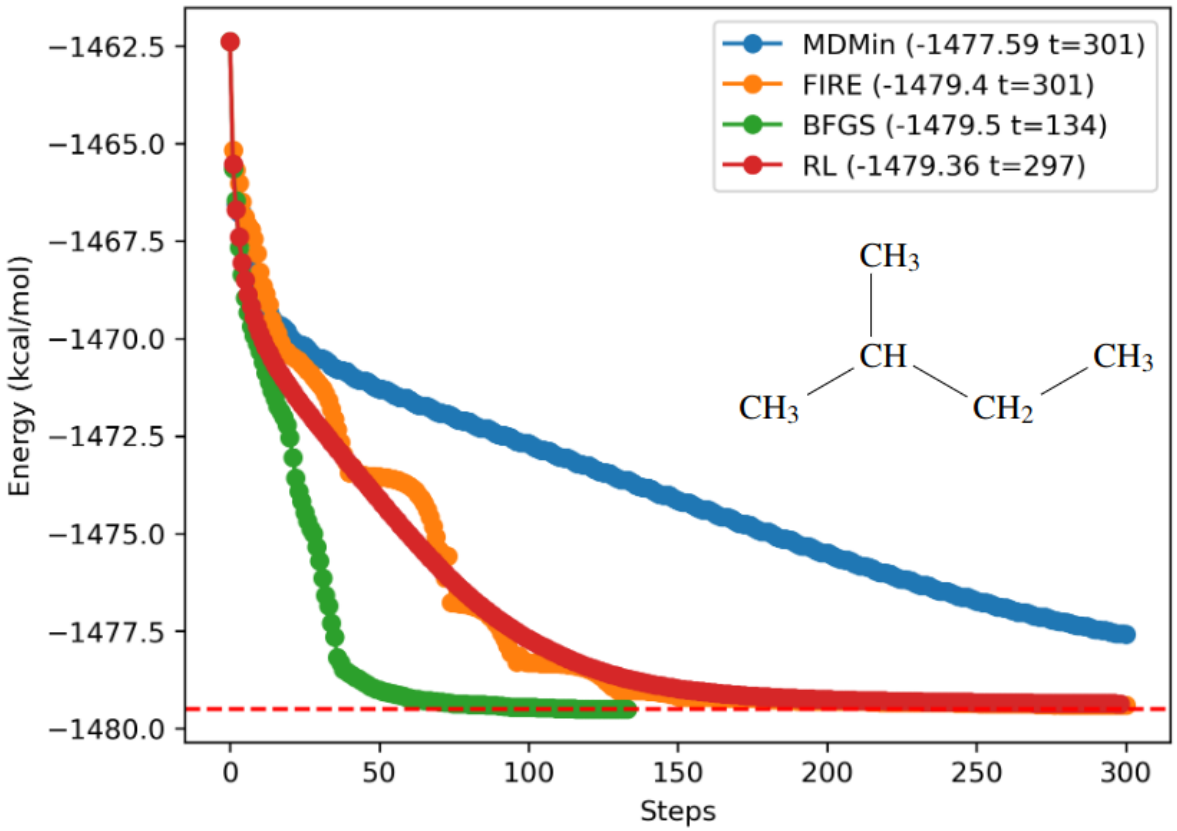


Figure 6.3: The plot shows geometry optimization trajectory of isopentane (empirical formula C_5H_{12}) using four different optimizers viz. MDMin, FIRE, BFGS and RL.

on an average, reached energy levels that were lower than MDMin optimized energy by 0.97 kcal/mol. We have summarized MolOpt comparison with FIRE and BFGS in Appendix C Table C.12 and C.13 respectively. Our model achieves an overall mean ΔE of -0.18±0.25 kcal/mol, an RMSD of 0.04±0.04 Å and ΔE of -0.53±0.64 kcal/mol, an RMSD of 0.10±0.08 Å compared to FIRE and BFGS, respectively. In Figure 6.3, 6.4, 6.5, and 6.6, we have compared geometry optimization trajectory of four different structures with empirical formulas C_5H_{12}, C_7H_{16} and C_8H_{18} with different optimizers. MolOpt performance is better than MDMin and similar to FIRE.

Unlike BFGS, which iteratively approximates the Hessian matrix using information from all previous time steps, MolOpt receives state information only at the current time step t, resulting in less accurate inverse Hessian estimation. As a result, MolOpt may struggle to predict step sizes as precisely as BFGS, leading to more steps being required to reach the minimum.

For the sake of completeness we also compare the performance of MolOpt in terms of speed or computational time required to do 72 optimizations. Out of 360 structures in the test set we pick every 5^{th} structures for optimization hence the number 72 this was done to reduce the computational time and cost. In Appendix C Table C.14 we show time taken by each method to

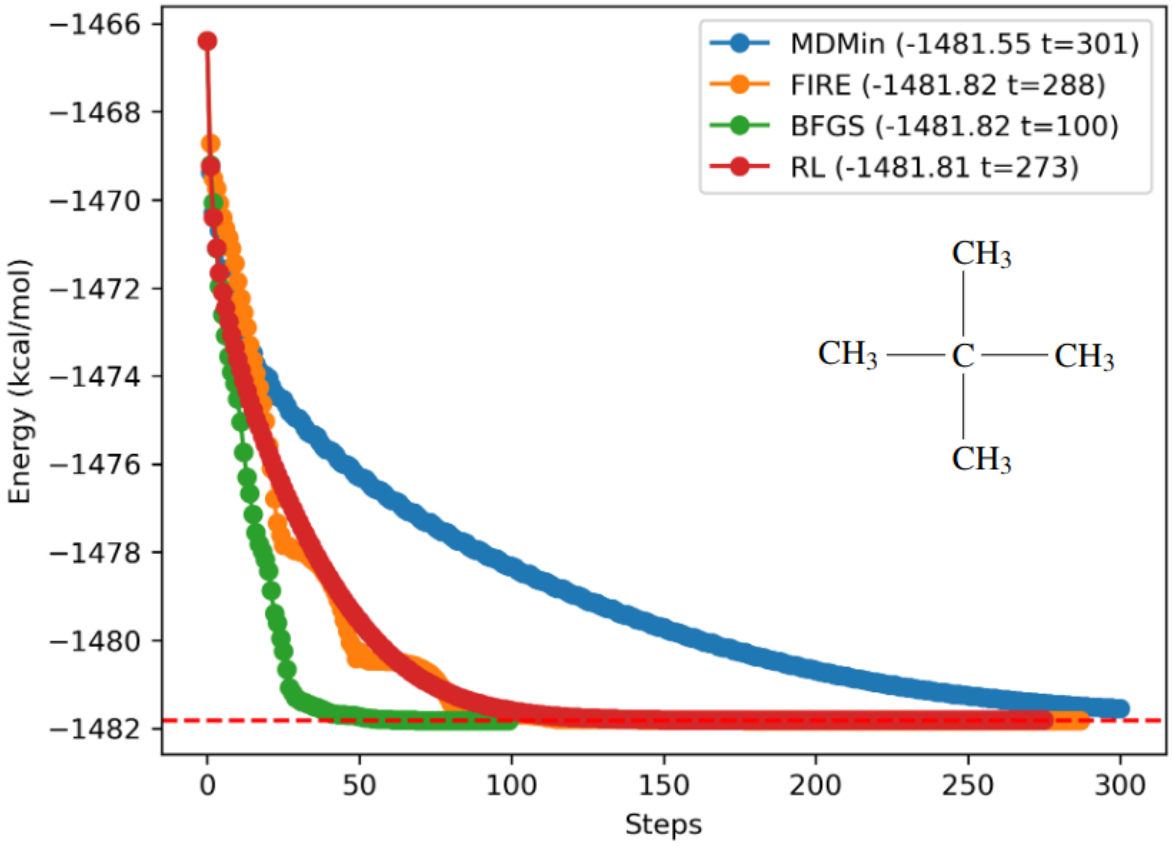


Figure 6.4: The plot shows geometry optimization trajectory of neopentane (empirical formula C_5H_{12}) using four different optimizers viz. MDMin, FIRE, BFGS and RL.

complete these 72 optimization. It is important to acknowledge that other established methods, available as packages, have been in existence for a long time with optimized code for enhanced performance. In contrast, our code was primarily developed to introduce a novel algorithms and has yet to undergo optimization for performance.

6.4 Conclusion and Discussion

In conclusion, we have introduced MolOpt, a robust MARL-based algorithm for MGO. This work serves as a proof-of-concept for the potential of MARL in MGO. The formulation of MGO as a MARL problem where each agent corresponds to a single atom in the molecule allows us to use the same model architecture across different molecular sizes. MolOpt is a "learned optimizer," which is in contrast to the hand-designed optimizer available for MGO. We were able to incorporate chemistry into the learned optimizer. We trained our optimizer on a few alkane molecules, i.e., Ethane and Butane (which include two isomers of butane). We tested MolOpt on different and comparatively larger alkane molecules not present in the

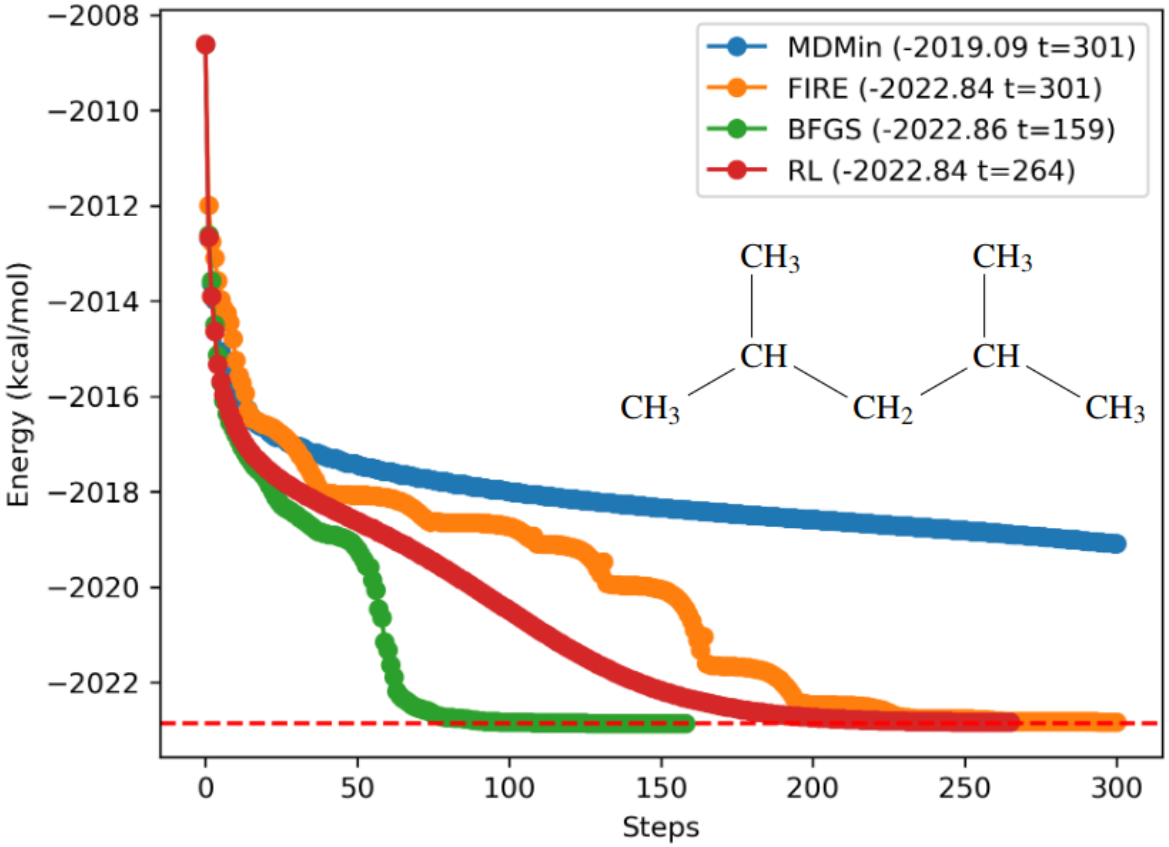


Figure 6.5: The plot shows geometry optimization trajectory of 2,4-dimethyl pentane (empirical formula C_7H_{16}) using four different optimizers viz. MDMin, FIRE, BFGS and RL.

training set, such as Propane, Pentane, Hexane, Heptane (which consists of 5 isomers), and Octane (which consists of 9 isomers). The performance of MolOpt on the test set, which also contains larger alkane molecules such as Heptane and Octane, demonstrates its transferability to other molecules.

Currently, MolOpt uses AEVs, atom type (one-hot-encoding), and forces as the state representation of each atom. Moreover, each agent can only see its own state. In the future, we aim to extend MolOpt to include information sharing and communication between the agents during optimization. In addition, we plan to explore the use of more advanced optimization methods, such as iteratively approximating the Hessian similar to BFGS, to improve MolOpt's performance. It should be noted that our focus in this paper was on demonstrating the feasibility of using MARL for autonomous molecular geometry optimization without any dependency on other hand-designed optimizers. We believe there is still scope for improvement and exploration to develop better MGO algorithms using RL. We hope our work will inspire and motivate further research in MGO using RL. As an afterthought, MolOpt has shown promise in predicting atom displacement using gradients and can potentially be used for molecular dy-

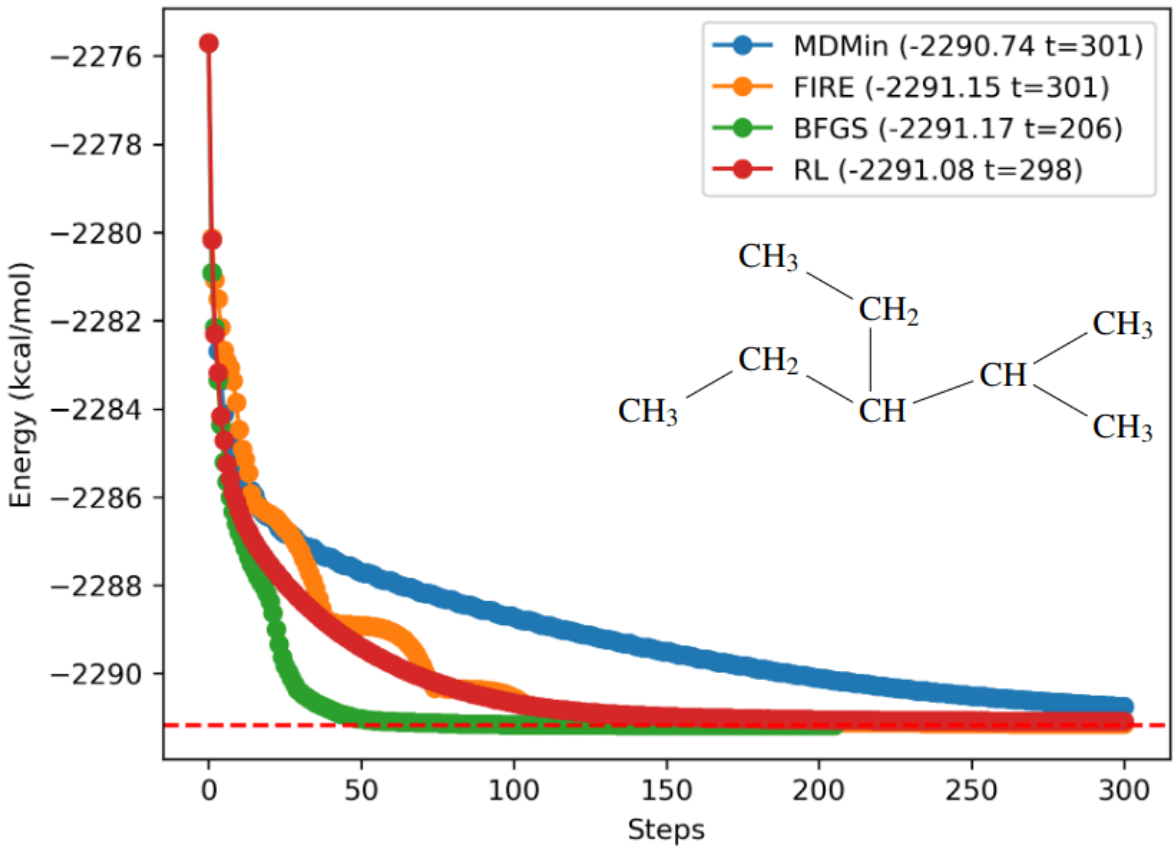


Figure 6.6: The plot shows geometry optimization trajectory of 3-ethyl-2-methyl pentane (empirical formula C_8H_{18}) using four different optimizers viz. MDMin, FIRE, BFGS and RL.

namics (MD) simulations. Training MolOpt to generate an MD simulation trajectory can offer a new perspective toward Molecular dynamics simulations.

This chapter serves as a proof-of-concept for the potential of MARL in MGO. Further enhancements in algorithm and data are necessary to extend MolOpt's optimization capabilities to molecules containing elements such as C, H, N, and O. These changes has been described in next chapter i.e. chapter 7.

Chapter 7

MolOpt2: Autonomous Molecular Geometry Optimization using Multi-Agent Reinforcement Learning

7.1 Introduction

Deep learning has demonstrated that learned functions or features dramatically outperform hand-designed functions or features in terms of accuracy and generalizability. This is due to the ability of DL methods to learn these functions and features from the structure in the data. Similarly, this implies that learned optimizers/algorithms may similarly outperform current hand-designed optimizers/algorithms and may learn task-specific structures, enabling dramatic performance improvements over more general hand-designed optimizers/algorithms. From a broader perspective, most optimization problems require that the user selects an algorithm and, to some extent, also tunes it for better performance. Although intuition and knowledge about the problem can speed up these selection and fine-tuning process, users often use trial-and-error methodologies, which can be time-consuming and inefficient. With all of that in mind and much more, the concept of "Learned optimizers," "learning to learn," and "meta-learning" has been gathering attention in recent years.[234, 235, 236, 237, 246, 247, 248, 249]

In computational chemistry, molecular geometry optimization (MGO) is a costly and crucial task. Local MGO can be considered a search algorithm that aims to find the nearest/local molecular conformation with minimum potential energy on the potential energy surface (PES), starting from a given initial 3D conformation. Over the past several decades, there have been a variety of well-established hand-designed optimization methods, such as conjugate gradient (CG), steepest descent(SD), Newton-Raphson, and quasi-Newton methods.[36, 69, 229, 230, 231] MGO depends on the efficiency of the search algorithm and the accuracy of the interatomic interaction potential.[60] Much work has been done on improving the accuracy, reducing the computational cost, and developing a generalizable neural network potential (NNP). These NNPs, to some extent, have addressed the second part, i.e., they can accurately capture the interatomic interactions with reduced computational cost to predict the chemical properties

of a molecule given its 3D structure. This work focuses on improving the search algorithm (optimizer) by formulating it as a Multi-agent RL (MARL) problem in a meta-learning setting.

Lately, there has been some progress in developing learned optimizers using DL to handle varied optimization problems. For instance, Egidio et al. introduced a "step-size policy" that predicts the step size for the L-BFGS algorithm using the local information at the current position.[235] Andrychowicz et al. developed learning optimization algorithms using a supervised learning technique using long short-term memory networks (LSTMs). They showed that their learning optimization algorithms could solve simple convex optimization problems and were able to optimize neural networks.[237] Metz et al. have discussed the two major difficulties in training the learned optimizers viz 1. bias introduced by truncated backpropagation through time (TBPTT) and 2. exploding gradients. To combat this, the authors constructed a variational bound of the outer objective and minimized this via a combination of reparameterization and ES-style gradient estimators.[236] Their other work includes generalizing the learned optimizers on various tasks, learning hyperparameter search strategies, and better training strategies for learned optimizers.[247, 248, 249] Using RL, Li, and Malik developed learned optimizers for different classes of convex and non-convex objective functions and showed that the autonomous optimizers converge in fewer steps and/or reach better optima than hand-designed optimizers.[234] Ahuja et al. have designed an RL-based optimizer that adds a corrective term to the BFGS algorithm.[238]

In our previous work, we developed MolOpt, a learned optimizer for MGO that utilizes multi-agent reinforcement learning (MARL). Previously we defined the input as an atomic environment vector (AEV) and forces on each atom and outputs displacements of each atom in the molecule. These displacements were used to update the Cartesian coordinates of the atoms in the molecule. MolOpt is a MARL-based model where each atom is treated as an agent. This formulation allows us to use AEVs as input; this mitigates several problems, viz. 1. Due to MARL formulation, the policy is defined for atoms; hence MolOpt can handle the different sizes of molecules and is permutationally invariant, 2. As we have used AEV as input, we have a fixed-size vector incorporating rotational and translational invariance into the model. All these above properties enable us to output action as displacement, which can be used to update the molecular structure directly in the Cartesian coordinate.

In this work, we have made improvements over our previous MolOpt model. In contrast to MolOpt, which was limited to alkanes, MolOpt2 can perform MGO on a wide variety of molecules (containing CHNO). We have tested MolOpt2 on the OptQM9 test set, which contains diverse molecules randomly sampled from QM9. MolOpt10 and OptQM9 are the training and testing dataset generated during this work. We compare MolOpt2 with our previous model by evaluating it on the alkanes test set, showing performance improvement. This improvement is attributed to two major changes we have made, the first being the change in the state representation and the second being the change in the architecture of the policy network. Minor

changes include preprocessing the state representation described in [237]. These improvements were made with the aim to mimic the BFGS algorithm. BFGS algorithm can get an excellent approximation of the inverse of Hessian just by predicting the initial inverse of Hessian and updating it iteratively during the optimization process just by using the gradient vectors and displacement vectors. The BFGS algorithm motivated us to use gradient and displacement vectors in our state representation along with one-hot representation for atom type. Regarding the policy network, we noted that we see improved performance when the critic-network architecture is similar to the actor-network, along with the use of skip connections. In the methods section, we briefly introduced MARL and described the MGO problem as MARL formulation, followed by the dataset used for training and testing and "training and implementation details". In the results section, we compare MolOpt2 with our previous model. Further, we evaluate our MolOpt2 model on the OptQM9 test set and demonstrate its performance and generalizability on a diverse set of molecules containing CHNO elements and various bond types. We also compare MolOpt2 with other optimizers such as MDMin, FIRE, and BFGS.

7.2 Method

This section explains the methodology/algorithms used in this work. We begin by establishing the preliminaries and terminologies in reinforcement learning in subsection 6.2.1. Following that, we delve into Multi-Agent Reinforcement Learning (MARL), which serves as the core framework of our MolOpt2 model in subsection 6.2.2. Within the MARL subsection, we elaborate on parameter sharing in subsubsection 6.2.2.1, a mechanism that facilitates efficient information exchange among the agents to enhance their collaborative decision-making process. Next, we present the formulation of our MGO problem, outlining the specifics such as state representation, actions, and reward function that guide our MolOpt2 model. Moving forward, we discuss dataset generation for training and evaluating the MolOpt2 model and provide insights into its composition. Finally, we provide details regarding the training process and implementation aspects, including the network architecture, hyperparameters, and training protocols.

7.2.1 Formulation

For an N atom molecule we have N agents, each agent receives partial observations of the molecular system, i.e., at time step t each i^{th} atom agent receives information about its own state S_t^i, it takes action a_t^i and gets reward r_t^i. For the sake of brevity we will drop the atom superscript in the rest of the chapter. The state representation consists of gradient $g_t = [F_x^t, F_y^t, F_z^t]$, where F_x^t, F_y^t, F_z^t are the unit forces in x,y,z direction and displacement $D_t = [D_x^t, D_y^t, D_z^t]$ where D_x^t, D_y^t, D_z^t are displacement to move the i^{th} atom in Cartesian coordinates at time t.

- State, s_t = [g_t, g_{t-1}, g_{t-2}, g_{t-3}, Atom-type (one-hot vector), D_t, D_{t-1}, D_{t-2}, D_{t-3}] where g_t is the gradient and D_t is atom displacement at time t.

- Action, $a_t = [D_x^t, D_y^t, D_z^t]$

- Reward, r_t = Total reward (see equation 6.3), where F_r is the resultant force (eV/Å) on each atom.

$$s_t = \begin{cases} \left(\frac{log(|s_t|)}{p}, sgn(s_t)\right) & \text{if } |s_t| \geq e^{-p} \\ (-1, e^p s_t) & \text{otherwise} \end{cases} \tag{7.1}$$

The atomic agents take in the state representations as observations and uses policy (π) to predict atomic actions (a_t). The next conformation in the optimization trajectory is obtained by using the actions (atomic displacements) and previous conformation as follows $x_{t+1} = x_t + a_t$. As we can compute state representation s_t from x_t (Cartesian coordinates) we can write $s_{t+1} = s_t + a_t$. Now that we have a new molecular conformation in the optimization trajectory, we check for optimization convergence, if convergence is reached, we stop the optimization, if convergence is not reached, we again compute the state representation of new conformation from Cartesian coordinates, pass it to the agent and this loop continues as shown in Figure 5.1. We use the proximal policy optimization (PPO) algorithm to train our policy network. PPO can be categorized as the policy-based RL method that aims to learn the optimal policy (π^*). A policy is a mapping function that predicts appropriate action given a state that results in the maximum possible cumulative reward.

We have designed a custom reward function (see Equation 6.3), which is a combination of the atomic reward and team reward (molecular reward). Now, why is this custom reward function important? One must realize that in our multi-agent RL (MARL) formulation, we predict atomwise actions, which leads to the "non-stationarity" problem. There are various ways to mitigate the "non-stationarity" problem; in our work, we use the reward function as an in-direct communication method to overcome the "non-stationarity" problem. Using the team reward function component, we provide the agent with information about what is happening with other agents (atoms). As shown in Equation 6.3, the total reward is the summation of atomic and team rewards. The atomic reward is given to each atom based on the resultant force (F_r) on that particular atom (see Equation 6.1). Atomic reward aims to reduce the forces on an individual atom. In molecular geometry optimization, the displacement of a single atom leads to changes in the forces of multiple other atoms. Hence, we introduce team reward to prevent actions that will cause the F_r on other atoms to increase dramatically. Team reward, Equation 6.2 aims to reduce forces on all atoms without drastically increasing forces on other atoms. In Equation 6.1, F_r^{max} is the maximum resultant force on a particular atom in a molecule. If F_r^{max}, which represents the largest resultant atomic force in the molecule, is below 0.01 eV/Å, we conclude that the optimization has converged and terminate the episode.

7.2.2 Dataset

We generated a dataset containing geometries of molecules listed in Table 7.1 known as OptMol10. Below are the steps followed to generate the OptMol10 dataset.

1. From SMILES generate "10 X n atoms" structures for each molecule using RDKit.[120] Therefore, in total we get 630 structures.

2. We then add random noise with mean = 0 and std = 0.1 to the geometries generated in step 1. We repeat this step 5 times, generating 5X gives 3150 structures.

3. Combine the initial 630 and perturbed 3150 structures to get a total of 3780 structures.

4. Optimize these 3780 structures using the BFGS algorithm provided in Atomic Simulation Environment (ASE)[71] package. We use ANI1ccx[68] for energy and force calculation.

5. From optimization trajectory sample frames with index = [0, 1, 2, 4, 6] as initial frames and also sample every ith frame from 10th till 10th last frame at the interval of 5.

OptMol10 dataset consists of 41,027 molecular structures of ten small organic molecules with varying bond types and numbers of atoms, as summarized in Table 7.1. The dataset was constructed by sampling from the optimization trajectory as described above. OptMol10 was used to train the MolOpt2 model. The availability of this dataset may offer opportunities for the development and evaluation of new algorithms and models in the field of molecular optimization.

Molecule	Formula	Bond type	# atoms(heavy atoms)
Hydroxylamine	NH_2OH	1	5(2)
Ethylene oxide	C_2H_4O	cyclic ether, 1	7(3)
Methoxymethane	C_2H_6O	1	9(3)
Nitromethane	CH_3NO_2	dipole, 1, 2	7(4)
Methyl formate	$C_2H_4O_2$	1, 2	8(4)
Methyl isocyanate	C_2H_3NO	1, 2	7(4)
Formaldehyde	CH_2O	1, 2	4(2)
Acetonitrile	C_2H_3N	1, 3	6(3)
Methyl isocyanide	C_2H_3N	1, 3	6(3)
Cyanogen	C_2N_2	1, 3	4(4)

Table 7.1: Summary of the molecules used in the training set of MolOpt2, including the molecule name, chemical formula, bond type for which the molecule was included in the dataset, and number of atoms (with the number of heavy atoms in parentheses). The training set consists of approximately 41k structures in total of these molecules.

For testing our model we create OptQM9 testset. OptQM9 contains 306 molecules (containing CHNO) which are randomly sampled from QM9. We have summarized OptQM9 in Table 7.2. To show the diversity of our OptQM9 test set we perform t-Distributed Stochastic

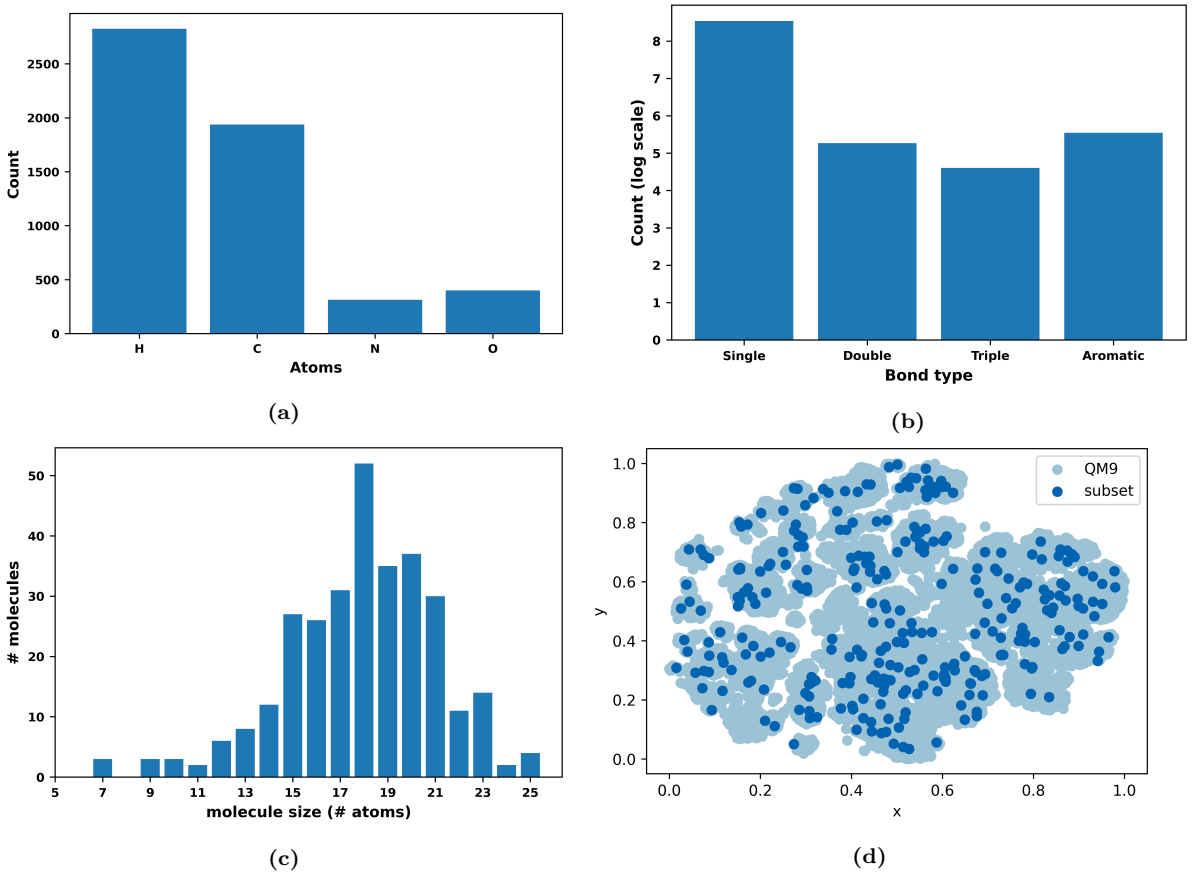


Figure 7.1: (a) Shows the bar plot of individual atom count and their distribution in OptQM9 test set. (b) Shows the bar plot of bond types and their frequency in OptQM9 test set. (c) Shows the distribution of molecules w.r.t size and (d) Shows the tSNE plot of QM9 and the subset a.k.a OptQM9. We have used features obtained from Mol2vec model to generate this tSNE. This shows the diversity of our test set OptQM9 w.r.t QM9

Neighbor Embedding (tSNE) on the feature vectors obtained from Mol2Vec[250] model, the tSNE plot is shown in Figure 7.1d. We have also shown the distribution of individual atom type, bond type and number of molecules w.r.t molecule size in Figure 7.1a, 7.1b, and 7.1c respectively.

7.2.3 Training and Implementation Details

We implemented MolOpt2 using RLLIB and PPO implemented in RLLIB to train the policy network. The objective function in equation 7.2 is optimized using PPO. In equation 7.2 the parameterized policy is given as $\pi_\theta(a_t|s_t)$ which predicts action a_t given the state s_t and $\pi_{\theta_{old}}(a_t|s_t)$ represents the older iteration of policy network. The clipping parameter ϵ guarantees that while updating the policy, we do not make excessively large updates, and we have set its value to 0.3. The advantage function $A(s_t, a_t)$ is defined as the advantage function

# atoms	# molecules	bond type
7	3	1, 2, 3, Ring
9	3	1, 2, 3
10	3	1, 2, 3
11	2	1, 2, Ring
12	6	1, 2, 3, Ring
13	8	1, 2, 3, Ring
14	12	1, 2, 3, Ring
15	27	1, 2, 3, Ring
16	26	1, 2, 3, Ring
17	31	1, 2, 3, Ring
18	52	1, 2, 3, Ring
19	35	1, 2, 3, Ring
20	37	1, 2, 3, Ring
21	30	1, 2, 3
22	11	1, 2, 3
23	14	1, 2, 3
24	2	1
25	4	1, 2

Table 7.2: The OptQM9 test set used to evaluate the performance of MolOpt2 has a total of 306 molecules given in the table.

Molecule name	# Isomers × # structures	# Atoms
Propane	1 × 10	11
Pentane	3 × 10	17
Hexane	5 × 10	20
Heptane	9 × 10	23
Octane	18 × 10	26
Total	360	

Table 7.3: Test set that contains 360 structures of alkanes (C_nH_{2n+2}) where n=3,5,6,7 and 8.

that determines how good, or bad the action a_t is on an average for a given state s_t. Other hyperparameters used during training are given in Appendix D Table D.1.

$$L(\theta) = \mathbb{E}_t \left[min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}(a_t|s_t)}} A(s_t, a_t), clip \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) A(s_t, a_t) \right) \right] \tag{7.2}$$

To train our model, we sample initial conformation from the OptMol10 dataset. We then compute the state representation (s_t). We preprocess the state representation (s_t) before feeding it to the network similar to Andrychowicz et al.[237] as $s_t = (ln(|s_t|), sign(s_t))$. We observe that using this preprocess, our model learns much better as gradients and displacement vectors in our state representation can have a large range of magnitude. The agent takes (s_t) as an input and predicts actions (a_t). We get new conformation at the time ($t+1$) in the optimization

Variant	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
MolOpt	360	−0.57	0.67	0.11	0.08
MolOpt2	360	−0.35	0.55	0.09	0.08

Table 7.4: Geometry optimization performance of different flavors of MolOpt on test set containing 360 structures. Mean $\Delta E = \frac{1}{N}\sum_{i=0}^{N} E^i_{BFGS} - E^i_{MolOpt}$ both E^i_{BFGS} and E^i_{MolOpt} are optimized energies and i runs over the structures in the test set i.e., $N = 360$. Similarly, STD ΔE represents the standard deviations within the ΔE. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure. The mean RMSD= $\frac{1}{N}\sum_{i=0}^{N} RMSD^i$ where i runs over the structures in the test set. The STD RMSD represents the standard deviations with the RMSD

trajectory. We then calculate the energy and forces of the new conformation using ANI1ccx.[68] For the agent to understand if the action a_t taken was good or bad, we calculate the reward as given in Equation 6.3. Once trained, we evaluate the model using OptQM9 test set.

7.3 Results

In this section, we show the ability of our learned optimizer, MolOpt2, to perform geometry optimization on OptQM9 test set. The performance evaluation metric for MolOpt2 is based on energy and all-atom RMSD. The all-atom RMSD is calculated by aligning the molecules to remove translational and rotational transformations before computation. All models were trained on the OptMol10 dataset and evaluated on a OptQM9 test set containing 306 molecules containing elements Carbon, Hydrogen, Nitrogen and Oxygen (CHNO).

7.3.1 Comparsion with previous model

In this section, we compare our MolOpt2 with our previous work. ALINCO dataset contains geometries of Ethane, n-Butane, and Isobutane. The ALINCO dataset was created similarly to OptMol10. We have also described the steps followed to generate the ALINCO dataset in our previous work. We train MolOpt2 using ALINCO dataset and evaluate using the test set as summarized in Table 7.3 so as to make a comparison between our previous model and MolOpt2. As shown in Table 7.4, MolOpt2 outperforms our previous model. MolOpt2 achieves mean $\Delta E = -0.35$ kcal/mol and standard deviation $\Delta E = 0.55$ kcal/mol. In terms of RMSD, MolOpt2 achieves an overall mean RMSD of 0.09±0.08 Å. Detailed performance of MolOpt2 in comparison with BFGS is given in Appendix D Table D.3. We have also compared MolOpt2 with other optimization methods, such as FIRE and MDMin, in which MolOpt2 achieves mean $\Delta E = 0.00 \pm 0.14$ kcal/mol and $\Delta E = 1.15 \pm 1.08$ kcal/mol, respectively. Detailed results of the performance of MolOpt2 in comparison with FIRE and MDMin are given in Table D.5 and Table D.4. Thus, our MolOpt2 model outperforms MDMin, performs similarly to FIRE, and is very close to the BFGS algorithm. This improvement is attributed to two major changes

we have made, the first being the change in the state representation and the second being the change in the architecture of the policy network. State representation s_t can have large range of magnitude which causes difficulty while training, hence we preprocess the state representation using Equation 7.1 as described in [237]. These improvements were made with the aim to mimic the BFGS algorithm. BFGS algorithm can get an excellent approximation of the inverse of Hessian just by predicting the initial inverse of Hessian and updating it iteratively during the optimization process just by using the gradient vectors and displacement vectors. The BFGS algorithm motivated us to use gradient and displacement vectors in our state representation along with one-hot representation for atom type. Regarding the policy network, we noted that we see improved performance when the critic-network architecture is similar to the actor-network, along with the use of skip connections.

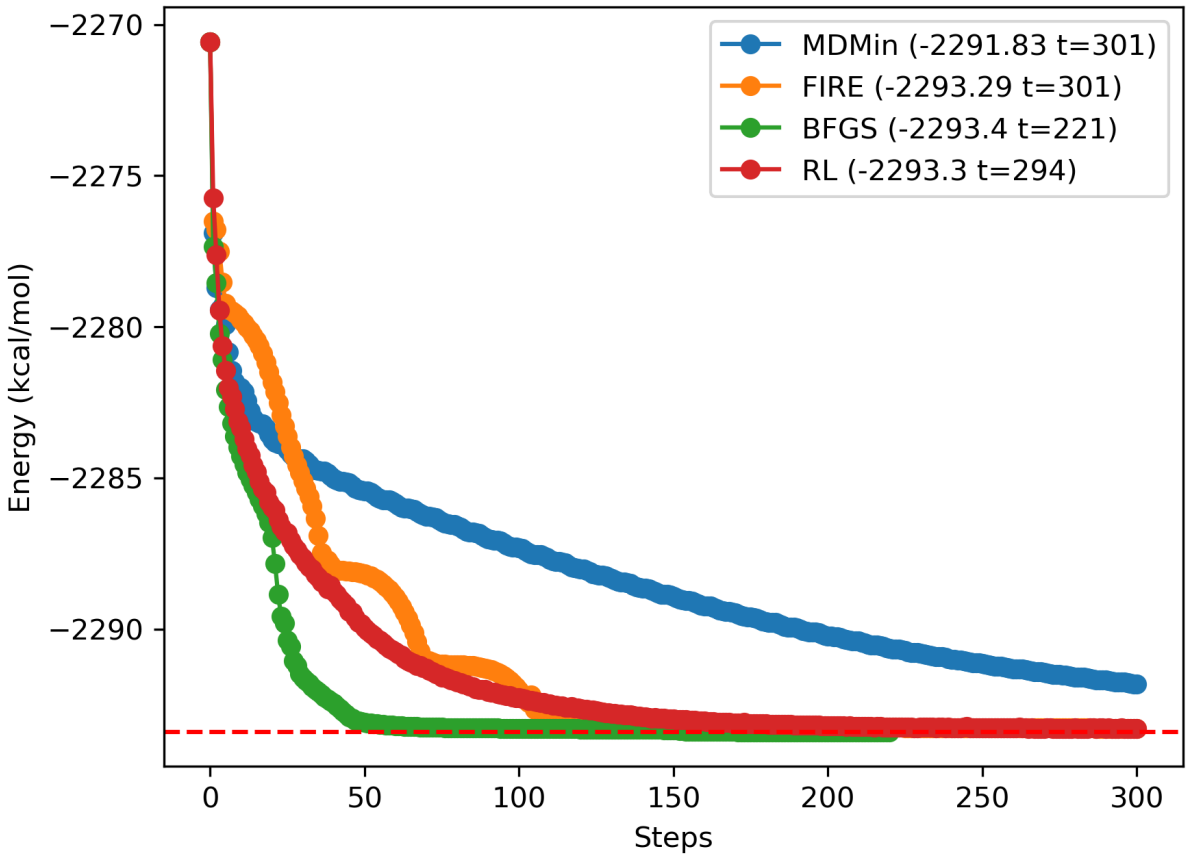


Figure 7.2: Geometry optimization of one of the structures with empirical formula C_8H_{18}. The plot shows that our RL model was able to reach $E = -2293.30$ kcal/mol in 294 steps whereas it took MDMin, FIRE and BFGS 301, 301 and 221 steps to reach $E = -2291.83$, $E = -2293.29$ and $E = -2293.40$ kcal/mol, respectively.

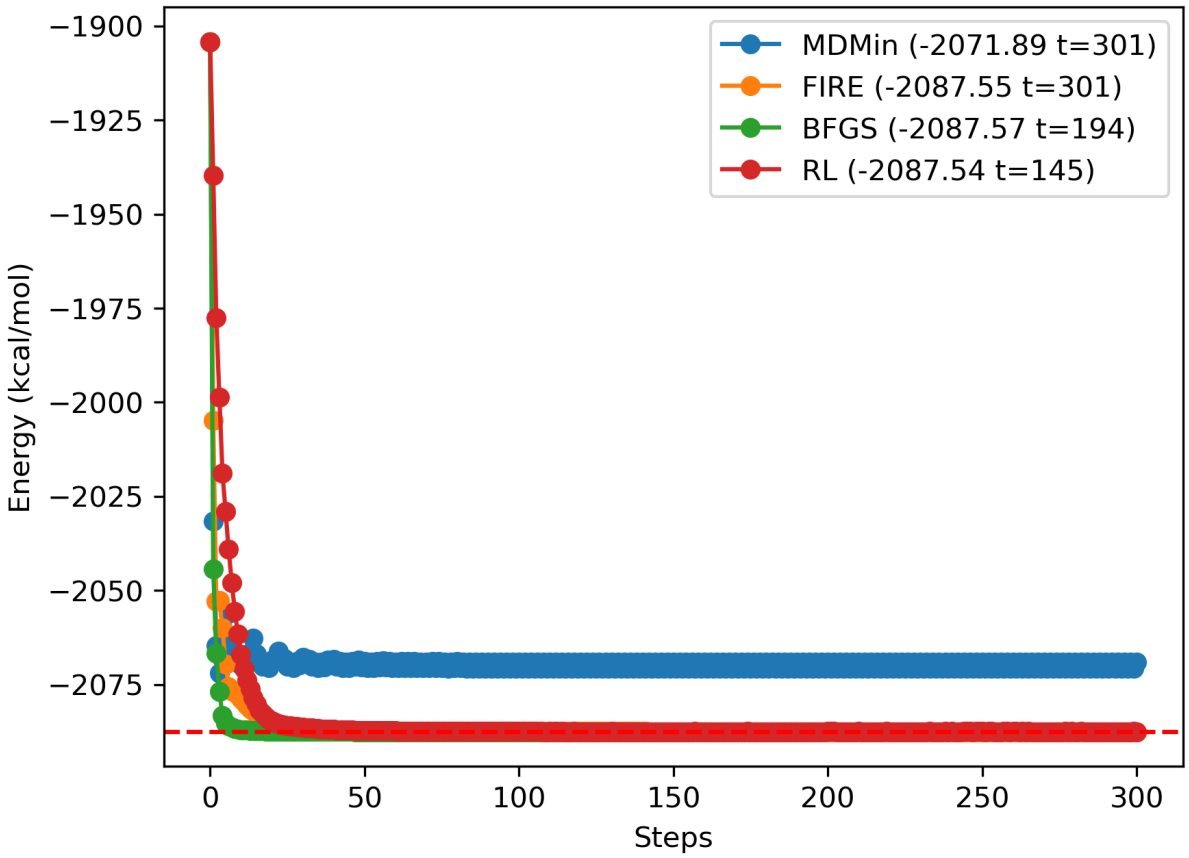


Figure 7.3: Geometry optimization of one of the structures with empirical formula C_7ONH_{15}. The plot shows that our RL model was able to reach $E = -2087.54$ kcal/mol in 145 steps whereas it took MDMin, FIRE and BFGS 301, 301 and 194 steps to reach $E = -2071.89$, $E = -2087.55$ and $E = -2087.57$ kcal/mol, respectively.

7.3.2 Performance on OptQM9 testset

In this section, we evaluate our model MolOpt2 on a new test set known as OptQM9. We have trained MolOpt2 using ALINCO + OptMol10 dataset. The aim is to show the robustness of our model to perform MGO on a wide variety of molecules that are well distributed in chemical space (containing CHNO). Test set OptQM9 was created by random sampling from QM9 with the goal to include a wide variety of molecules with sizes of up to nine heavy atoms containing CHNO elements. To ascertain the molecular diversity within the OptQM9 test set, we perform t-Distributed Stochastic Neighbor Embedding (tSNE) on the feature vectors obtained from Mol2Vec[250] model. The tSNE plot is shown in Figure 7.1d. MolOpt2 achieves mean $\Delta E = -0.09$ kcal/mol and standard deviation $\Delta E = 0.19$ kcal/mol. In terms of RMSD, MolOpt2 achieves an overall mean RMSD of 0.03±0.05 Å. The performance of the MolOpt2 model on the OptQM9 test set in comparison with BFGS, FIRE, and MDmin is summarized in Table 7.5, Appendix D Table D.6 and Table D.7, respectively.

# Atoms	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
7	0.00	0.00	0.00	0.00
9	0.00	0.00	0.00	0.00
10	−0.01	0.00	0.01	0.01
11	0.00	0.00	0.00	0.00
12	−0.01	0.01	0.01	0.00
13	−0.08	0.20	0.02	0.05
14	−0.04	0.08	0.02	0.02
15	−0.08	0.14	0.03	0.04
16	−0.05	0.05	0.02	0.02
17	−0.19	0.38	0.06	0.10
18	−0.07	0.08	0.03	0.04
19	−0.07	0.09	0.03	0.03
20	−0.11	0.17	0.04	0.04
21	−0.12	0.22	0.04	0.06
22	−0.30	0.37	0.07	0.08
23	−0.10	0.18	0.02	0.03
24	−0.03	0.00	0.02	0.02
25	−0.07	0.05	0.02	0.01
overall	−0.09	0.19	0.03	0.05

Table 7.5: Trained on all molecules(OptMol10 and alkanes)

7.4 Conclusion

Most of the work done in learned optimizers and meta-learning is to develop learned optimizers to accelerate the training process of machine learning algorithms and models. There are very few attempts to develop learned optimizers in computational chemistry for MGO. Ahuja et al.[238] this and our previous work on learned optimizers for MGO are some of the initial attempts to explore this field. In this work, we developed a learned optimizer called MolOpt2 for MGO. We have created OptMol10 and OptQM9 datasets to train and evaluate MolOpt2. We show the diversity of the molecules in the OptQM9 test set by comparing it with the QM9 dataset using a tSNE plot. We compare the performance of MolOpt2 with our previous model MolOpt. We have also compared MolOpt2 with hand-made optimizers such as BFGS, FIRE, and MDMin. Our MolOpt2 model, when trained on just three molecules (Ethane and Butane (includes two isomers)), can optimize much larger molecules such as heptane (includes nine isomers) and octane (includes 18 isomers). Moreover, when trained on 13 molecules containing CHNO elements MolOpt2 can optimize randomly chosen molecules from the QM9 dataset. We observe that BFGS outperforms MolOpt2 w.r.t number of optimization steps for small molecules. However, this is different with larger molecules where MolOpt2 and BFGS performance are similar in the context of the number of steps. We would like to explore learned optimizers for small peptides in the future.

Chapter 8

Conclusion

While deep NNPs have demonstrated great potential in achieving accurate energies, systematic benchmarking of these NNPs was necessary for broader applicability and transferability. We did the standard comparative evaluation study of four NNPs, i.e., ANI, PhysNet, SchNet, and BAND-NN. These models were originally trained and tested on different data sets, which makes their applicability and comparison difficult. The benchmark study discusses the descriptors used in these NNPs and their architectures. The critical factor is that NNPs must exploit/incorporate the molecular invariances. Transformations such as rotation, translation, and shuffling of atom index do not change the properties of the molecule; hence descriptors used in NNPs need to be invariant to such transformations. We show that ANI, which uses manually constructed BPSF as a descriptor, outperforms all the models in most of the experiments. On the contrary, PhysNet, an end-to-end data-driven model that takes only distances and atom types as input without requiring additional hand engineering, is just a little behind ANI. The major disadvantage of ANI is that with the increase in the type of elements, the complexity of the descriptor increases exponentially. The message-passing NNPs complexity does not change with the increase in the type of elements by design.

Further, we apply the knowledge of atomic descriptors and NNP's architecture to develop ML and RL-based techniques for 3D structure generation of Gallium metal clusters and geometry optimization of small organic molecules. In this thesis, we introduce a novel descriptor called Topological Atomic Descriptor (TAD), which effectively encodes the Gallium clusters topology and is used to train the DART model, which predicts the energy of Gallium clusters. We show our DART model's ability to predict the energy of the Gallium cluster, demonstrating the trained network's robustness and TAD's adequacy as a feature vector compared to symmetry functions. We also demonstrate the ability of the DART model to accelerate the workflow of identifying low-energy structures and increase the probability of finding unique low-energy structures. Further, we develop a 3D structure generator model named as MeGen (**Me**tal cluster structure **Gen**eration). MeGen generates energetically low-lying 3D structures of Ga clusters in cartesian coordinates using Reinforcement Learning and DART as a reward

function. We show that MeGen is significantly more efficient than the conventional workflow for generating ground state geometries as well as low-lying isomers in terms of time and computational resources. Analysis of the Gallium clusters 3D structures generated using MeGen indicates that MeGen has learned the underlying cluster growth pattern. From 3D structure generation, we realized that the generator model does not always generate equilibrium structures, and we need to further optimize few of these structures to be sure. There are many algorithms available for MGO, but all these algorithms share one commonality: they are all hand-engineered – that is, human experts carefully design the steps of these algorithms. Just as deep learning has achieved tremendous success by automating feature engineering, automating algorithm design could open the way to similar performance gains. Hence using the Multi-agent reinforcement learning (MARL) approach, we developed MolOpt. We show the ability of our learned optimizer, MolOpt, to perform MGO on different classes of alkanes, demonstrating the transferability of our model to different molecules. We also compare MolOpt with other optimizers such as MDMin, FIRE, and BFGS. Further we developed MolOpt2 which is an improved version of MolOpt. In MolOpt2 we do improvements in the state representation and policy network. MolOpt2 is also trained on new training data set known as OptMol10 which has ten molecules containing elements CHNO. We evaluate MolOpt2 on OptQM9 test set which has molecules containing elements CHNO and upto nine heavy atoms.

Appendix A

Benchmark study on Deep Neural Network Potentials for Small Organic Molecules

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.48	0.59	0.67	0.72	0.5	0.59	1.01	0.65	5
PhysNet	0.45	0.59	0.7	0.72	0.42	0.61	1.03	0.66	3
Schnet	0.48	0.64	0.71	0.74	0.5	0.64	1.07	0.68	0
BAND-NN	0.55	0.66	0.76	0.74	0.58	0.64	1.13	0.71	0

Table A.1: Optimization of 8 decane structures using CauchyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.46	0.62	0.68	0.72	0.48	0.62	1.0	0.66	4
PhysNet	0.47	0.63	0.67	0.71	0.45	0.6	1.01	0.68	4
Schnet	0.57	0.65	0.74	0.73	0.48	0.72	1.08	0.7	0
BAND-NN	0.59	0.69	0.79	0.77	0.6	0.73	1.21	0.73	0

Table A.2: Optimization of 8 decane structures using Cobyla optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.48	0.63	0.69	0.68	0.5	0.62	1.02	0.66	3
PhysNet	0.48	0.63	0.7	0.69	0.49	0.63	1.0	0.64	5
Schnet	0.5	0.73	0.9	0.78	0.55	0.68	1.22	0.73	0
BAND-NN	0.58	0.8	0.93	0.84	0.57	0.69	1.25	0.75	0

Table A.3: Optimization of 8 decane structures using DiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.48	0.63	0.71	0.7	0.5	0.6	1.03	0.66	5
PhysNet	0.48	0.63	0.69	0.69	0.5	0.63	1.04	0.65	3
Schnet	0.57	0.66	0.77	0.93	0.54	0.63	1.04	0.8	0
BAND-NN	0.66	0.75	0.81	0.95	0.69	0.69	1.24	0.84	0

Table A.4: Optimization of 8 decane structures using DoubleFastGADiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.49	0.61	0.71	0.73	0.51	0.63	0.98	0.69	4
PhysNet	0.49	0.62	0.69	0.73	0.49	0.63	0.99	0.69	4
Schnet	0.56	0.69	0.77	0.75	0.62	0.72	1.04	0.78	0
BAND-NN	0.61	0.69	0.84	0.79	0.65	0.73	1.2	0.82	0

Table A.5: Optimization of 8 decane structures using MultiScaleCMA optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.49	0.63	0.69	0.73	0.51	0.63	1.04	0.69	5
PhysNet	0.49	0.54	0.71	0.73	0.48	0.63	1.04	0.69	3
Schnet	0.53	0.69	0.71	0.73	0.52	0.71	1.13	0.7	0
BAND-NN	0.62	0.72	0.84	0.73	0.57	0.71	1.22	0.7	0

Table A.6: Optimization of 8 decane structures using NelderMead optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.49	0.64	0.71	0.73	0.51	0.63	1.04	0.69	2
PhysNet	0.49	0.64	0.71	0.73	0.51	0.63	1.04	0.69	2
Schnet	0.5	0.64	0.61	0.73	0.5	0.63	1.04	0.65	3
BAND-NN	0.51	0.64	0.71	0.73	0.51	0.63	1.04	0.69	1

Table A.7: Optimization of 8 decane structures using NoisyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.47	0.62	0.69	0.73	0.49	0.62	1.04	0.68	7
PhysNet	0.49	0.63	0.7	0.73	0.49	0.62	1.02	0.69	1
Schnet	0.51	0.65	0.76	0.73	0.56	0.68	1.07	0.69	0
BAND-NN	0.53	0.7	0.77	0.74	0.57	0.7	1.09	0.73	0

Table A.8: Optimization of 8 decane structures using OnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.41	0.57	0.64	0.64	0.44	0.56	0.93	0.58	4
PhysNet	0.42	0.57	0.61	0.65	0.43	0.56	0.93	0.59	4
Schnet	0.53	0.65	0.66	0.72	0.47	0.64	1.1	0.69	0
BAND-NN	0.58	0.67	0.8	0.83	0.58	0.65	1.13	0.72	0

Table A.9: Optimization of 8 decane structures using Powell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.54	0.64	1.23	0.77	0.54	0.68	1.09	0.71	2
PhysNet	0.5	0.66	0.91	0.74	0.52	0.72	1.14	0.69	1
Schnet	0.51	0.65	1.21	0.72	0.53	0.62	1.21	0.72	2
BAND-NN	0.49	0.65	0.76	0.76	0.51	0.63	1.25	0.72	3

Table A.10: Optimization of 8 decane structures using QORandomSearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.5	0.64	0.72	0.73	0.51	0.65	1.07	0.74	4
PhysNet	0.5	0.64	0.72	0.75	0.52	0.64	1.43	0.69	4
Schnet	0.51	0.64	0.73	0.84	0.53	0.66	1.25	0.74	0
BAND-NN	0.76	0.65	0.73	0.88	0.55	1.03	1.4	0.74	0

Table A.11: Optimization of 8 decane structures using QOScrHammersleySearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.45	0.58	0.67	0.66	0.47	0.59	0.99	0.63	3
PhysNet	0.46	0.59	0.65	0.68	0.47	0.59	0.99	0.63	5
Schnet	0.56	0.62	0.71	0.7	0.53	0.65	1.1	0.69	0
BAND-NN	0.59	0.7	0.78	0.86	0.54	0.65	1.12	0.72	0

Table A.12: Optimization of 8 decane structures using chainCMAPowell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	0.49	0.63	0.71	0.73	0.51	0.62	1.04	0.69	4
PhysNet	0.49	0.63	0.7	0.73	0.48	0.63	1.04	0.63	4
Schnet	0.52	0.65	0.77	0.73	0.53	0.69	1.07	0.72	0
BAND-NN	0.6	0.69	0.78	0.74	0.6	0.7	1.14	0.73	0

Table A.13: Optimization of 8 decane structures using BFGS optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.23	5.18	4.42	4.4	5.56	6.54	6.8	5.08	4
PhysNet	5.47	5.5	4.31	4.92	6.85	6.37	7.42	5.61	2
Schnet	5.17	5.25	4.49	4.36	6.01	6.57	6.85	5.23	1
BAND-NN	4.99	5.42	4.49	4.37	6.09	6.64	7.0	5.26	1

Table A.14: Optimization of 8 fentanyl structures using CMandAS3 optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.02	5.13	4.29	4.37	5.32	6.31	6.91	5.1	5
PhysNet	5.04	7.66	4.3	4.38	5.47	7.14	6.91	5.41	0
Schnet	5.02	5.16	4.31	4.33	5.48	6.33	6.81	5.1	2
BAND-NN	5.01	5.31	4.32	4.34	6.02	6.33	6.91	5.24	1

Table A.15: Optimization of 8 fentanyl structures using CauchyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.18	5.11	4.34	4.33	5.51	6.44	6.92	5.1	6
PhysNet	5.61	7.55	4.31	6.01	6.46	6.45	9.83	5.83	1
Schnet	5.15	5.12	4.4	4.34	5.61	6.61	6.92	5.23	0
BAND-NN	5.1	5.51	4.54	4.42	5.93	6.61	6.92	5.24	1

Table A.16: Optimization of 8 fentanyl structures using Cobyla optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.35	5.21	4.44	4.38	5.55	6.56	6.79	5.15	7
PhysNet	5.78	7.01	5.15	5.78	6.63	9.44	8.21	5.6	0
Schnet	5.28	5.41	4.47	4.39	5.57	6.72	6.86	5.19	0
BAND-NN	5.04	5.42	4.79	4.44	5.7	7.0	6.95	5.26	1

Table A.17: Optimization of 8 fentanyl structures using DiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.33	5.25	4.52	4.39	5.61	6.55	6.94	5.2	5
PhysNet	5.9	7.42	5.26	4.89	6.66	10.09	7.75	5.8	0
Schnet	5.31	5.48	4.43	4.47	5.82	6.63	6.94	5.21	0
BAND-NN	5.08	5.74	4.37	4.5	5.85	6.67	6.93	5.21	3

Table A.18: Optimization of 8 fentanyl structures using DoubleFastGADiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.03	5.11	4.3	4.36	5.45	6.26	6.89	5.1	8
PhysNet	5.35	7.1	4.51	4.6	5.48	7.07	7.29	5.33	0
Schnet	5.04	5.28	4.32	4.39	5.91	6.29	6.9	5.18	0
BAND-NN	5.04	5.29	4.34	4.39	6.17	6.37	6.92	5.2	0

Table A.19: Optimization of 8 fentanyl structures using MultiScaleCMA optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.11	5.19	4.29	4.33	5.34	6.48	6.91	5.06	6
PhysNet	5.5	6.76	5.0	5.43	5.9	7.92	7.63	5.48	0
Schnet	5.06	5.2	4.32	4.35	5.37	6.4	6.96	5.14	0
BAND-NN	5.04	5.26	4.32	4.35	5.79	6.35	6.97	5.23	2

Table A.20: Optimization of 8 fentanyl structures using NelderMead optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.02	5.24	4.31	4.34	5.66	6.34	6.93	5.19	1
PhysNet	5.02	5.24	4.31	4.34	5.64	6.4	6.93	5.27	3
Schnet	4.92	5.17	4.31	4.34	5.67	6.34	6.92	5.19	3
BAND-NN	5.02	5.24	4.31	4.34	5.69	6.34	6.93	5.19	1

Table A.21: Optimization of 8 fentanyl structures using NoisyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.01	5.08	4.3	4.33	5.58	6.31	6.9	5.11	6
PhysNet	5.05	6.93	4.31	4.35	5.43	6.52	6.9	5.15	2
Schnet	5.02	5.25	4.31	4.33	5.67	6.39	6.92	5.19	0
BAND-NN	5.03	5.27	4.34	4.36	6.0	6.4	6.92	5.21	0

Table A.22: Optimization of 8 fentanyl structures using OnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.24	5.15	4.4	4.4	5.41	6.51	6.8	5.07	4
PhysNet	5.45	5.5	4.31	4.8	6.79	6.37	7.38	5.59	2
Schnet	5.17	5.15	4.45	4.4	6.04	6.56	6.96	5.15	0
BAND-NN	4.99	5.39	4.5	4.37	6.09	6.63	7.0	5.26	2

Table A.23: Optimization of 8 fentanyl structures using Powell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	7.23	5.24	4.86	5.74	5.77	8.51	49.42	43.41	3
PhysNet	5.58	6.02	10.72	6.53	7.85	6.6	71.79	22.4	1
Schnet	7.26	5.24	5.4	4.51	6.52	6.33	30.76	36.25	2
BAND-NN	7.28	5.55	8.29	4.9	6.97	6.34	10.27	12.97	2

Table A.24: Optimization of 8 fentanyl structures using QORandomSearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	12.09	5.41	21.32	4.64	5.67	7.7	57.16	22.45	1
PhysNet	55.27	5.5	11.94	5.65	5.68	6.47	16.87	5.29	3
Schnet	9.26	5.2	16.49	4.53	5.68	6.45	64.29	20.46	2
BAND-NN	5.88	5.23	12.61	4.5	5.69	9.06	72.82	14.17	2

Table A.25: Optimization of 8 fentanyl structures using QOScrHammersleySearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.27	5.21	4.48	4.45	5.52	6.59	6.86	5.13	4
PhysNet	5.61	5.49	4.31	4.51	5.63	6.36	7.45	5.55	1
Schnet	4.98	5.22	4.36	4.21	5.88	6.35	6.87	5.16	2
BAND-NN	5.01	5.25	4.35	4.35	6.13	6.33	6.94	5.18	1

Table A.26: Optimization of 8 fentanyl structures using chainCMAPowell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	5.03	5.22	4.3	4.34	5.62	6.36	6.9	5.15	6
PhysNet	5.12	7.98	4.31	8.3	8.12	8.93	7.32	5.53	0
Schnet	5.02	5.25	4.31	4.34	6.0	6.36	6.91	5.16	0
BAND-NN	5.02	5.27	4.31	4.35	6.17	6.36	6.92	5.23	2

Table A.27: Optimization of 8 fentanyl structures using BFGS optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.12	2.7	3.48	4.05	3.9	3.05	3.14	133.21	8
PhysNet	4.16	2.81	3.67	4.14	4.02	3.09	3.33	135.76	0
Schnet	4.24	2.99	3.9	4.16	4.08	3.31	3.38	134.48	0
BAND-NN	4.52	3.01	3.9	4.42	4.09	3.4	3.42	134.86	0

Table A.28: Optimization of 8 retinol structures using CMandAS3 optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.16	2.72	3.34	3.92	3.94	3.24	3.26	138.79	6
PhysNet	4.25	2.84	3.71	4.06	4.02	3.21	3.32	138.79	1
Schnet	4.28	2.84	3.82	3.99	4.03	3.32	3.31	138.77	0
BAND-NN	4.44	3.02	3.86	4.56	4.13	3.34	3.36	138.77	1

Table A.29: Optimization of 8 retinol structures using CauchyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.14	2.77	3.45	3.91	3.89	3.22	3.26	138.47	7
PhysNet	4.14	2.84	3.63	4.09	3.93	3.28	3.33	138.79	1
Schnet	4.27	2.96	3.49	4.01	3.91	3.29	3.39	138.61	0
BAND-NN	4.69	2.98	3.93	4.61	4.06	3.41	3.53	139.22	0

Table A.30: Optimization of 8 retinol structures using Cobyla optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.18	2.83	3.76	4.17	3.96	3.23	3.27	134.61	8
PhysNet	4.18	2.9	3.93	4.22	4.02	3.23	3.39	138.15	0
Schnet	4.26	2.91	3.81	4.36	4.04	3.36	3.47	135.9	0
BAND-NN	4.57	3.12	3.83	4.51	4.23	3.43	3.5	136.06	0

Table A.31: Optimization of 8 retinol structures using DiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.21	2.8	3.81	4.24	4.0	3.4	3.28	134.09	6
PhysNet	4.26	2.9	3.78	4.37	4.05	3.32	3.39	138.66	2
Schnet	4.34	2.98	3.85	4.39	4.11	3.49	3.41	135.13	0
BAND-NN	4.46	3.09	3.85	4.45	4.2	3.68	3.5	135.92	0

Table A.32: Optimization of 8 retinol structures using DoubleFastGADiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.2	2.76	3.59	4.05	3.94	3.18	3.29	138.79	6
PhysNet	4.15	2.85	3.78	4.05	3.95	3.22	3.31	138.8	1
Schnet	4.41	2.88	3.66	4.12	3.95	3.33	3.29	138.79	0
BAND-NN	4.47	2.96	3.85	4.51	4.05	3.36	3.33	138.79	1

Table A.33: Optimization of 8 retinol structures using MultiScaleCMA optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	3.99	2.77	3.44	3.96	3.96	3.04	3.18	136.69	8
PhysNet	4.01	2.84	3.79	3.99	3.98	3.18	3.3	138.78	0
Schnet	4.36	2.85	3.53	4.02	4.07	3.29	3.26	136.78	0
BAND-NN	4.45	2.95	3.86	4.49	4.1	3.37	3.35	136.79	0

Table A.34: Optimization of 8 retinol structures using NelderMead optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.38	2.94	3.85	4.37	4.03	3.36	3.33	138.79	6
PhysNet	4.39	2.95	3.9	4.37	4.03	3.34	3.33	138.8	2
Schnet	4.38	2.94	3.85	4.37	4.03	3.37	3.38	138.79	0
BAND-NN	4.39	2.95	3.85	4.37	4.03	3.37	3.39	138.79	0

Table A.35: Optimization of 8 retinol structures using NoisyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.18	2.81	3.66	4.16	3.96	3.31	3.3	138.79	6
PhysNet	4.29	2.87	3.87	4.25	3.98	3.3	3.31	138.79	2
Schnet	4.19	2.88	3.79	4.26	4.03	3.32	3.32	138.79	0
BAND-NN	4.4	2.97	3.85	4.47	4.04	3.34	3.32	138.79	0

Table A.36: Optimization of 8 retinol structures using OnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.12	2.69	3.48	4.04	3.91	3.05	3.14	133.24	8
PhysNet	4.15	2.82	3.68	4.15	4.02	3.07	3.33	136.34	0
Schnet	4.46	2.8	3.82	4.2	4.01	3.09	3.23	133.57	0
BAND-NN	4.51	3.04	3.9	4.42	4.09	3.38	3.42	134.84	0

Table A.37: Optimization of 8 retinol structures using Powell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.38	3.06	3.95	7.15	6.26	3.68	5.87	1892.83	3
PhysNet	4.72	2.98	4.19	4.37	13.03	4.93	9.34	2682.14	1
Schnet	4.42	3.0	4.01	6.25	4.3	3.67	5.94	1800.69	0
BAND-NN	4.47	2.98	4.03	4.5	4.25	3.62	6.05	1688.33	4

Table A.38: Optimization of 8 retinol structures using QORandomSearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	7.61	4.18	3.84	4.37	4.03	6.86	6.59	2033.73	4
PhysNet	4.46	4.87	3.89	4.37	4.22	3.38	3.88	2155.18	2
Schnet	6.67	4.58	3.84	4.4	5.08	5.85	4.74	1531.5	0
BAND-NN	5.38	4.61	3.84	4.4	7.28	3.9	3.52	135.78	2

Table A.39: Optimization of 8 retinol structures using QOScrHammersleySearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.19	2.72	3.61	4.11	3.95	3.12	3.17	134.53	8
PhysNet	4.19	2.86	3.74	4.2	4.02	3.12	3.33	136.66	0
Schnet	4.29	2.95	3.74	4.41	4.03	3.27	3.41	135.56	0
BAND-NN	4.52	3.05	3.81	4.45	4.07	3.34	3.42	135.64	0

Table A.40: Optimization of 8 retinol structures using chainCMAPowell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	4.3	2.94	3.84	4.23	4.01	3.36	3.33	138.79	1
PhysNet	4.3	2.94	3.8	4.29	4.01	3.35	3.32	138.79	6
Schnet	4.38	2.93	3.89	4.4	4.02	3.37	3.34	138.8	0
BAND-NN	4.42	2.93	3.9	4.52	4.04	3.37	3.36	138.8	1

Table A.41: Optimization of 8 retinol structures using BFGS optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.12	1.12	1.56	1.05	0.81	1.38	0.87	1.47	5
PhysNet	1.16	1.16	1.48	1.26	0.88	1.36	0.92	1.43	3
Schnet	1.12	1.38	1.63	1.18	0.84	1.76	0.95	1.52	0
BAND-NN	1.21	1.4	1.64	1.35	1.08	1.83	0.97	1.72	0

Table A.42: Optimization of 8 methamphetamine structures using CMandAS3 optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.01	1.13	1.49	1.0	0.83	1.5	1.05	1.23	5
PhysNet	1.07	1.06	1.62	1.16	0.84	1.27	0.97	1.36	3
Schnet	1.27	1.24	1.65	1.08	1.12	1.77	1.07	1.88	0
BAND-NN	1.36	1.39	1.95	1.39	1.13	1.78	1.11	2.03	0

Table A.43: Optimization of 8 methamphetamine structures using CauchyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.19	1.21	1.78	1.32	0.93	1.43	1.19	1.54	2
PhysNet	1.23	1.17	1.67	1.25	0.88	1.44	1.06	1.5	6
Schnet	1.45	1.26	1.78	1.46	1.08	1.65	1.16	1.89	0
BAND-NN	1.47	1.45	1.8	1.47	1.14	1.69	1.13	1.99	0

Table A.44: Optimization of 8 methamphetamine structures using Cobyla optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.33	1.17	1.58	1.38	0.92	1.47	0.97	1.51	3
PhysNet	1.24	1.28	1.56	1.29	0.94	1.52	0.95	1.48	5
Schnet	1.34	1.17	1.62	1.48	0.94	1.83	0.97	1.58	0
BAND-NN	1.35	1.32	1.62	1.52	0.94	1.84	0.97	1.69	0

Table A.45: Optimization of 8 methamphetamine structures using DiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.27	1.19	1.52	1.26	0.96	1.53	1.03	1.59	4
PhysNet	1.37	1.32	1.53	1.39	0.99	1.49	0.98	1.62	2
Schnet	1.34	1.37	1.51	1.32	0.94	1.73	1.03	1.75	2
BAND-NN	1.37	1.43	1.81	1.5	1.1	1.76	1.03	1.8	0

Table A.46: Optimization of 8 methamphetamine structures using DoubleFastGADiscreteOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.08	1.46	1.65	1.24	0.89	1.79	1.3	1.51	3
PhysNet	1.19	1.24	1.64	1.22	0.95	1.52	1.03	1.51	5
Schnet	1.2	1.45	1.77	1.27	1.09	1.79	1.14	1.68	0
BAND-NN	1.35	1.39	1.96	1.41	1.1	1.76	1.14	1.84	0

Table A.47: Optimization of 8 methamphetamine structures using MultiScaleCMA optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.08	1.37	1.68	1.08	0.97	1.9	1.27	1.46	2
PhysNet	1.06	1.13	1.62	1.11	0.93	1.61	1.02	1.5	6
Schnet	1.22	1.4	1.89	1.25	1.03	1.84	1.19	1.76	0
BAND-NN	1.34	1.43	1.96	1.46	1.12	1.79	1.16	1.91	0

Table A.48: Optimization of 8 methamphetamine structures using NelderMead optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.37	1.51	1.94	1.47	1.13	1.75	1.15	1.94	2
PhysNet	1.35	1.51	1.93	1.46	1.16	1.75	1.16	1.95	3
Schnet	1.37	1.5	1.95	1.47	1.14	1.74	1.16	1.94	2
BAND-NN	1.39	1.51	1.96	1.47	1.17	1.75	1.17	1.94	1

Table A.49: Optimization of 8 methamphetamine structures using NoisyOnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.1	1.44	1.63	1.25	0.96	1.8	1.19	1.53	4
PhysNet	1.21	1.35	1.78	1.23	1.09	1.55	1.07	1.59	4
Schnet	1.13	1.45	1.87	1.34	0.97	1.79	1.19	1.55	0
BAND-NN	1.31	1.48	2.08	1.49	1.16	1.73	1.18	1.96	0

Table A.50: Optimization of 8 methamphetamine structures using OnePlusOne optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.12	1.1	1.58	1.03	0.87	1.34	0.91	1.42	4
PhysNet	1.15	1.16	1.43	1.28	0.91	1.35	0.89	1.43	2
Schnet	1.11	1.2	1.64	1.02	1.03	1.66	0.95	1.59	2
BAND-NN	1.22	1.4	1.64	1.35	1.06	1.87	0.96	1.71	0

Table A.51: Optimization of 8 methamphetamine structures using Powell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.36	1.51	1.95	1.52	1.14	2.02	1.17	1.97	0
PhysNet	1.36	1.51	1.94	1.46	1.16	1.75	1.33	1.97	5
Schnet	1.37	1.51	1.95	1.49	1.13	1.95	1.16	1.93	3
BAND-NN	1.39	1.53	1.95	1.47	1.16	1.78	1.18	1.94	0

Table A.52: Optimization of 8 methamphetamine structures using QORandomSearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.35	1.75	2.22	1.47	1.16	1.79	1.18	1.91	0
PhysNet	1.38	1.53	2.01	1.54	1.15	1.75	1.2	1.96	1
Schnet	1.33	1.57	1.94	1.46	1.17	1.76	1.16	1.87	3
BAND-NN	1.35	1.52	1.98	1.46	1.24	1.75	1.16	1.9	4

Table A.53: Optimization of 8 methamphetamine structures using QOScrHammersleySearch optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.19	1.17	1.62	1.12	0.83	1.52	0.9	1.57	4
PhysNet	1.17	1.23	1.58	1.24	0.89	1.44	0.96	1.51	4
Schnet	1.21	1.23	1.67	1.22	0.98	1.78	0.91	1.58	0
BAND-NN	1.21	1.39	1.69	1.29	1.03	1.8	0.99	1.64	0

Table A.54: Optimization of 8 methamphetamine structures using chainCMAPowell optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Structures	1	2	3	4	5	6	7	8	Best count
ANI	1.23	1.44	1.81	1.41	1.15	1.74	1.08	1.43	1
PhysNet	1.2	1.42	1.36	1.34	0.96	1.61	0.95	1.5	7
Schnet	1.32	1.52	1.94	1.42	1.15	1.74	1.09	1.43	0
BAND-NN	1.33	1.53	1.97	1.47	1.14	1.75	1.16	1.81	0

Table A.55: Optimization of 8 methamphetamine structures using BFGS optimization method. Values shown are the differences between optimized DFT and ML energies (kcal/mol)

Appendix B

DART: Deep Learning Enabled Topological Interaction Model for Energy Prediction of Metal Clusters and its Application in Identifying Unique Low Energy Isomers

Model	DART (values in kcal/mol)		
	MAE	RMSE	# structures
Test case 1 (overall)	4.77	6.16	427
Test set Ga-46	10.13	10.81	48
Test set Ga-57	3.59	4.82	197
Test set Ga-60	4.63	5.77	182

Table B.1: Performance of DART model on first test case which has total of 427 structures in the test set consisting of Ga - 46, 57, and 60 size clusters.

Model	DART (values in kcal/mol)		
	MAE	RMSE	# structures
Test case 2 (overall)	4.76	6.02	436
Test set Ga-46	7.45	8.07	48
Test set Ga-57	4.15	5.59	197
Test set Ga-66	4.72	5.84	191

Table B.2: Performance of DART model on second test case which has total of 436 structures in the test set consisting of Ga - 46, 57, and 66 size clusters.

Model	DART (values in kcal/mol)		
	MAE	RMSE	# structures
Test case 3 (overall)	4.34	5.58	570
Test set Ga-57	3.69	4.78	197
Test set Ga-60	4.93	6.15	182
Test set Ga-66	4.45	5.77	191

Table B.3: Performance of DART model on third test case which has total of 570 structures in the test set consisting of Ga - 57, 60, and 66 size clusters.

Appendix C

MolOpt: Autonomous Molecular Geometry Optimization using Multi-Agent Reinforcement Learning

C.1 Variants of MolOpt

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	−1.53	1.87	0.13	0.12
Pentane	30	−2.19	2.53	0.13	0.08
Hexane	50	−2.52	1.86	0.15	0.07
Heptane	90	−3.52	2.19	0.18	0.07
Octane	180	−4.11	1.84	0.20	0.07
Overall	360	−3.51	2.12	0.18	0.08

Table C.1: Variant 1 geometry optimization performance on test set containing 360 structures. $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between optimized BFGS structure and optimized MolOpt structure.

		Before optimization		After optimization	
Molecule name	# Structures	Mean RMSD (Å)	STD RMSD (Å)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.26	0.05	0.13	0.12
Pentane	30	0.25	0.08	0.13	0.08
Hexane	50	0.23	0.07	0.15	0.07
Heptane	90	0.24	0.07	0.18	0.07
Octane	180	0.25	0.06	0.20	0.07
Overall	360	0.25	0.07	0.18	0.08

Table C.2: Variant 1 geometry optimization performance on test set containing 360 structures. We calculate all-atom root mean squared deviation (RMSD) between initial structures vs optimized BFGS structures shown under "Before optimization" and optimized BFGS structure vs optimized MolOpt structure shown under "After optimization".

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	−0.14	0.11	0.04	0.02
Pentane	30	−0.67	0.75	0.08	0.07
Hexane	50	−1.22	0.89	0.14	0.08
Heptane	90	−1.40	0.96	0.15	0.08
Octane	180	−2.12	1.33	0.18	0.08
Overall	360	−1.64	1.25	0.15	0.09

Table C.3: Variant 2 geometry optimization performance on the test set containing 360 structures. $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure.

		Before optimization		After optimization	
Molecule name	# Structures	Mean RMSD (Å)	STD RMSD (Å)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.26	0.05	0.04	0.02
Pentane	30	0.25	0.08	0.08	0.07
Hexane	50	0.23	0.07	0.14	0.08
Heptane	90	0.24	0.07	0.15	0.08
Octane	180	0.25	0.06	0.18	0.08
Overall	360	0.25	0.07	0.15	0.09

Table C.4: Variant 2 geometry optimization performance on test set containing 360 structures. We calculate all-atom root mean squared deviation (RMSD) between Initial structures vs optimized BFGS structures shown under "Before optimization" and optimized BFGS structure vs optimized MolOpt structure shown under "After optimization".

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	−0.74	0.71	0.08	0.04
Pentane	30	−0.85	0.99	0.08	0.06
Hexane	50	−1.25	1.29	0.12	0.08
Heptane	90	−1.24	1.00	0.14	0.07
Octane	180	−1.52	1.17	0.16	0.07
Overall	360	−1.34	1.14	0.14	0.07

Table C.5: Variant 3 geometry optimization performance on test set containing 360 structures. $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between optimized BFGS structure and optimized MolOpt structure.

		Before optimization		After optimization	
Molecule name	# Structures	Mean RMSD (Å)	STD RMSD (Å)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.26	0.05	0.08	0.04
Pentane	30	0.25	0.08	0.08	0.06
Hexane	50	0.23	0.07	0.12	0.08
Heptane	90	0.24	0.07	0.14	0.07
Octane	180	0.25	0.06	0.16	0.07
Overall	360	0.25	0.07	0.14	0.07

Table C.6: Variant 3 geometry optimization performance on test set containing 360 structures. We calculate all-atom root mean squared deviation (RMSD) between Initial structures vs optimized BFGS structures shown under "Before optimization" and optimized BFGS structure vs optimized MolOpt structure shown under "After optimization".

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	−0.29	0.61	0.04	0.04
Pentane	30	−0.31	0.57	0.07	0.08
Hexane	50	−0.51	0.46	0.11	0.08
Heptane	90	−0.85	0.73	0.12	0.07
Octane	180	−1.34	1.05	0.15	0.07
Overall	360	−0.99	0.94	0.13	0.08

Table C.7: Variant 4 geometry optimization performance on the test set containing 360 structures. $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure.

		Before optimization		After optimization	
Molecule name	# Structures	Mean RMSD (Å)	STD RMSD (Å)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.26	0.05	0.04	0.04
Pentane	30	0.25	0.08	0.07	0.08
Hexane	50	0.23	0.07	0.11	0.08
Heptane	90	0.24	0.07	0.12	0.07
Octane	180	0.25	0.06	0.15	0.07
Overall	360	0.25	0.07	0.13	0.08

Table C.8: Variant 4 geometry optimization performance on the test set containing 360 structures. We calculate all-atom root mean squared deviation (RMSD) between Initial structures vs optimized BFGS structures shown under "Before optimization" and optimized BFGS structure vs optimized MolOpt structure shown under "After optimization".

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	−0.01	0.01	0.01	0.00
Pentane	30	−0.06	0.06	0.04	0.07
Hexane	50	−0.24	0.32	0.08	0.07
Heptane	90	−0.43	0.41	0.10	0.07
Octane	180	−0.84	0.78	0.14	0.07
Overall	360	−0.57	0.67	0.11	0.08

Table C.9: Variant 5 geometry optimization performance on the test set containing 360 structures. $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure.

		Before optimization		After optimization	
Molecule name	# Structures	Mean RMSD (Å)	STD RMSD (Å)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.26	0.05	0.01	0.00
Pentane	30	0.25	0.08	0.04	0.07
Hexane	50	0.23	0.07	0.08	0.07
Heptane	90	0.24	0.07	0.10	0.07
Octane	180	0.25	0.06	0.14	0.07
Overall	360	0.25	0.07	0.11	0.08

Table C.10: Variant 5 geometry optimization performance on the test set containing 360 structures. We calculate all-atom root mean squared deviation (RMSD) between Initial structures vs optimized BFGS structures shown under "Before optimization" and optimized BFGS structure vs optimized MolOpt structure shown under "After optimization".

C.2 Benchmark of MolOpt

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	1.20	1.12	0.10	0.05
Pentane	30	1.14	1.13	0.08	0.06
Hexane	50	1.09	1.12	0.08	0.05
Heptane	90	1.05	1.18	0.08	0.05
Octane	180	0.85	0.96	0.07	0.04
Overall	360	0.97	1.06	0.08	0.04

Table C.11: Bench-marked variant 5 against MDMin (max number of steps = 300). $\Delta E = E_{MDMin} - E_{MolOpt}$ both E_{MDMin} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized MDMin structure and the optimized MolOpt structure.

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	-0.01	0.00	0.01	0.00
Pentane	30	-0.03	0.03	0.03	0.07
Hexane	50	-0.09	0.11	0.03	0.05
Heptane	90	-0.12	0.18	0.03	0.02
Octane	180	-0.27	0.30	0.05	0.04
Overall	360	-0.18	0.25	0.04	0.04

Table C.12: Bench-marked variant 5 against FIRE (max number of steps = 300). $\Delta E = E_{FIRE} - E_{MolOpt}$ both E_{FIRE} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized FIRE structure and the optimized MolOpt structure.

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	-0.01	0.00	0.01	0.00
Pentane	30	-0.05	0.04	0.04	0.07
Hexane	50	-0.21	0.30	0.07	0.07
Heptane	90	-0.40	0.39	0.10	0.07
Octane	180	-0.79	0.75	0.13	0.08
Overall	360	-0.53	0.64	0.10	0.08

Table C.13: Bench-marked variant 5 against BFGS (max number of steps = 300). $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure.

Optimizer	Time (s)
MDMin	425.88
FIRE	407.38
BFGS	256.20
MolOpt	605.05

Table C.14: Time taken in seconds by each optimizer to optimize 72 structures sampled at equal intervals from 360 structures test set.

Appendix D

MolOpt2: Autonomous Molecular Geometry Optimization using Multi-Agent Reinforcement Learning

Parameters	Value
Entropy coefficient	0.0001
KL coefficient	1.0
KL target	0.01
gamma	1.0
Clip parameter	0.3
vf clip parameter	10.0
Horizon	20
lr	5e-05
train batch size	2048

Table D.1: Hyperparameter used during training and evaluation of MolOpt

Molecule name	# Isomers × # structures	# Atoms
Propane	1 × 10	11
Pentane	3 × 10	17
Hexane	5 × 10	20
Heptane	9 × 10	23
Octane	18 × 10	26
Total	360	

Table D.2: Test set that contains 360 structures of alkanes (C_nH_{2n+2}) where n=3,5,6,7 and 8.

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.00	0.00	0.00	0.00
Pentane	30	−0.02	0.02	0.03	0.07
Hexane	50	−0.11	0.22	0.06	0.07
Heptane	90	−0.27	0.35	0.08	0.07
Octane	180	−0.53	0.68	0.11	0.08
overall	360	−0.35	0.55	0.09	0.08

Table D.3: Log transformed features trained on alkanes. MolOpt 2 geometry optimization performance on the test set containing 360 structures of alkanes. $\Delta E = E_{BFGS} - E_{MolOpt}$ both E_{BFGS} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized BFGS structure and the optimized MolOpt structure.

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	1.21	1.12	0.10	0.05
Pentane	30	1.18	1.14	0.09	0.06
Hexane	50	1.19	1.12	0.09	0.05
Heptane	90	1.18	1.18	0.09	0.04
Octane	180	1.12	1.02	0.09	0.04
overall	360	1.15	1.08	0.09	0.05

Table D.4: Log transformed features trained on alkanes. MolOpt 2 geometry optimization performance on the test set containing 360 structures of alkanes. $\Delta E = E_{MDmin} - E_{MolOpt}$ both E_{MDmin} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized MDmin structure and the optimized MolOpt structure.

Molecule name	# Structures	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
Propane	10	0.00	0.00	0.00	0.00
Pentane	30	0.01	0.01	0.02	0.07
Hexane	50	0.01	0.04	0.02	0.05
Heptane	90	0.01	0.10	0.02	0.01
Octane	180	−0.01	0.18	0.02	0.03
overall	360	0.00	0.14	0.02	0.04

Table D.5: Log transformed features trained on alkanes. MolOpt 2 geometry optimization performance on the test set containing 360 structures of alkanes. $\Delta E = E_{FIRE} - E_{MolOpt}$ both E_{FIRE} and E_{MolOpt} are optimized energies. We calculate the all-atom root mean squared deviation (RMSD) between the optimized FIRE structure and the optimized MolOpt structure.

# Atoms	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
7	0.00	0.00	0.00	0.00
9	0.00	0.01	0.01	0.01
10	−0.01	0.00	0.01	0.01
11	0.00	0.00	0.00	0.00
12	−0.01	0.01	0.01	0.00
13	0.03	0.08	0.01	0.02
14	−0.03	0.06	0.01	0.01
15	−0.01	0.09	0.02	0.02
16	−0.02	0.05	0.01	0.01
17	−0.01	0.07	0.02	0.01
18	−0.03	0.03	0.01	0.01
19	−0.03	0.05	0.01	0.01
20	−0.04	0.06	0.02	0.01
21	−0.04	0.04	0.01	0.01
22	−0.08	0.10	0.02	0.01
23	−0.07	0.12	0.02	0.02
24	−0.02	0.01	0.01	0.01
25	−0.06	0.04	0.02	0.00
overall	−0.03	0.06	0.01	0.01

Table D.6: Trained on all molecules(OptMol10 and alkanes), FIRE

# Atoms	Mean ΔE (kcal/mol)	STD ΔE (kcal/mol)	Mean RMSD (Å)	STD RMSD (Å)
7	−12.70	28.60	0.75	0.29
9	2.56	3.15	0.38	0.65
10	27.74	20.25	0.70	0.05
11	11.35	39.16	1.30	0.04
12	23.64	73.75	1.16	0.33
13	22.28	27.02	1.29	0.66
14	42.40	43.74	1.04	0.22
15	41.83	41.60	1.06	0.30
16	44.68	54.31	0.95	0.46
17	55.11	46.13	0.98	0.34
18	54.40	49.42	0.74	0.43
19	43.23	41.42	0.80	0.80
20	39.53	42.45	0.48	0.38
21	29.32	40.62	0.46	0.44
22	24.44	22.19	0.40	0.36
23	19.75	34.41	0.25	0.40
24	7.81	11.07	0.02	0.01
25	6.09	4.25	0.02	0.01
overall	39.69	44.76	0.74	0.53

Table D.7: Trained on all molecules(OptMol10 and alkanes), MDmin

Appendix E

List of Abbreviations

ACSF ..... Atom Centered Symmetry Function

ADAM ..... Adaptive Moment Estimation

AEV ..... atomic environment vector

AI ..... Artificial intelligence

ANN ..... artificial neural networks

AOI ..... atom of interest

ASE ..... Atomic Simulation Environment

BAND ..... bonds, angles, non-bonds and dihedrals

BFGS ..... Broyden-Fletcher-Goldfarb-Shanno

BO ..... Born-Oppenheimer

BPSF ..... Behler and Parrinello symmetry function

CCSD ..... coupled-cluster singles and doubles

CELU ..... Continuously Differentiable Exponential Linear Unit

CG ..... conjugate gradient

CM ..... Coulomb Matrix

CMA ..... Covariance matrix adaptation

CoM ..... center of mass

DFT ..... density functional theory

DL ..... deep learning

DPMD ..... Deep Potential Molecular Dynamics

ES ..... evolutionary strategies

FF ..... force fields

FFNN ..... feed-forward neural networks

FIRE ..... Fast inertial relaxation engine

GAN ..... Generative Adversarial Networks

GGA ..... generalized gradient approximation

GMM ..... Gaussian mixture model

GNN ..... Graph Neural Network

GS ..... Ground State

HDF ..... High dimensional function

HDNN ..... high dimensional atomic NN

HF ..... Hartree-Fock

HIP-NN ..... Hierarchically interacting particle neural network

LDA ..... local density approximation

LSTM ..... long short-term memory network

MAE ..... Mean Absolute Error

MARL ..... multi-agent reinforcement learning

MD ..... molecular dynamics

MDN ..... mixture density network

MDP ..... Markov decision process

MGO ..... molecular geometry optimizer

ML ..... Machine learning

MLP ..... multi-layered perceptrons

MM ..... molecular mechanics

MMDP ..... Multi-agent MDP

NLP ..... natural language processing

NNP ..... neural network potential

NMR ..... nuclear magnetic resonance

PBE ..... Perdew-Burke-Ernzerhof

PES ..... potential energy surface

POSG ..... Partially-Observable Stochastic Games

PPO ..... policy optimization

PRDF ..... Partial Radial Distribution Function

QM ..... Quantum mechanical

RL ..... reinforcement learning

RMSD ..... root mean squared deviation

RMSE ..... Root Mean Squared Error

ReLU ..... Rectified Linear Unit

SCF ..... self-consistent field

SD ..... steepest descent

SE ..... "Schr"odinger equation"

SGD ..... Stochastic gradient descent

SOAP ..... Smooth Overlap of Atomic Positions

TAD ..... Topological Atomic Descriptor

TBPTT ..... truncated backpropagation through time

t-SNE ..... t-Distributed Stochastic Neighbor Embedding

Related Publications

Published

1. R. Modee, S. Mehta, S. Laghuvarapu and U. D. Priyakumar, MolOpt: Autonomous Molecular Geometry Optimization Using Multiagent Reinforcement Learning, J. Phys. Chem. B, 2023, 127, 10295–10303.

2. R. Modee, A. Verma, K. Joshi and U. Deva Priyakumar, MeGen - generation of gallium metal clusters using reinforcement learning, Mach. Learn. Sci. Technol., 2023, 4, 025032.

3. D. B. Korlepara, C. S. Vasavi, S. Jeurkar, P. K. Pal, S. Roy, S. Mehta, S. Sharma, V. Kumar, C. Muvva, B. Sridharan, A. Garg, R. Modee, A. P. Bhati, D. Nayar and U. D. Priyakumar, PLAS-5k: Dataset of Protein-Ligand Affinities from Molecular Dynamics for Machine Learning Applications, Sci. Data, 2022, 9, 1–10.

4. R. Modee, S. Laghuvarapu and U. D. Priyakumar, Benchmark study on deep neural network potentials for small organic molecules, J. Comput. Chem., 2022, 43, 308–318.

5. R. Modee, S. Agarwal, A. Verma, K. Joshi and U. D. Priyakumar, DART: Deep learning enabled topological interaction model for energy prediction of metal clusters and its application in identifying unique low energy isomers, Phys. Chem. Chem. Phys., 2021, 23, 21995–22003.

Unpublished

1. R Modee, U. D. Priyakumar. MolOpt2: Autonomous Molecular Geometry Optimization using Multi-Agent Reinforcement Learning. (to be submitted)

2. B Sridharan, A Sinha, J Bardhan, R Modee, M Ehara, and U. D. Priyakumar, Deep Reinforcement Learning in Chemistry: A Review. (Submitted)

Scientific Meets

Conference - Talk

1. Machine Learning for Molecular Sciences (ML4Science), Kodaikanal, India, 2023. Talk on "MeGen: Generation of Gallium Metal Clusters Using Reinforcement Learning."
2. Molecular Simulation: Focus on Method, Tata Institute of Fundamental Research (TIFR), Hyderabad , India, 2022. Talk on "OptNet: Molecular Geometry Optimization Using Reinforcement Learning."

Conference - Poster

1. Designing Catalysts on Computers (DCC), Indian Association for the Cultivation of Science (IACS), Kolkata, West Bengal, India, 2022. Presented poster titled:- "OptNet: Molecular Geometry Optimization Using Reinforcement Learning."
2. The virtual 3rd RSC-BMCS / RSC-CICAG Artificial Intelligence in Chemistry, 2020. Presented poster titled:- "Neural network potentials for representing potential energy surface and their applicability for geometry optimization."
3. Workshop and Symposium on Advanced Simulation Methods: DFT, MD and Beyond. Indian Institute of Technology (IIT) Delhi, New Delhi, India, 2019. Presented poster titled:- "Cosolvents Effect on Protein (Un)Folding Equilibrium."
4. Indian Peptide Society (IPS) 7th Symposium. Birla Institute of Technology and Science (BITS) Pilani, Hyderabad, India, 2019. Presented poster titled:- "Synergistic Play of Cosolvents in Protein Folding/Unfolding Equilibrium."

Conference - Attended

1. Machine learning how to coarse-grain, CECAM, (Virtual) 2020. Co-organised by Dr. Denis Andrienko (Max Planck Institute for Polymer Research) and Dr. Tristan Bereau (University of Amsterdam).

2. Machine Learning For Science (ML4Science). International Institute of Information Technology (IIIT), Hyderabad, India, 2019.

Award/Achievements

1. Awarded the prize for best method in a machine learning hackathon focused on property prediction using SMILES at the ML4Science-2023 Meeting.
2. Awarded four year scholarship for Ph.D. by TCS Research Scholar Program (RSP) 2019.
3. Qualified GATE-2015 in biotechnology with 85.54 percentile (All India Rank (AIR) - 1290).

Bibliography

[1] P. A. M Dirac. Quantum mechanics of many-electron systems. *Proc. R. Soc. London. Ser. A, Contain. Pap. a Math. Phys. Character*, 123(792):714–733, apr 1929. doi: 10.1098/rspa.1929.0094.

[2] Oliver T. Unke and Markus Meuwly. A reactive, scalable, and transferable model for molecular energies from a neural network approach based on local information. *J. Chem. Phys.*, 148(24):241708, jun 2018. doi: 10.1063/1.5017898.

[3] Jonas Danielsson and Markus Meuwly. Atomistic simulation of adiabatic reactive processes based on multi-state potential energy surfaces. *J. Chem. Theory Comput.*, 4(7): 1083–1093, 2008. doi: 10.1021/ct800066q.

[4] Tibor Nagy, Juvenal Yosa Reyes, and Markus Meuwly. Multisurface adiabatic reactive molecular dynamics. *J. Chem. Theory Comput.*, 10(4):1366–1375, 2014. doi: 10.1021/ct400953f.

[5] Bernd Hartke and Stefan Grimme. Reactive force fields made simple. *Phys. Chem. Chem. Phys.*, 17(26):16715–16718, oct 2015. doi: 10.1039/c5cp02580j.

[6] Tobias Morawietz and Nongnuch Artrith. Machine learning-accelerated quantum mechanics-based atomistic simulations for industrial applications. *J. Comput. Aided. Mol. Des.*, 35(4):557–586, 2021. ISSN 15734951.

[7] Justin S. Smith, Olexandr Isayev, and Adrian E. Roitberg. Data Descriptor: ANI-1, A data set of 20 million calculated off-equilibrium conformations for organic molecules. *Sci. Data*, 4:170193, 2017. doi: 10.1038/sdata.2017.193.

[8] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, 2014. ISSN 20524463. doi: 10.1038/sdata.2014.22.

[9] V. Prathyusha, S. Ramakrishna, and U. Deva Priyakumar. Transannular Diels-Alder reactivities of 14-membered macrocylic trienes and their relationship with the conformational preferences of the reactants: A combined quantum chemical and molecular dynamics study. *J. Org. Chem.*, 77(12):5371–5380, 2012. ISSN 00223263. doi: 10.1021/jo300812q.

[10] Nitish Alodia, Tanashree Jaganade, and U. Deva Priyakumar. Quantum mechanical investigation of the nature of nucleobase-urea stacking interaction, a crucial driving force in RNA unfolding in aqueous urea. *J. Chem. Sci.*, 130(11):1–13, 2018. ISSN 09737103. doi: 10.1007/s12039-018-1563-8. URL `https://doi.org/10.1007/s12039-018-1563-8`.

[11] Shampa Raghunathan, Komal Yadav, V. C. Rojisha, Tanashree Jaganade, V. Prathyusha, Swetha Bikkina, Upakarasamy Lourderaj, and U. Deva Priyakumar. Transition between [: R] - And [S] -stereoisomers without bond breaking. *Phys. Chem. Chem. Phys.*, 22(26): 14983–14991, 2020. ISSN 14639076. doi: 10.1039/d0cp02918a. URL `http://xlink.rsc.org/?DOI=D0CP02918A`.

[12] Navneet Bung, Arijit Roy, U. Deva Priyakumar, and Gopalakrishnan Bulusu. Computational modeling of the catalytic mechanism of hydroxymethylbilane synthase. *Phys. Chem. Chem. Phys.*, 21(15):7932–7940, 2019. ISSN 14639076. doi: 10.1039/c9cp00196d.

[13] U. Deva Priyakumar, G. Narahari Sastry, and Goverdhan Mehta. Development of predictive models of π-facial selectivity; a critical study of nucleophilic addition to sterically unbiased ketones. *Tetrahedron*, 60(15):3465–3472, 2004. ISSN 00404020. doi: 10.1016/j.tet.2004.02.026.

[14] U. Deva Priyakumar and G. Narahari Sastry. Theoretical study of silabenzene and its valence isomers. *Organometallics*, 21(7):1493–1499, 2002. ISSN 02767333. doi: 10.1021/om011001i.

[15] W. Kohn and L. J. Sham. Self-consistent equations including exchange and correlation effects. *Phys. Rev.*, 140(4A):A1133, 1965. ISSN 0031899X. doi: 10.1103/PhysRev.140.A1133.

[16] M. J. Seaton. Hartree–Fock method. *Nature*, 269(5629):631–631, 1977. ISSN 0028-0836. doi: 10.1038/269631a0. URL `https://doi.org/10.1038/269631a0`.

[17] Michael J.S. Dewar, Eve G. Zoebisch, Eamonn F. Healy, and James J.P. Stewart. AM1: A New General Purpose Quantum Mechanical Molecular Model1. *J. Am. Chem. Soc.*, 107(13):3902–3909, 1985. ISSN 15205126. doi: 10.1021/ja00299a024.

[18] Michael J.S. Dewar and Walter Thiel. Ground States of Molecules. 38. The MNDO Method. Approximations and Parameters. *J. Am. Chem. Soc.*, 99(15):4899–4907, 1977. ISSN 15205126. doi: 10.1021/ja00457a004.

[19] James J.P. Stewart. Optimization of parameters for semiempirical methods I. Method. *J. Comput. Chem.*, 10(2):209–220, 1989. ISSN 1096987X. doi: 10.1002/jcc.540100208.

[20] Anders S. Christensen, Tomáš Kubař, Qiang Cui, and Marcus Elstner. Semiempirical Quantum Mechanical Methods for Noncovalent Interactions for Chemical and Biochemical Applications. *Chem. Rev.*, 116(9):5301–5337, 2016. ISSN 15206890. doi: 10.1021/acs.chemrev.5b00584.

[21] W. Kohn, A. D. Becke, and R. G. Parr. Density functional theory of electronic structure. *J. Phys. Chem.*, 100(31):12974–12980, 1996. ISSN 00223654. doi: 10.1021/jp960669l. URL `https://doi.org/10.1021/jp960669l`.

[22] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77(18):3865–3868, 1996. ISSN 10797114. doi: 10.1103/PhysRevLett.77.3865.

[23] A. D. Becke. Density-functional exchange-energy approximation with correct asymptotic behavior. *Phys. Rev. A*, 38(6):3098–3100, sep 1988. ISSN 10502947. doi: 10.1103/PhysRevA.38.3098. URL `https://link.aps.org/doi/10.1103/PhysRevA.38.3098`.

[24] Chengteh Lee, Weitao Yang, and Robert G. Parr. Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. *Phys. Rev. B*, 37 (2):785–789, jan 1988. ISSN 0163-1829. doi: 10.1103/PhysRevB.37.785. URL `https://link.aps.org/doi/10.1103/PhysRevB.37.785`.

[25] Burkhard Miehlich, Andreas Savin, Hermann Stoll, and Heinzwerner Preuss. Results obtained with the correlation energy density functionals of becke and Lee, Yang and Parr. *Chem. Phys. Lett.*, 157(3):200–206, may 1989. ISSN 00092614. doi: 10.1016/0009-2614(89)87234-3. URL `https://linkinghub.elsevier.com/retrieve/pii/0009261489872343`.

[26] Jianmin Tao, John P. Perdew, Viktor N. Staroverov, and Gustavo E. Scuseria. Climbing the density functional ladder: Nonempirical meta–generalized gradient approximation designed for molecules and solids. *Phys. Rev. Lett.*, 91(14):3–6, 2003. ISSN 10797114. doi: 10.1103/PhysRevLett.91.146401.

[27] Yan Zhao and Donald G. Truhlar. A new local density functional for main-group thermochemistry, transition metal bonding, thermochemical kinetics, and noncovalent interactions. *J. Chem. Phys.*, 125(19), 2006. ISSN 00219606. doi: 10.1063/1.2370993.

[28] John P. Perdew, Adrienn Ruzsinszky, Gábor I. Csonka, Lucian A. Constantin, and Jianwei Sun. Workhorse semilocal density functional for condensed matter physics and quantum chemistry. *Phys. Rev. Lett.*, 103(2):10–13, 2009. ISSN 00319007. doi: 10.1103/PhysRevLett.103.026403.

[29] Frank Jensen. Atomic orbital basis sets. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 3(3): 273–295, 2013. ISSN 17590876. doi: 10.1002/wcms.1123.

[30] Peter W Atkins and Ronald S Friedman. *Molecular quantum mechanics*. Oxford university press, 2011.

[31] Frank Jensen. *Introduction to Computational Chemistry*. John wiley & sons, 2007. ISBN 0470058048. doi: 10.1007/s00214-013-1372-6. URL `https://books.google.com/books/about/Introduction{_}to{_}Computational{_}Chemistry.html?id=RDIG48UcZfYC{&}pgis=1`.

[32] A. Rahman. Correlations in the motion of atoms in liquid argon. *Phys. Rev.*, 136:A405–A411, Oct 1964. doi: 10.1103/PhysRev.136.A405. URL `https://link.aps.org/doi/10.1103/PhysRev.136.A405`.

[33] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T.K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiórkiewicz-Kuczera, D. Yin, and M. Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102(18):3586–3616, 1998. ISSN 15206106. doi: 10.1021/jp973084f.

[34] Alexander D. Mackerell. Empirical force fields for biological macromolecules: Overview and issues. *J. Comput. Chem.*, 25(13):1584–1604, 2004. ISSN 01928651. doi: 10.1002/jcc.20082.

[35] Stewart A. Adcock and J. Andrew McCammon. Molecular dynamics: Survey of methods for simulating the activity of proteins. *Chem. Rev.*, 106(5):1589–1615, 2006. ISSN 00092665. doi: 10.1021/cr040426m.

[36] Andrew R. Leach. Empirical Force Field Models: Molecular Mechanics. In *Mol. Model. Princ. Appl.*, chapter 4, pages 165–252. Prentice Hall, 2001. ISBN 0582382106.

[37] Francisco Rodríguez-Ropero, Philipp Rötzscher, and Nico F.A. Van Der Vegt. Comparison of Different TMAO Force Fields and Their Impact on the Folding Equilibrium of a Hydrophobic Polymer. *J. Phys. Chem. B*, 120(34):8757–8767, sep 2016. ISSN 15205207. doi: 10.1021/acs.jpcb.6b04100. URL `https://pubs.acs.org/doi/10.1021/acs.jpcb.6b04100`.

[38] Deepak R. Canchi, Dietmar Paschek, and Angel E. Garcia. Equilibrium study of protein denaturation by urea. *J. Am. Chem. Soc.*, 132(7):2338–2344, 2010. ISSN 00027863. doi: 10.1021/ja909348c.

[39] Dietmar Paschek and Angel E. García. Reversible temperature and pressure denaturation of a protein fragment: A replica exchange molecular dynamics simulation study. *Phys. Rev. Lett.*, 93(23):10–13, 2004. ISSN 00319007. doi: 10.1103/PhysRevLett.93.238105. URL https://link.aps.org/doi/10.1103/PhysRevLett.93.238105.

[40] Durba Sengupta, Xavier Prasanna, Madhura Mohole, and Amitabha Chattopadhyay. Exploring GPCR-Lipid Interactions by Molecular Dynamics Simulations: Excitements, Challenges, and the Way Forward. *J. Phys. Chem. B*, 122(22):5727–5737, 2018. ISSN 15205207. doi: 10.1021/acs.jpcb.8b01657.

[41] Shweta Kumari, Abhijit Mitra, and Gopalakrishnan Bulusu. Structural dynamics of Smoothened (SMO) in the ciliary membrane and its interaction with membrane lipids. *Biochim. Biophys. Acta - Biomembr.*, 1864(8):183946, 2022. ISSN 18792642. doi: 10.1016/j.bbamem.2022.183946. URL https://doi.org/10.1016/j.bbamem.2022.183946.

[42] Shweta Kumari, Abhijit Mitra, and Gopalakrishnan Bulusu. Putative Role of Cholesterol in Shaping the Structural and Functional Dynamics of Smoothened (SMO). *J. Phys. Chem. B*, 127(44):9476–9495, nov 2023. ISSN 1520-6106. doi: 10.1021/acs.jpcb.3c02255. URL https://pubs.acs.org/doi/10.1021/acs.jpcb.3c02255.

[43] Loup Verlet. Computer "experiments" on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Phys. Rev.*, 159(1):98–103, jul 1967. ISSN 0031899X. doi: 10.1103/PhysRev.159.98. URL https://link.aps.org/doi/10.1103/PhysRev.159.98.

[44] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation: From Algorithms to Applications, Third Edition*. Elsevier, 2023. ISBN 9780323902922. doi: 10.1016/C2009-0-63921-0.

[45] William C. Swope, Hans C. Andersen, Peter H. Berens, and Kent R. Wilson. A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters. *J. Chem. Phys.*, 76 (1):637–649, 1982. ISSN 00219606. doi: 10.1063/1.442716.

[46] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5(4):115–133, dec 1943. ISSN 00074985. doi: 10.1007/BF02478259. URL http://link.springer.com/10.1007/BF02478259.

[47] Steven Walczak and Narciso Cerpa. Artificial Neural Networks. In Robert A Meyers, editor, *Encycl. Phys. Sci. Technol.*, pages 631–645. Elsevier, New York, third edit edition, 2003. ISBN 978-0-12-227410-7. doi: 10.1016/B0-12-227410-5/00837-1.

[48] Nishant Kumar and Martin Raubal. Applications of deep learning in congestion detection, prediction and alleviation: A survey. *Transp. Res. Part C Emerg. Technol.*, 133 (November):103432, 2021. ISSN 0968090X. doi: 10.1016/j.trc.2021.103432.

[49] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Adv. Neural Inf. Process. Syst.*, 2, 1989. ISSN 15244725. doi: 10.1111/dsu.12130.

[50] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, nov 1997. ISSN 0899-7667.

[51] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. GAN Generative Adversarial Nets . *J. Japan Soc. Fuzzy Theory Intell. Informatics*, 29(5):177–177, oct 2017. doi: 10.3156/jsoft.29.5_177_2.

[52] Cheng Yuan Liou, Wei Chen Cheng, Jiun Wei Liou, and Daw Ran Liou. Autoencoder for words. *Neurocomputing*, 139:84–96, 2014. ISSN 18728286. doi: 10.1016/j.neucom.2013.09.055.

[53] Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using variational autoencoders. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 11139 LNCS:412–422, 2018.

[54] Matt J. Kusner, Brooks Paige, and José Miguel Hemández-Lobato. Grammar variational autoencoder. *34th Int. Conf. Mach. Learn. ICML 2017*, 4:3072–3084, 2017.

[55] Elman Mansimov, Omar Mahmood, Seokho Kang, and Kyunghyun Cho. Molecular Geometry Prediction using a Deep Generative Graph Neural Network. *Sci. Rep.*, 9(1):1–13, 2019.

[56] Weitao Du, He Zhang, Yuanqi Du, Qi Meng, Wei Chen, Nanning Zheng, Bin Shao, and Tie Yan Liu. SE(3) Equivariant Graph Neural Networks with Complete Local Frames. *Proc. Mach. Learn. Res.*, 162(3):5583–5608, 2022. ISSN 26403498.

[57] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, 2018.

[58] Sina Stocker, Johannes Gasteiger, Florian Becker, Stephan Günnemann, and Johannes T Margraf. How robust are modern graph neural network potentials in long and hot molecular dynamics simulations? *Mach. Learn. Sci. Technol.*, 3(4):045010, 2022. ISSN 26322153. doi: 10.1088/2632-2153/ac9955.

[59] Gregor N. C. Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-Aware Actor-Critic for 3D Molecular Design. *arXiv*, pages 1–18, nov 2020.

[60] Rohit Modee, Ashwini Verma, Kavita Joshi, and U Deva Priyakumar. MeGen - Generation of gallium metal clusters using Reinforcement Learning. *Mach. Learn. Sci. Technol.*, 4(2):025032, jun 2023. ISSN 2632-2153.

[61] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[62] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2016-December:770–778, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.90.

[63] Dongxian Wu, Yisen Wang, Shu Tao Xia, James Bailey, and Xingjun Ma. Skip Connections Matter: on the Transferability of Adversarial Examples Generated With Resnets. *8th Int. Conf. Learn. Represent. ICLR 2020*, pages 1–15, 2020.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Adv. Neural Inf. Process. Syst.*, 2017-Decem(Nips):5999–6009, 2017. ISSN 10495258.

[65] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014. ISSN 15337928.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Commun. ACM*, volume 60, pages 84–90, 2017.

[67] Esben Jannik Bjerrum, Christian Margreitter, Thomas Blaschke, Simona Kolarova, and Raquel López Ríos de Castro. Faster and more diverse de novo molecular optimization with double-loop reinforcement learning using augmented SMILES. *J. Comput. Aided. Mol. Des.*, 37(8):373–394, 2023. ISSN 15734951. doi: 10.1007/s10822-023-00512-6.

[68] Justin S. Smith, Benjamin T. Nebgen, Roman Zubatyuk, Nicholas Lubbers, Christian Devereux, Kipton Barros, Sergei Tretiak, Olexandr Isayev, and Adrian E. Roitberg. Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning. *Nat. Commun.*, 10(1):1–8, 2019. ISSN 20411723.

[69] Erik Bitzek, Pekka Koskinen, Franz Gähler, Michael Moseler, and Peter Gumbsch. Structural relaxation made simple. *Phys. Rev. Lett.*, 97(17):1–4, 2006. ISSN 00319007.

[70] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand. (1934).*, 49(6):409, 1952. ISSN 0091-0635.

[71] Ask Hjorth Larsen, Jens JØrgen Mortensen, Jakob Blomqvist, Ivano E. Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N. Groves, BjØrk Hammer, Cory Hargus, Eric D. Hermes, Paul C. Jennings, Peter Bjerre Jensen, James Kermode, John R. Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob SchiØtz, Ole Schütt, Mikkel Strange, Kristian S. Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng, and Karsten W. Jacobsen. The atomic simulation environment - A Python library for working with atoms. *J. Phys. Condens. Matter*, 29(27):273002, jul 2017. ISSN 1361648X.

[72] C. G. Broyden. The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA J. Appl. Math. (Institute Math. Its Appl.*, 6(1):76–90, 1970. ISSN 02724960.

[73] FLETCHER R. New approach to variable metric algorithms. *Comput. J.*, 13(3):317–322, 1970. ISSN 00104620.

[74] Donald Goldfarb. A Family of Variable-Metric Methods Derived by Variational Means. *Math. Comput.*, 24(109):23, 1970. ISSN 00255718.

[75] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Math. Comput.*, 24(111):647, 1970. ISSN 00255718.

[76] Matthias Rupp, O. Anatole Von Lilienfeld, and Kieron Burke. Guest Editorial: Special Topic on Data-Enabled Theoretical Chemistry. *J. Chem. Phys.*, 148(24):241401, jun 2018. doi: 10.1063/1.5043213.

[77] Asher Mullard. The drug-maker's guide to the galaxy. *Nature*, 549(7673):445–447, sep 2017. doi: 10.1038/549445a.

[78] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement Learning: A Survey. *J. Artif. Intell. Res.*, 4(19):237–285, may 1996. doi: 10.1613/jair.301.

[79] Purdue University Marion Dwain Waltz. *A study of learning control system using a reinforment technique*. PhD thesis, Purdue University, 1964.

[80] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.*, 4(1):120–131, jan 2018. doi: 10.1021/acscentsci.7b00512.

[81] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv*, may 2017.

[82] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence generative adversarial nets with policy gradient. *31st AAAI Conf. Artif. Intell.*, pages 2852–2858, sep 2017.

[83] Bastiaan J. Braams and Joel M. Bowman. Permutationally invariant potential energy surfaces in high dimensionality. *Int. Rev. Phys. Chem.*, 28(4):577–606, oct 2009. doi: 10.1080/01442350903234923.

[84] J. S. Smith, O. Isayev, and A. E. Roitberg. ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical Science*, 8(4):3192–3203, 2017.

[85] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98(14):146401, 2007.

[86] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Phys. Rev. B - Condens. Matter Mater. Phys.*, 87(18):184115, may 2013. doi: 10.1103/PhysRevB.87.184115.

[87] Evgeny V. Podryabinkin and Alexander V. Shapeev. Active learning of linearly parametrized interatomic potentials. *Comput. Mater. Sci.*, 140:171–180, dec 2017. doi: 10.1016/j.commatsci.2017.08.031.

[88] Evgeny V. Podryabinkin, Evgeny V. Tikhonov, Alexander V. Shapeev, and Artem R. Oganov. Accelerating crystal structure prediction by machine-learning interatomic potentials with active learning. *Phys. Rev. B*, 99(6):064114, feb 2019. doi: 10.1103/PhysRevB.99.064114.

[89] Stefan Chmiela, Alexandre Tkatchenko, Huziel E. Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Sci. Adv.*, 3(5):e1603015, may 2017. doi: 10.1126/sciadv.1603015.

[90] Stefan Chmiela, Huziel E. Sauceda, Klaus-Robert Müller, and Alexandre Tkatchenko. Towards exact molecular dynamics simulations with machine-learned force fields. *Nat. Commun.*, 9(1):3887, dec 2018. doi: 10.1038/s41467-018-06169-2.

[91] Kevin Ryczko, David A. Strubbe, and Isaac Tamblyn. Deep learning and density-functional theory. *Phys. Rev. A*, 100(2):022512, aug 2019. doi: 10.1103/PhysRevA.100.022512.

[92] Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus Robert Müller. Bypassing the Kohn-Sham equations with machine learning. *Nat. Commun.*, 8 (1):872, dec 2017. doi: 10.1038/s41467-017-00839-3.

[93] James Lyons, Abdollah Dehzangi, Rhys Heffernan, Alok Sharma, Kuldip Paliwal, Abdul Sattar, Yaoqi Zhou, and Yuedong Yang. Predicting backbone Cα angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network. *J. Comput. Chem.*, 35(28):2040–2046, oct 2014. doi: 10.1002/jcc.23718.

[94] Qian Jiang, Xin Jin, Shin Jye Lee, and Shaowen Yao. Protein secondary structure prediction: A survey of the state of the art. *J. Mol. Graph. Model.*, 76:379–402, sep 2017. doi: 10.1016/j.jmgm.2017.07.015.

[95] Jack Hanson, Kuldip Paliwal, Thomas Litfin, Yuedong Yang, and Yaoqi Zhou. Improving prediction of protein secondary structure, backbone angles, solvent accessibility and contact numbers by using predicted contact maps and an ensemble of recurrent and residual convolutional neural networks. *Bioinformatics*, 35(14):2403–2410, jul 2019. doi: 10.1093/bioinformatics/bty1006.

[96] Sandra Romero-Molina, Yasser B. Ruiz-Blanco, Mirja Harms, Jan Münch, and Elsa Sanchez-Garcia. PPI-Detect: A support vector machine model for sequence-based prediction of protein–protein interactions. *J. Comput. Chem.*, 40(11):1233–1242, apr 2019. doi: 10.1002/jcc.25780.

[97] Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, jul 2018. doi: 10.1038/s41586-018-0337-2.

[98] Paul Raccuglia, Katherine C. Elbert, Philip D.F. Adler, Casey Falk, Malia B. Wenny, Aurelio Mollo, Matthias Zeller, Sorelle A. Friedler, Joshua Schrier, and Alexander J. Norquist. Machine-learning-assisted materials discovery using failed experiments. *Nature*, 533(7601):73–76, 2016. doi: 10.1038/nature17439.

[99] Logan Ward, Ankit Agrawal, Alok Choudhary, and Christopher Wolverton. A general-purpose machine learning framework for predicting properties of inorganic materials. *npj Comput. Mater.*, 2:16028, 2016. doi: 10.1038/npjcompumats.2016.28.

[100] Marwin H.S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, 555(7698):604–610, 2018. doi: 10.1038/nature25978.

[101] Pavlo O. Dral. MLatom: A program package for quantum chemical research assisted by machine learning. *J. Comput. Chem.*, 40(26):2339–2347, 2019. doi: 10.1002/jcc.26004.

[102] Yashaswi Pathak, Siddhartha Laghuvarapu, Sarvesh Mehta, and U. Deva Priyakumar. Chemically Interpretable Graph Interaction Network for Prediction of Pharmacokinetic

Properties of Drug-Like Molecules. *Proc. AAAI Conf. Artif. Intell.*, 34(01):873–880, apr 2020. doi: 10.1609/aaai.v34i01.5433.

[103] John A. Pople. Quantum Chemical Models (Nobel Lecture). *Angew. Chemie Int. Ed.*, 38(13-14):1894–1902, jul 1999. doi: 10.1002/(SICI)1521-3773(19990712)38:13/14<1894::AID-ANIE1894>3.0.CO;2-H.

[104] Juvenal Yosa Reyes, Sebastian Brickel, Oliver T. Unke, Tibor Nagy, and Markus Meuwly. HSO3Cl: A prototype molecule for studying OH-stretching overtone induced photodissociation. *Phys. Chem. Chem. Phys.*, 18(9):6780–6788, 2016. doi: 10.1039/c5cp07319g.

[105] Sebastian Brickel and Markus Meuwly. OH-Stretching Overtone Induced Dynamics in HSO3F from Reactive Molecular Dynamics Simulations. *J. Phys. Chem. A*, 121(27): 5079–5087, 2017. doi: 10.1021/acs.jpca.7b02950.

[106] Mojtaba Haghighatlari and Johannes Hachmann. Advances of machine learning in molecular modeling and simulation. *Curr. Opin. Chem. Eng.*, 23:51–57, 2019. doi: 10.1016/j.coche.2019.02.009.

[107] Christopher M Bishop. Pattern Recognition and Machine Learning. *J. Electron. Imaging*, 16(4):049901, jan 2007. doi: 10.1117/1.2819119.

[108] Ruhi Sarikaya, Geoffrey E. Hinton, and Anoop Deoras. Application of deep belief networks for natural language understanding. *IEEE Trans. Audio, Speech Lang. Process.*, 22(4): 778–784, apr 2014. doi: 10.1109/TASLP.2014.2303296.

[109] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.

[110] Felix A. Faber, Anders S. Christensen, Bing Huang, and O. Anatole Von Lilienfeld. Alchemical and structural distribution based representation for universal quantum machine learning. *J. Chem. Phys.*, 148(24):241717, 2018. doi: 10.1063/1.5020710.

[111] Kristof T. Schütt, Farhad Arbabzadah, Stefan Chmiela, Klaus R. Müller, and Alexandre Tkatchenko. Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, 8:6–13, 2017.

[112] K. T. Schütt, H. E. Sauceda, P. J. Kindermans, A. Tkatchenko, and K. R. Müller. SchNet - A deep learning architecture for molecules and materials. *Journal of Chemical Physics*, 148(24):241722, jun 2018.

[113] Siddhartha Laghuvarapu, Yashaswi Pathak, and U. Deva Priyakumar. BAND NN: A Deep Learning Framework for Energy Prediction and Geometry Optimization of Organic Small Molecules. *J. Comput. Chem.*, 41(8):790–799, mar 2020. doi: 10.1002/jcc.26128.

[114] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and E. Weinan. Deep Potential Molecular Dynamics: A Scalable Model with the Accuracy of Quantum Mechanics. *Phys. Rev. Lett.*, 120(14):143001, 2018. ISSN 10797114. doi: 10.1103/PhysRevLett.120.143001. URL `https://doi.org/10.1103/PhysRevLett.120.143001`.

[115] Nicholas Lubbers, Justin S. Smith, and Kipton Barros. Hierarchical modeling of molecular energies using a deep neural network. *J. Chem. Phys.*, 148(24):241715, 2018. doi: 10.1063/1.5011181.

[116] Oliver T. Unke and Markus Meuwly. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, 2019.

[117] Geoffrey R. Hutchison, Dakota L. Folmsbee, and David R. Koes. Evaluation of thermochemical machine learning for potential energy curves and geometry optimization. *J. Phys. Chem. A*, 125(9):1987–1993, 2021. doi: 10.1021/acs.jpca.0c10147.

[118] Michael Gastegger, Clemens Kauffmann, Jörg Behler, and Philipp Marquetand. Comparing the accuracy of high-dimensional neural network potentials and the systematic molecular fragmentation method: A benchmark study for all-trans alkanes. *J. Chem. Phys.*, 144(19), 2016. doi: 10.1063/1.4950815.

[119] Roy Dennington, Todd Keith, John Millam, and Others. GaussView, version 5, 2009.

[120] Greg Landrum. RDKit : A software suite for cheminformatics , computational chemistry , and predictive modeling. *Components*, 8, 2011.

[121] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pages 1–15, 2015.

[122] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, pages 1–23, 2018.

[123] K. T. Schütt, P. J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K. R. Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions. *Adv. Neural Inf. Process. Syst.*, 2017-Decem(1):992–1002, 2017.

[124] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. *Adv. Neural Inf. Process. Syst.*, 29:667–675, 2016.

[125] M J Frisch, G W Trucks, H B Schlegel, G E Scuseria, M A Robb, J R Cheeseman, G Scalmani, V Barone, B Mennucci, G A Petersson, H Nakatsuji, M Caricato, X Li, H P Hratchian, A F Izmaylov, J Bloino, G Zheng, J L Sonnenberg, M Hada, M Ehara,

K Toyota, R Fukuda, J Hasegawa, M Ishida, T Nakajima, Y Honda, O Kitao, H Nakai, T Vreven, J A Montgomery Jr., J E Peralta, F Ogliaro, M Bearpark, J J Heyd, E Brothers, K N Kudin, V N Staroverov, R Kobayashi, J Normand, K Raghavachari, A Rendell, J C Burant, S S Iyengar, J Tomasi, M Cossi, N Rega, J M Millam, M Klene, J E Knox, J B Cross, V Bakken, C Adamo, J Jaramillo, R Gomperts, R E Stratmann, O Yazyev, A J Austin, R Cammi, C Pomelli, J W Ochterski, R L Martin, K Morokuma, V G Zakrzewski, G A Voth, P Salvador, J J Dannenberg, S Dapprich, A D Daniels, Ö Farkas, J B Foresman, J V Ortiz, J Cioslowski, and D J Fox. Gaussian 09 Revision E.01, 2009.

[126] O. Rapin, J., Teytaud. Nevergrad - A gradient-free optimization platform. *Github Repos.*, 2018.

[127] Puru Jena and A. W. Castleman. Clusters: A bridge across the disciplines of physics and chemistry. *Proceedings of the National Academy of Sciences*, 103(28):10560–10569, 2006.

[128] AW Castleman Jr and SN Khanna. Clusters, superatoms, and building blocks of new materials. *The Journal of Physical Chemistry C*, 113(7):2664–2675, 2009.

[129] Keith D Ball, R Stephen Berry, Ralph E Kunz, Feng-Yin Li, Ana Proykova, and David J Wales. From topographies to dynamics on multidimensional potential energy surfaces of atomic clusters. *Science*, 271(5251):963–966, 1996.

[130] Martin Schmidt, Robert Kusche, Bernd von Issendorff, and Hellmut Haberland. Irregular variations in the melting point of size-selected atomic clusters. *Nature*, 393(6682):238–240, 1998.

[131] Alexandre A Shvartsburg and Martin F Jarrold. Solid clusters above the bulk melting point. *Physical Review Letters*, 85(12):2530, 2000.

[132] R Stephen Berry. Potential surfaces and dynamics: What clusters tell us. *Chemical Reviews*, 93(7):2379–2394, 1993.

[133] Kavita Joshi, Sailaja Krishnamurty, and DG Kanhere. "magic melters" have geometrical origin. *Physical Review Letters*, 96(13):135703, 2006.

[134] Chris J Pickard and RJ Needs. Ab initio random structure searching. *Journal of Physics: Condensed Matter*, 23(5):053201, 2011.

[135] David M Deaven and Kai-Ming Ho. Molecular geometry optimization with a genetic algorithm. *Physical Review Letters*, 75(2):288, 1995.

[136] Bernd Hartke. Global geometry optimization of clusters using genetic algorithms. *The Journal of Physical Chemistry*, 97(39):9973–9976, 1993.

[137] Stefan Goedecker. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *The Journal of Chemical Physics*, 120(21):9911–9917, 2004.

[138] Zhenqin Li and Harold A Scheraga. Monte carlo-minimization approach to the multiple-minima problem in protein folding. *Proceedings of the National Academy of Sciences*, 84 (19):6611–6615, 1987.

[139] Zhenqin Li and Harold A Scheraga. Structure and free energy of complex thermodynamic systems. *Journal of Molecular Structure: THEOCHEM*, 179(1):333–352, 1988.

[140] Scott Kirklin, James E Saal, Bryce Meredig, Alex Thompson, Jeff W Doak, Muratahan Aykol, Stephan Rühl, and Chris Wolverton. The open quantum materials database (oqmd): assessing the accuracy of dft formation energies. *npj Computational Materials*, 1(1):1–15, 2015.

[141] Bryce Meredig, Ankit Agrawal, Scott Kirklin, James E Saal, JW Doak, Alan Thompson, Kunpeng Zhang, Alok Choudhary, and Christopher Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Physical Review B*, 89(9):094104, 2014.

[142] Jianshu Jie, Zongxiang Hu, Guoyu Qian, Mouyi Weng, Shunning Li, Shucheng Li, Mingyu Hu, Dong Chen, Weiji Xiao, Jiaxin Zheng, et al. Discovering unusual structures from exception using big data and machine learning techniques. *Science Bulletin*, 64(9):612–616, 2019.

[143] Tim Mueller, Aaron Gilad Kusne, and Rampi Ramprasad. Machine learning in materials science: Recent progress and emerging applications. *Reviews in Computational Chemistry*, 29:186–273, 2016.

[144] Ryosuke Jinnouchi, Hirohito Hirata, and Ryoji Asahi. Extrapolating energetics on clusters and single-crystal surfaces to nanoparticles by machine-learning scheme. *The Journal of Physical Chemistry C*, 121(47):26397–26405, 2017.

[145] Yashaswi Pathak, Karandeep Singh Juneja, Girish Varma, Masahiro Ehara, and U. Deva Priyakumar. Deep learning enabled inorganic material generator. *Physical Chemistry Chemical Physics*, 22(46):26935–26943, 2020.

[146] April M. Cooper, Johannes Kästner, Alexander Urban, and Nongnuch Artrith. Efficient training of ANN potentials by including atomic forces via Taylor expansion and application to water and a transition-metal oxide. *npj Computational Materials*, 6(1):1–14, 2020.

[147] Sheena Agarwal, Shweta Mehta, and Kavita Joshi. Understanding the ml black box with simple descriptors to predict cluster–adsorbate interaction energy. *New Journal of Chemistry*, 44(20):8545–8553, 2020.

[148] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.

[149] Søren A Meldgaard, Esben L Kolsbjerg, and Bjørk Hammer. Machine learning enhanced global optimization by clustering local environments to enable bundled atomic energies. *The Journal of Chemical Physics*, 149(13):134104, 2018.

[150] Gabriel R Schleder, Antonio CM Padilha, Carlos Mera Acosta, Marcio Costa, and Adalberto Fazzio. From dft to machine learning: recent approaches to materials science–a review. *Journal of Physics: Materials*, 2(3):032001, 2019.

[151] Rampi Ramprasad, Rohit Batra, Ghanshyam Pilania, Arun Mannodi-Kanakkithodi, and Chiho Kim. Machine learning in materials informatics: recent applications and prospects. *npj Computational Materials*, 3(1):1–13, 2017.

[152] Ghanshyam Pilania, Chenchen Wang, Xun Jiang, Sanguthevar Rajasekaran, and Ramamurthy Ramprasad. Accelerating materials property predictions using machine learning. *Scientfic Reports*, 3:2810, 2013.

[153] Samadhan Kapse, Shazia Janwari, Umesh V Waghmare, and Ranjit Thapa. Energy parameter and electronic descriptor for carbon based catalyst predicted using qm/ml. *Applied Catalysis B: Environmental*, 286:119866, 2021.

[154] Shivam Saxena, Tuhin Suvra Khan, Fatima Jalid, Manojkumar Ramteke, and M Ali Haider. In silico high throughput screening of bimetallic and single atom alloys using machine learning and ab initio microkinetic modelling. *Journal of Materials Chemistry A*, 8(1):107–123, 2020.

[155] Atsuto Seko, Hiroyuki Hayashi, Keita Nakayama, Akira Takahashi, and Isao Tanaka. Representation of compounds for machine-learning prediction of physical properties. *Physical Review B*, 95(14):144110, 2017.

[156] Xin Chen, Dong Chen, Mouyi Weng, Yi Jiang, Guo-Wei Wei, and Feng Pan. Topology-based machine learning strategy for cluster structure prediction. *The Journal of Physical Chemistry Letters*, 11(11):4392–4401, 2020.

[157] Praveen Pankajakshan, Suchismita Sanyal, Onno E de Noord, Indranil Bhattacharya, Arnab Bhattacharyya, and Umesh Waghmare. Machine learning and statistical analysis for materials science: stability and transferability of fingerprint descriptors and chemical insights. *Chemistry of Materials*, 29(10):4190–4201, 2017.

[158] Ankit Jain and Thomas Bligaard. Atomic-position independent descriptor for machine learning of material properties. *Physical Review B*, 98(21):214112, 2018.

[159] Siddhartha Laghuvarapu, Yashaswi Pathak, and U Deva Priyakumar. Band nn: A deep learning framework for energy prediction and geometry optimization of organic small molecules. *Journal of Computational Chemistry*, 41(8):790–799, 2020.

[160] Felix Faber, Alexander Lindmaa, O Anatole von Lilienfeld, and Rickard Armiento. Crystal structure representations for machine learning models of formation energies. *International Journal of Quantum Chemistry*, 115(16):1094–1101, 2015.

[161] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *The Journal of Chemical Physics*, 134(7):074106, 2011.

[162] Helmut Gassner, Michael Probst, Albert Lauenstein, and Kersti Hermansson. Representation of intermolecular potential functions by neural networks. *The Journal of Physical Chemistry A*, 102(24):4596–4605, 1998.

[163] Sönke Lorenz, Matthias Scheffler, and Axel Gross. Descriptions of surface chemical reactions using a neural network representation of the potential-energy surface. *Physical Review B*, 73(11):115431, 2006.

[164] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108(5):058301, 2012.

[165] Albert P Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, 2013.

[166] Shweta Jindal, Siva Chiriki, and Satya S. Bulusu. Spherical harmonics based descriptor for neural network potentials: Structure and dynamics of au147 nanocluster. *The Journal of Chemical Physics*, 146(20):204301, 2017.

[167] Kristof T Schütt, Henning Glawe, Felix Brockherde, Antonio Sanna, Klaus-Robert Müller, and Eberhard KU Gross. How to represent crystal structures for machine learning: Towards fast prediction of electronic properties. *Physical Review B*, 89(20):205118, 2014.

[168] Nongnuch Artrith, Alexander Urban, and Gerbrand Ceder. Efficient and accurate machine-learning interpolation of atomic energies in compositions with many species. *Physical Review B*, 96(1):1–5, 2017.

[169] Connor W Coley, Regina Barzilay, William H Green, Tommi S Jaakkola, and Klavs F Jensen. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of Chemical Information and Modeling*, 57(8):1757–1772, 2017.

[170] Logan Ward, Ruoqian Liu, Amar Krishna, Vinay I Hegde, Ankit Agrawal, Alok Choudhary, and Chris Wolverton. Including crystal structure attributes in machine learning models of formation energies via voronoi tessellations. *Physical Review B*, 96(2):024104, 2017.

[171] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-aided Molecular Design*, 30(8):595–608, 2016.

[172] April M. Miksch, Tobias Morawietz, Johannes Kästner, Alexander Urban, and Nongnuch Artrith. Strategies for the construction of machine-learning potentials for accurate and efficient atomic-scale simulations. *Mach. Learn. Sci. Technol.*, 2(3):1–21, 2021. ISSN 26322153.

[173] P. E. Blöchl. Projector augmented-wave method. *Physical Review B*, 50:17953–17979, Dec 1994.

[174] G. Kresse and D. Joubert. From ultrasoft pseudopotentials to the projector augmented-wave method. *Physical Review B*, 59:1758–1775, Jan 1999.

[175] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical Review Letters*, 77:3865–3868, Oct 1996.

[176] John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Physical Review Letters*, 78:1396–1396, Feb 1997.

[177] G. Kresse and J. Hafner. *Ab initio* molecular-dynamics simulation of the liquid-metal–amorphous-semiconductor transition in germanium. *Physical Review B*, 49:14251–14269, May 1994.

[178] G. Kresse and J. Furthmüller. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. *Physical Review B*, 54:11169–11186, Oct 1996.

[179] G. Kresse and J. Furthmüller. Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Material Science*, 6(1): 15 – 50, 1996. ISSN 0927-0256.

[180] Vaibhav Kaware and Kavita Joshi. Scaling up the shape: A novel growth pattern of gallium clusters. *The Journal of Chemical Physics*, 141(5):054308, 2014.

[181] Anju Susan, Aniruddha Kibey, Vaibhav Kaware, and Kavita Joshi. Correlation between the variation in observed melting temperatures and structural motifs of the global minima of gallium clusters: An ab initio study. *The Journal of Chemical Physics*, 138(1):014303, 2013.

[182] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[183] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Proceedings of the IEEE International Conference on Computer Vision*, 2015 Inter:1026–1034, feb 2015.

[184] Raghu Nath Dhital, Keigo Nomura, Yoshinori Sato, Setsiri Haesuwannakij, Masahiro Ehara, and Hidehiro Sakurai. Pt-Pd nanoalloy for the unprecedented activation of carbon-fluorine bond at low temperature. *Bulletin of the Chemical Society of Japan*, 93(10): 1180–1185, 2020.

[185] Jian Lv, Yanchao Wang, Li Zhu, and Yanming Ma. Particle-swarm structure prediction on clusters. *Journal of Chemical Physics*, 137(8), 2012. ISSN 00219606.

[186] Martin Saunders. Stochastic Search for Isomers on a Quantum Mechanical Surface. *Journal of Computational Chemistry*, 25(5):621–626, 2004.

[187] Puru Jena and AW Castleman. Clusters: A bridge across the disciplines of physics and chemistry. *Proceedings of the National Academy of Sciences*, 103(28):10560–10569, 2006.

[188] Walt A De Heer. The physics of simple metal clusters: experimental aspects and simple models. *Reviews of Modern Physics*, 65(3):611, 1993.

[189] S Chacko, Kavita Joshi, DG Kanhere, and SA Blundell. Why do gallium clusters have a higher melting point than the bulk? *Physical review letters*, 92(13):135506, 2004.

[190] Francesca Baletto and Riccardo Ferrando. Structural properties of nanoclusters: Energetic, thermodynamic, and kinetic effects. *Reviews of modern physics*, 77(1):371, 2005.

[191] Joachim Bansmann, SH Baker, C Binns, JA Blackman, J-P Bucher, J Dorantes-Dávila, V Dupuis, Luc Favre, D Kechrakos, A Kleibert, et al. Magnetic and structural properties of isolated and assembled clusters. *Surface Science Reports*, 56(6-7):189–275, 2005.

[192] Luc T Wille and Jan Vennik. Computational complexity of the ground-state determination of atomic clusters. *Journal of Physics A: Mathematical and General*, 18(8):L419, 1985.

[193] Chris J Pickard and RJ Needs. Ab initio random structure searching. *Journal of Physics: Condensed Matter*, 23(5):053201, 2011.

[194] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[195] Bernd Hartke. Global geometry optimization of clusters using genetic algorithms. *The Journal of Physical Chemistry*, 97(39):9973–9976, 1993.

[196] Stefan Goedecker. Minima hopping: An efficient search method for the global minimum of the potential energy surface of complex molecular systems. *The Journal of chemical physics*, 120(21):9911–9917, 2004.

[197] Zhenqin Li and Harold A Scheraga. Monte carlo-minimization approach to the multiple-minima problem in protein folding. *Proceedings of the National Academy of Sciences*, 84 (19):6611–6615, 1987.

[198] Yanchao Wang, Jian Lv, Li Zhu, and Yanming Ma. Crystal structure prediction via particle-swarm optimization. *Physical Review B*, 82(9):094116, 2010.

[199] Yonghong Tian, Weiguo Sun, Bole Chen, Yuanyuan Jin, and Cheng Lu. Cluster structure prediction via calypso method. *Chinese Physics B*, 28(10):103104, 2019.

[200] Jun Zhang and Vassiliki-Alexandra Glezakou. Global optimization of chemical cluster structures: Methods, applications, and challenges. *International Journal of Quantum Chemistry*, 121(7):e26553, 2021.

[201] Puru Jena and Qiang Sun. Super atomic clusters: design rules and potential for building blocks of materials. *Chemical reviews*, 118(11):5755–5870, 2018.

[202] Jonathan PK Doye, David J Wales, and R Stephen Berry. The effect of the range of the potential on the structures of clusters. *The Journal of chemical physics*, 103(10): 4234–4249, 1995.

[203] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Cent. Sci.*, 4(2): 268–276, feb 2018.

[204] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction Tree Variational Autoencoder for Molecular Graph Generation. *RSC Drug Discov. Ser.*, 2021-Janua(75):228–249, feb 2018.

[205] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. *6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc.*, pages 1–17, 2018.

[206] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models. *arXiv preprint arXiv:1705.10843*, may 2017.

[207] Nicola De Cao and Thomas Kipf. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, may 2018.

[208] Marwin H. S. Segler, Thierry Kogej, Christian Tyrchan, and Mark P. Waller. Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks. *ACS Cent. Sci.*, 4(1):120–131, jan 2018.

[209] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Sci. Adv.*, 4(7):1–28, 2018.

[210] Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. Mol-CycleGAN: A generative model for molecular optimization. *J. Cheminform.*, 12(1):1–18, 2020.

[211] Niklas W.A. Gebauer, Michael Gastegger, and Kristof T. Schött. Generating equilibrium molecules with deep neural networks. *arXiv*, oct 2018.

[212] Niklas W.A. Gebauer, Michael Gastegger, and Kristof T. Schütt. Symmetry-adapted generation of 3d point sets for the targeted discovery of molecules. *arXiv*, 2019.

[213] Juhwan Noh, Jaehoon Kim, Helge S. Stein, Benjamin Sanchez-Lengeling, John M. Gregoire, Alan Aspuru-Guzik, and Yousung Jung. Inverse Design of Solid-State Materials via a Continuous Representation. *Matter*, 1(5):1370–1384, 2019.

[214] Sungwon Kim, Juhwan Noh, Geun Ho Gu, Alan Aspuru-Guzik, and Yousung Jung. Generative Adversarial Networks for Crystal Structure Prediction. *ACS Cent. Sci.*, 6(8): 1412–1420, 2020.

[215] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, 2017-January:77–85, 2017.

[216] Asma Nouira, Nataliya Sokolovska, and Jean Claude Crivello. CrystalGAN: Learning to discover crystallographic structures with generative adversarial networks. *CEUR Workshop Proc.*, 2350(Umr 7182), 2019.

[217] Callum J. Court, Batuhan Yildirim, Apoorv Jain, and Jacqueline M. Cole. 3-D inorganic crystal structure generation and property prediction via representation learning. *J. Chem. Inf. Model.*, 60(10):4518–4535, 2020.

[218] Rohit Modee, Sheena Agarwal, Ashwini Verma, Kavita Joshi, and U. Deva Priyakumar. DART: Deep learning enabled topological interaction model for energy prediction of metal clusters and its application in identifying unique low energy isomers. *Phys. Chem. Chem. Phys.*, 23(38):21995–22003, 2021. ISSN 14639076. doi: 10.1039/d1cp02956h.

[219] Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. *Adv. Neural Inf. Process. Syst.*, 32(NeurIPS), 2019.

[220] Rohit Modee, Siddhartha Laghuvarapu, and U. Deva Priyakumar. Benchmark study on deep neural network potentials for small organic molecules. *J. Comput. Chem.*, 43(5): 308–318, 2022. ISSN 1096987X.

[221] Frank Noé, Alexandre Tkatchenko, Klaus Robert Müller, and Cecilia Clementi. Machine learning for molecular simulation. *Annu. Rev. Phys. Chem.*, 71:361–390, 2020.

[222] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N. Zare, and Patrick Riley. Optimization of Molecules via Deep Reinforcement Learning. *Sci. Rep.*, 9(1):10752, dec 2019.

[223] Tarun Gogineni, Ziping Xu, Exequiel Punzalan, Runxuan Jiang, Joshua Kammeraad, Ambuj Tewari, and Paul Zimmerman. TorsionNet: A reinforcement learning approach to sequential conformer search. *Adv. Neural Inf. Process. Syst.*, 2020-December(MD):1–12, 2020. ISSN 10495258.

[224] Kento Shin, Duy Phuoc Tran, Kazuhiro Takemura, Akio Kitao, Kei Terayama, and Koji Tsuda. Enhancing Biomolecular Sampling with Reinforcement Learning: A Tree Search Molecular Dynamics Simulation Method. *ACS Omega*, 4(9):13853–13862, 2019. ISSN 24701343. doi: 10.1021/acsomega.9b01480.

[225] Zahra Shamsi, Kevin J. Cheng, and Diwakar Shukla. Reinforcement Learning Based Adaptive Sampling: REAPing Rewards by Exploring Protein Conformational Landscapes. *J. Phys. Chem. B*, 122(35):8386–8395, 2018. ISSN 15205207. doi: 10.1021/acs.jpcb.8b06521.

[226] Lucian Chan, Geoffrey R. Hutchison, and Garrett M. Morris. Bayesian optimization for conformer generation. *J. Cheminform.*, 11(1):1–11, 2019. ISSN 17582946. doi: 10.1186/s13321-019-0354-7. URL `https://doi.org/10.1186/s13321-019-0354-7`.

[227] Lincan Fang, Xiaomi Guo, Milica Todorović, Patrick Rinke, and Xi Chen. Exploring the Conformers of an Organic Molecule on a Metal Cluster with Bayesian Optimization. *J. Chem. Inf. Model.*, 63(3):745–752, 2023. ISSN 1549960X. doi: 10.1021/acs.jcim.2c01120.

[228] Gregor N.C. Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. *arXiv*, 2020.

[229] Jorge Nocedal and Stephen J. Wright. Numerical optimization. In *Springer Ser. Oper. Res. Financ. Eng.*, pages 1–664. Chapman and Hall/CRC, sep 2006.

[230] H. Bernhard Schlegel. Exploring potential energy surfaces for chemical reactions: An overview of some practical methods. *J. Comput. Chem.*, 24(12):1514–1527, sep 2003. ISSN 01928651.

[231] H. Bernhard Schlegel. Geometry optimization. *Wiley Interdiscip. Rev. Comput. Mol. Sci.*, 1(5):790–809, sep 2011. ISSN 17590876.

[232] Jacques Hadamard. *Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées*, volume 33. Imprimerie nationale, 1908.

[233] A Cauchy. Methode gènèrale pour la rèsolution des systemes d'èqations simultan´ees. *Comptes Rendus*, 25(1847):536, 1847.

[234] Ke Li and Jitendra Malik. Learning to Optimize. *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, jun 2016.

[235] Lucas N. Egidio, Anders Hansson, and Bo Wahlberg. Learning the Step-size Policy for the Limited-Memory Broyden-Fletcher-Goldfarb-Shanno Algorithm. *Proc. Int. Jt. Conf. Neural Networks*, 2021-July(2), 2021.

[236] Luke Metz, Niru Maheswaranathan, Jeremy Nixon, C. Daniel Freeman, and Jascha Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. *36th Int. Conf. Mach. Learn. ICML 2019*, 2019-June:8016–8035, 2019.

[237] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. *Adv. Neural Inf. Process. Syst.*, 29(Nips):3988–3996, jun 2016. ISSN 10495258.

[238] Kabir Ahuja, William H. Green, and Yi Pei Li. Learning to Optimize Molecular Geometries Using Reinforcement Learning. *J. Chem. Theory Comput.*, 17(2):818–825, 2021. ISSN 15499626.

[239] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. *Proc. Theor. Asp. Reason. about Knowledge, TARK-96*, 1996.

[240] L. S. Shapley. Stochastic Games. *Proc. Natl. Acad. Sci.*, 39(10):1095–1100, 1953. ISSN 0027-8424.

[241] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.*, 2017-Decem:6380–6391, 2017. ISSN 10495258.

[242] Samuel P.M. Choi, Dit Yan Yeung, and Nevin L. Zhang. An environment model for nonstationary reinforcement learning. *Adv. Neural Inf. Process. Syst.*, pages 307–313, 2000. ISSN 10495258.

[243] Georgios Papoudakis, Filippos Christianos, Arrasy Rahman, and Stefano V. Albrecht. Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning. *arXiv Prepr. arXiv1906.04737*, jun 2019. doi: 10.48550.

[244] Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative Multi-agent Control Using Deep Reinforcement Learning. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 10642 LNAI:66–83, may 2017. ISSN 16113349.

[245] Xiangxiang Chu and Hangjun Ye. Parameter Sharing Deep Deterministic Policy Gradient for Cooperative Multi-agent Reinforcement Learning. *arXiv Prepr. arXiv1710.00336*, oct 2017.

[246] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2130:87–94, 2001. ISSN 16113349.

[247] Luke Metz, Niru Maheswaranathan, Ruoxi Sun, C. Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Using a thousand optimization tasks to learn hyperparameter search strategies. feb 2020.

[248] Luke Metz, C. Daniel Freeman, James Harrison, Niru Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. 2022.

[249] Luke Metz, Niru Maheswaranathan, C. Daniel Freeman, Ben Poole, and Jascha Sohl-Dickstein. Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves. 2020.

[250] Sabrina Jaeger, Simone Fulle, and Samo Turk. Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition. *J. Chem. Inf. Model.*, 58(1):27–35, 2018. ISSN 1549960X. doi: 10.1021/acs.jcim.7b00616.