Deep learning for quantum chemistry using density functional tight-binding method

Masters Thesis

Submitted in fulfillment of the requirements of
BITS F421T Thesis

By

Raghav Arora
ID No. 2017B2A31016P

Under the supervision of:

Dr. Alexandre Tkatchenko
&
Dr. Ram Kinkar Roy

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS
November 2023

Declaration of Authorship

I, Raghav Arora, declare that this Masters Thesis titled, 'Deep learning for quantum chemistry using density functional tight-binding method' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: RAGHAV ARORA

Date:26/12/2021

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI, PILANI CAMPUS

Abstract

Masters of Science (Hons.) Chemistry

Deep learning for quantum chemistry using density functional tight-binding method

by Raghav Arora

Machine Learning (ML) is an efficient tool capable of providing accurate prediction of quantum mechanical properties without the computational overheads of high-level electronic structure methods. This has led to numerous investigations, which have accelerated the exploration of the chemical compound space for organic molecules and materials of interest. However, for developing a ML model to predict physicochemical properties, we need a way to mathematically represent the molecules and integrate it to a ML methodology. For this, several geometric descriptors have been developed in the literature. Although these descriptors are able to capture the geometric structure of a molecule, they have their own drawbacks, especially in the context of predicting intensive properties and when dealing with non-equilibrium molecular structures.

In this regard, the present work aims to develop novel geometric and/or electronic descriptors to accurately predict the physicochemical properties of highly distorted and equilibrium molecular structures from the QM7-X dataset. These electronic descriptors will be generated using semi-empirical quantum-chemical method such as Density Functional Tight Binding (DFTB). Diverse neural network architectures will be also considered to test the new molecular representations for the prediction of PBE0 atomization energy and HOMO-LUMO gap of these molecules. Our results show that for intensive properties like HOMO-LUMO gap, the electronic descriptors are able to provide better prediction results than geometric descriptors. Whereas, for extensive properties like atomization energy, the combination of geometric with electronic descriptors can efficiently represent the molecule and, then, generate more efficient ML models. We expect that our findings open up the route to a low-cost search for accurate prediction of physicochemical properties of equilibrium as well as non-equilibrium molecular structures, which is relevant for a better understanding of chemical processes such as chemical reactivity and molecular design.

Acknowledgements

A special thank you for the invaluable guidance and support provided by my Master Thesis supervisors Prof Alexandre Tkatchenko from the University of Luxembourg and Prof. Ram Kinkar Roy from BITS Pilani. I would also like to express my thanks to Dr. Leonardo Medrano Sandonas from the Theoretical Chemical Physics Research Group for his guidance and inspiration throughout the project.

I acknowledge the Center for Information Services and High Performance Computing (ZIH) from Techincal Universität Dresden, for the computational resources provided for this thesis.

A thank you to Yolande Edjogo for the logistical support, and to the whole TCP group for their welcoming attitude. I would also like to thank my family, whom without this would have not been possible. I also appreciate all the support I received from my friends. Lastly, I would like to thank the BITS that provided an opportunity to conduct this thesis.

Contents

List of Figures

List of Tables

Abbreviations

MAE	**M**ean **A**bsolute **E**rror
MSE	**M**ean **S**quare **E**rror
CM	**C**oulomb **M**atrix
BOB	**B**ag **O**f **B**onds
SLATM	**S**pectrum of **L**ondon and **A**xillrod **T**eller **M**uto potential
CCS	**C**hemical **C**ompound **S**pace
HOMO	**H**igher **O**ccupied **M**olecular **O**rbital
LUMO	**L**owest **U**occupied **M**olecular **O**rbital
ML	**M**achine **L**earning
NN	**N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
KRR	**K**ernel **R**idge **R**egression
QML	**Q**uantum **M**achine **L**earning

Chapter 1

Introduction

1.1 Background

An important aspect of chemical and material science research relies on the discovery and characterisation of novel compounds that have the optimal properties for a given purpose. The advances in this field have been accelerated significantly by the development of diverse quantum-mechanical (QM) methods. The combination of these methods with machine learning (ML) techniques has been widely used to gain insights into the understanding of complex structure-property and property-property relationships of organic and inorganic materials [1] [2]. ML used in the context of quantum mechanics is always referred as "Quantum Machine Learning", or QML [3]. Thus, applications QML in the field of chemical and material sciences include: (i) computational design and discovery of molecules [3–5], (ii) development of efficient force fields for molecular dynamics simulations [6–8], (iii) generation of extensive datasets [9–11], (iv) molecular property prediction [2, 12, 13].

Looking at some Chemistry Journals, categorized based on technical divisions of the American Chemical Society [1] that contain ML keywords, it was found that there were total 5,833 such publications published in the year 2000, while a total of 26,836 published in 2010, and 1,16,750 publications in 2021 [1]. It is evident that the prevalence of ML is increasing over time. This is due to the availability of extensive QM datasets, as well as the recent developments in the computer hardware, such as parallel processing architectures in GPUs (Graphical Processing Units). These hardware developments allow us to handle large datasets and train deep learning models which are very computational expensive.

ML-based approaches mainly use well-established statistical algorithms, however, the performance of the ML models developed with these methods strongly depend on the quality of the training

[1] The script for this calculation can be found at `https://github.com/keithgroup/scopus_searching_ML_in_chem_literature`

data. Hence, one of the most important factors for the success of ML models for the prediction of QM properties of molecules is the availability of QM datasets. Due to the expensive nature of highly-accurate QM calculations, there are only a few number of QM datasets which match the requirements for accurate molecular property predictions, and some of them are quite recent. To mention a group of notable datasets, we have: QM7 [14, 15], QM8 [16, 17], QM9 [17, 18], ANI-1 [10] and QM7-X [11]. All datasets consist of small organic molecules ranging in size from QM7 with ≈7k equilibrium molecular structures formed of up to 7 non-hydrogen atoms (C, N, O, S, Cl) to the QM9 with ≈134k equilibrium molecular structures formed of up to 9 non-hydrogen atoms (C, N, O, F). Besides dataset size, another key difference is the presence of non-equilibrium molecular structures, only included in ANI-1 (≈20 million conformations) and QM7-X (≈4.2 million conformations).

A challenge of ML for property prediction is the mathematical representation of a molecule, known as molecular descriptor [19] [20] The numerical values of these descriptors are used to quantitatively describe the physical and chemical information of each molecule. The main idea in the definition of a descriptor is to include the maximum information about the molecules while maintaining the physical invariances, e.g. rotational and translational invariances. The more information about the molecules is provided as input to the model, the better the model would perform. Diverse molecular descriptors are currently available to encode the geometric structures of molecules. For instance, SMILES (Simplified Molecular-Input Line-Entry System) [21] is a simple chemical notation representing a molecule in the form of a string of atom types and the bonds between them. To incorporate three-dimensional information about molecules, more complex and physically-or chemically-inspired geometric descriptors were also defined.

A descriptor based on the Coulomb interaction between atomic charges was thus defined and named as Coulomb Matrix (CM) [14]. This descriptor was initially tested for the prediction of the atomization energy of small molecules in QM7 dataset. Bag-of-Bonds (BoB) representation was posteriorly used and it showed a better performance than CM representation [22]. Similar to the bag-of-words representation used for natural language processing, BoB representation uses the interatomic distances in a molecule. A bag represents a particular bond type, and the final representation is a vector formed by concatenating all bag of bonds in a specific order. Unlike CM, BoB does not distinguish between homometric molecules (molecules with different geometries but equal set of pairwise distances between nuclei) [22]. This representation includes more details about the interactions than CM, and is hence more computationally expensive. A further complicated and computationally expensive representation is the Spectrum of London and Axilrod-Teller-Muto (ATM) potential, or also known as SLATM representation [23]. SLATM was developed and initially tested on the QM7b and QM9 datasets.

Although these representations have worked well for various purposes, including property prediction of molecules, they have their limitations. As all of these representations are based on

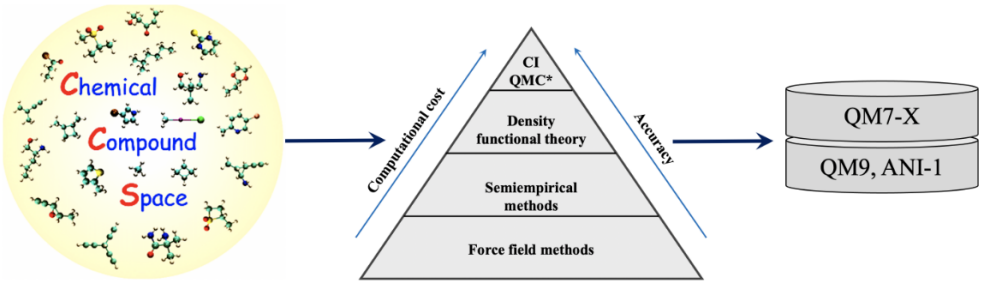


FIGURE 1.1: The route of the computational modelling of molecular systems for the generation of QM datasets: using the chemical compound space shown in the left panel, computations and simulations are performed at different levels of accuracy and computational cost (see middle panel). Finally, we store the results in datasets, containing information about molecules such as quantum-mechanical properties and molecular structure. Figure taken from [24]

the 3D structure of molecules, they work properly for the prediction of extensive QM properties. However, the performance for intensive QM properties, like HOMO-LUMO gap and dipole moment, is very low [25]. Moreover, some geometric representations such as SLATM present high dimensionality, which makes them very computational expensive. In brief, the definition of low-cost and effective QM descriptors to develop reliable ML models for the prediction of both extensive and intensive physicochemical properties is still an open challenge in ML investigations.

1.2 Objectives

The goal of the present master thesis is to address the challenge described above by defining new geometric and/or electronic descriptors for accurate prediction of diverse physicochemical properties of small drug-like molecules. To do so, we will combine geometric descriptors such as CM, BOB, and SLATM with electronic properties computed by using a well-established semi-empirical method such as density functional tight-binding (DFTB) method [26]. These descriptors will be integrated into neural networks (NN) architectures. Here, we will make use of information about equilibrium and non-equilibrium molecular structures from the QM7-X dataset [11]. Additionally, a feature reduction technique is used to identify the most relevant DFTB electronic properties for property prediction.

The present work is organized as follows: in Chapter 2, a brief introduction to QM methods along with the description of the generation procedure of QM7-X are given. An explanation about neural networks is also provided in Chapter 2. The results obtained in this project are presented and discussed in Chapter 3. Finally, the synopsis, final remarks, and outlook of the master thesis can be found in Chapter 4.

Chapter 2

Fundamental Concepts

2.1 Theoretical Background

Quantum mechanics is based on a series of postulates formulated at the turn of the 20th century. It is stated that any closed quantum-mechanical (QM) system is fully described by a wave function, ψ. Any physically-acceptable wave function thereby has to be continuous, single-valued and square-integrable (with the exception of continuum states). For a system of nuclei and electrons, a relation between the wave function and the system is given by Schrödinger equation (SE):

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

$\hat{H}$ is the Hamiltonian operator, and when applied to the wave function ψ, the eigenvalue of this operator gives us the total energy of the system. After separating the above differential equation into a time-dependent phase factor oscillating in the complex plane and a stationary part of the wave function, one arrives at the time-independent SE,

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V\psi = E\psi$$

According to the Born-Oppenheimer approximation, this equation can be further factorized into nuclear and electronic parts. So, for an N-electron system, the many-electron time-independent Schrödinger equation can not be separated into simpler single-particle equations because of the electron-electron interaction $\hat{U}$. Hence, for the three spatial coordinates, the equation is 3N dimensional.

2.1.1 Density Functional Theory

The idea behind Density Functional Theory [27, 28] is to express the total energy as a functional of $\rho(r)$, the electronic charge density. This reduces the dimensionality of the problem from 3N, to simply three coming from the three spatial coordinates. According to the second Hohenberg–Kohn theorem [27], the electron density that minimizes the energy of the overall functional is the true electron density corresponding to the full solution of the Schrödinger equation. Thus, the ground state wave function, ψ_0 is a unique functional of the ground state electron density ρ_0, i.e, $\psi_0 = \psi[\rho_0]$. Consequently, the ground state energy is also a functional of ρ_0:

$$E_0 = E[\rho_0] = \langle \psi[\rho_0] | \hat{T} + \hat{V} + \hat{U} | \psi[\rho_0] \rangle \ . \tag{2.1}$$

Here, $\hat{T}$ is the kinetic energy operator, $\hat{V}$ is the potential energy from the external field due to positively charged nuclei, and $\hat{U}$ is the electron-electron interaction energy. Unlike $\hat{T}$ and $\hat{U}$, $\hat{V}$ depends on the system under study and can be expressed as: $V[\rho] = \int V(\mathbf{r})\rho(\mathbf{r})d^3\mathbf{r}$. $\hat{T}$ and $\hat{U}$ are called universal operators, because they are the same for any N-electron system.

The ground state energy is then obtained by minimizing the following functional with respect to $\rho(\mathbf{r})$:

$$E[\rho] = T[\rho] + U[\rho] + \int V(\mathbf{r})\rho_0(\mathbf{r})d^3\mathbf{r} \tag{2.2}$$

Although the dimensionality is reduced, this minimization is still a complex problem for many-electron systems. To solve this, Kohn and Sham formulated a theory known as the Kohn-Sham DFT (KS-DFT). Here, exchange and electron correlation effects are collected into the exchange-correlation (xc) functional. The exact mathematical form of the E_{xc} is still unknown and most applications rely on approximated methods, which use different models of the density functional.

$$E[\rho(\mathbf{r})] = T_s + E_{ext} + E_H + E_{xc} + E_{II} \tag{2.3}$$

Here, T_s is the non-interacting kinetic energy, E_H is the hartree energy, and E_xc is the exchange-correlation energy. Expanding the terms, we get:

$$E[\rho(\mathbf{r})] = \sum_a f_a \langle \psi_a | -1/2\nabla^2 + V_{ext} + 1/2 \int \frac{\rho(\mathbf{r'})}{|r' - r|} d^3r' | \psi_a \rangle + E_{xc}[\rho] + E_{II} \tag{2.4}$$

Here, $f_a \in \{0, 2\}$, is the occupation of a single-particle state ψ_a with energy ϵ_a. The Hartree potential used in the equation is given by:

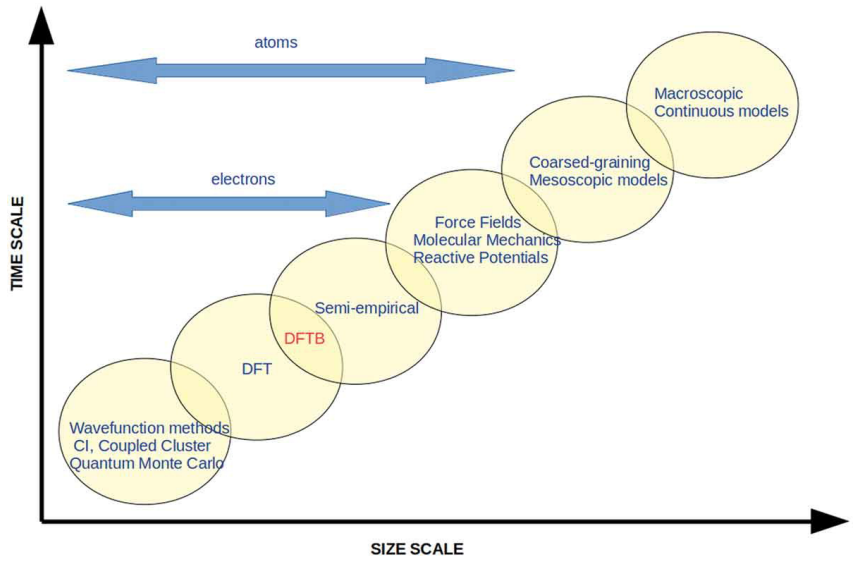


FIGURE 2.1: The position of density functional tight-binding (DFTB) method in time-size graph is shown. Figure taken from [29]

$$V_H(r) = \int \frac{\rho(\mathbf{r'})}{|\mathbf{r} - \mathbf{r'}|} d^3\mathbf{r'} \tag{2.5}$$

2.1.2 Density Functional Tight Binding

The Density Functional Tight-Binding (DFTB) [26] is a semi-empirical method based on the perturbation expansion of the Kohn-Sham energy functional, in a minimal basis expansion of atomic orbitals. In the equation for the KS-DFT, the ground state electron density is expressed as a combination of a reference electron density ($\rho_0(\mathbf{r})$) and some fluctuations ($\delta\rho(\mathbf{r})$) [30].

$$\rho(\mathbf{r}) = \rho_0(\mathbf{r}) + \delta\rho(\mathbf{r}) \tag{2.6}$$

This relation is substituted in the KS-DFT equation, and perturbation expansion is done in the form of Taylor Series about the reference electron density. The non-interacting reference electron density ρ_0 is taken as the superposition of the effective atomic densities $\rho_A(\mathbf{r})$: $\rho_0(r) = \sum_j \rho_A^j(r)$. On expanding the equation up to the 3rd order, we get the following expression for the total energy:

$$E^{DFTB3}[\rho_0 + \delta\rho] = E^0[\rho_0] + E^1[\rho_0, \delta\rho] + E^2[\rho_0, (\delta\rho)^2] + E^3[\rho_0, (\delta\rho)^3] \tag{2.7}$$

$$
\begin{aligned}
E_{tot} = \sum_a f_a \langle\psi_a|-1/2\nabla^2 + V_{ext} + V_H(\rho\prime) + \frac{\delta E_{xc}[\rho]}{\delta\rho}|\psi_a\rangle \\
+\frac{1}{2}\iint_{R^6}\left(\frac{1}{||r-r'||} + \frac{\delta^2 E_{xc}[\rho]}{\delta\rho}\Big|_{\rho_0}\right)\delta\rho\delta\rho\prime d(r)d\mathbf{r'} \\
+\frac{1}{6}\iiint_{R^9}\frac{\delta^3 E_{xc}[\rho]}{\delta\rho\delta\rho'\delta\rho''}\Big|_{\rho_0}\delta\rho\delta\rho\prime\delta\rho\prime\prime d(r)d\mathbf{r'}d\mathbf{r''} \\
+\sum_{A<B}\frac{Z_A Z_B}{\|R_A - R_B\|} - \frac{1}{2}\int_{R^3} V_H[\rho_0\prime]\rho_0 d(r) + E_{xc}[\rho_0] - \int_{R^3}\frac{\delta E_{xc}[\rho]}{\delta\rho}\Big|_{\rho_0}\rho_0 d\mathbf{r}
\end{aligned}
\tag{2.8}
$$

As previously mentioned, f_a denotes the occupation number of the single particle state ψ_a, ρ denotes $\rho(\mathbf{r})$, $\rho\prime$ denotes $\rho((r')$, and so on. Z_A and Z_B represent the atomic numbers of the atoms, and $\delta/\delta\rho$ denotes the functional derivative with respect to ρ. The first line is the band structure term, which equals the DFT energy of the starting density, ρ_0. The 2nd term is the 2nd order DFTB term, and denotes the coulomb energy. The third line the third-order contribution and the last line is collectively referred to as the repulsive energy [30]. For the Band-structure term, the electrons are considered tightly-bound to their respective atoms, and the effects arising from the relaxation of the charge on the atoms is provided by the 2nd and the 3rd order terms. For the 0th and 1st order approximations, the orbitals are written as linear combination of atomic orbitals (LCAO) of valence-only minimal basis set ϕ_μ:

$$\psi_i = \sum_\mu c_{\mu i}\phi_\mu \tag{2.9}$$

The 0_{th} order term in equation 2.7, $E^0[\rho_0]$ is approximated as a sum of pair potentials, which are either determined by comparison with DFT calculations or fitted to empirical data. $E^0[\rho_0] \approx E_{rep} = \frac{1}{2}\sum_{AB} V_{AB}^{rep}$ For the approximation of 2nd and 3rd order terms, E^2 and E^3, the density perturbations, $\delta\rho$ are written as superposition of atomic contributions, taken to be exponentially decaying spherically symmetric charge densities,

$$\delta\rho(\mathbf{r}) = \sum_A \delta\rho_A(\mathbf{r}-\mathbf{R}_A) \approx \frac{1}{\sqrt{4\pi}}\sum_A\left(\frac{\tau_A^3}{8\pi}e^{-\tau_A|\mathbf{r}-\mathbf{R}_A|}\right)\Delta q_A \tag{2.10}$$

Δq_A is the charge fluctuation from the Mulliken charge $\{q_A\}$. Neglecting the XC effects for the moment, E^2 leads to an analytical function γ_{AB},

$$E^2(\tau_A,\tau_B,R_{AB}) = \frac{1}{2}\sum_{AB(\neq A)}\gamma_{AB}(\tau_A,\tau_B,R_{AB})\Delta q_A\Delta q_B \tag{2.11}$$

The energy depends on the mulliken charges $\{q_A\}$, which in turn depend on the molecular orbital coefficients $c_{\mu i}$. Hence, the equations need to be solved self consistently. At large distances, γ_{AB}

approaches $1/R_{AB}$, while at short distances, it represents electron–electron interactions within one atom. The third order terms describe the change of the chemical hardness of an atom and are computed from DFT. A function Γ_{AB} results as the derivative of the γ-function with respect to charge, and the DFTB3 total energy is then given by:

$$E^{DFTB3} = \sum_{a} f_a \langle \psi_A | \hat{H}_0 | \psi_A \rangle + \frac{1}{2} \sum_{A,B} \Delta q_a \Delta q_B \gamma_{AB} + \frac{1}{3} \Delta q_A^2 \Delta q_B \Gamma_{AB} + \frac{1}{2} \sum_{AB} V_{AB}^{rep} \tag{2.12}$$

For practical purposes, multiple computational modeling softwares have been implemented which contain DFTB method. For example, ADF (Amsterdam density-functional software) [31], Gaussian [32], DeMon[33], which were originally implemented as standard DFT codes. However, the original DFTB implementations can be found in the DFTB+ code [34], which has been used to obtain the electronic structure information of molecules in this thesis.

2.1.3 Limitations

Although DFTB is able to achieve a remarkable speed-up and favorable scaling with system size in comparison to full DFT, it has its own limitations. Owing to the approximate nature, there is a trade-off between computational efficiency and accuracy. One major source of inaccuracy and reduced transferability is the minimal basis set of confined atomic wave functions chosen in DFTB. As a result of the minimal basis, polarization effects, which require more diffuse, delocalized basis function, are often poorly described. This is particularly affecting the description of hydrogen bonds, the hybridization states of nitrogen or phosphorous, or conduction bands in solids [34, 35]. Moreover, as DFTB is based on semi-local density functionals, DFTB inherits the shortcomings known for these functionals. The most relevant deficiency is the lack of (dynamic) long-range electron correlation that DFTB inherits from semi-local DFT. Despite having these limitations, DFTB is a good reference method to describe the main electronic characteristic of organic and inorganic materials. Indeed, In the present work, we will use the output data obtained from DFTB calculations to define the electronic descriptors of small organic molecules.

2.2 Datasets

As discussed earlier, the quality of the dataset used to train any ML model directly affects the performance in molecular property prediction. In this research work, we have used three subsets of the QM7-X dataset [11], which is a recent dataset containing 42 physicochemical (global and local) properties for almost 4.2 million equilibrium and non-equilibrium structures of small drug-like molecules.

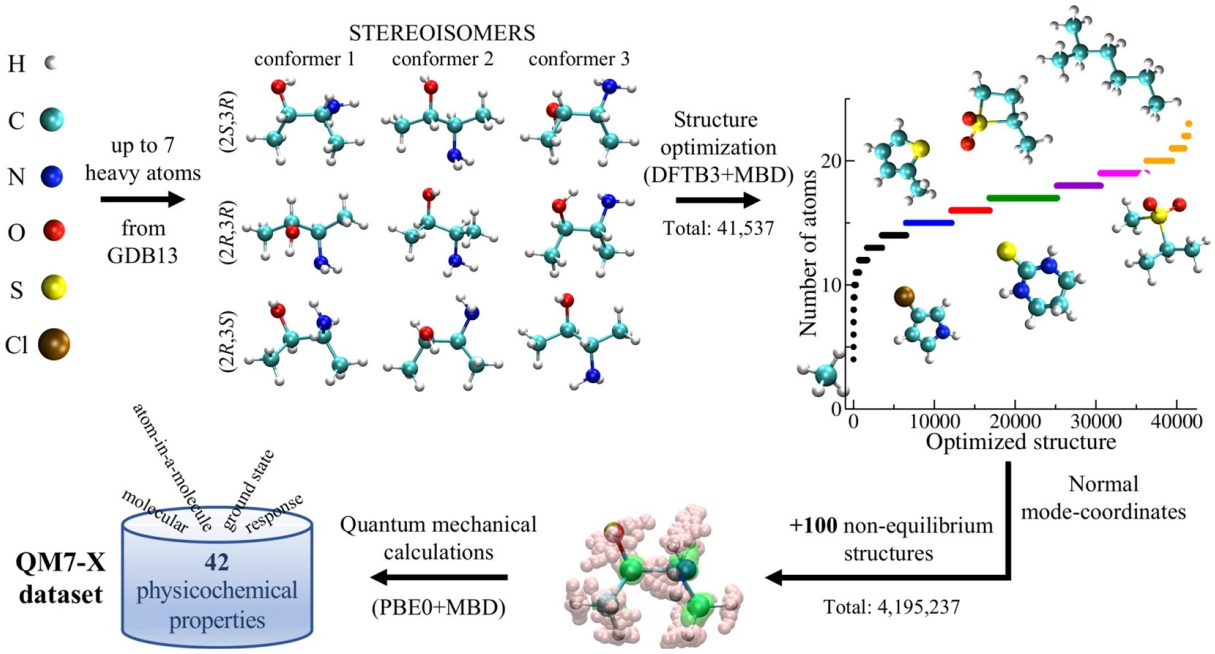
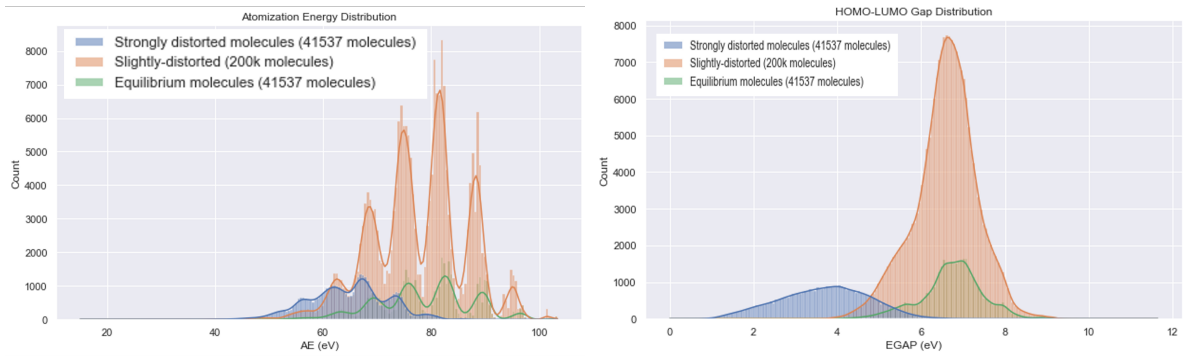


FIGURE 2.2: Generation procedure of the QM7X dataset. Figure taken from [11]

2.2.1 Generation Procedure

Figure 2.2 shows a schematic representation of the generation procedure of QM7-X dataset. QM7-X was built using a set of around 7k molecules with up to 7 heavy atoms (C,N,O,S,Cl) along with Hydrogen in the GDB13 dataset [**Blum09**]. As GDB-13 only contained the chemical connectivity information (encoded as SMILES strings [21]), all the structural/constitutional isomers and stereoisomers were generated for each molecular formula by generating canonical SMILES strings for each possible molecular structure. 3D structures were obtained using MMFF94 force field via the gen3d option in Open Babel. For each of these structures, a set of sufficiently different (meta-)stable conformational isomers was generated. Conformational isomer search was done using the Confab tool, and only those conformers were retained that were within 50 kcal/mol of the most stable structure and differed by a root-mean-square deviation (RMSD) of 0.5 Å. This produced the set of ≈42k equilibrium molecules.

All the structures were optimized using DFTB3 (third-order self-consistent charge density functional tight-binding) method supplemented with a treatment of Many-Body Dispersion (MBD) interactions. For each of the (meta-)equilibrium structure generated, around 100 non-equilibrium structures were generated. This was achieved through a distortion of each equilibrium geometry along a linear combination of their vibrational normal modes computed at the DFTB3+MBD level. This subset contains close to 4 million unique non-equilibrium structures. For these ≈ 4.2 million structures, a set of 42 physicochemical properties were computed using DFT with PBE0 hybrid functional (tight basis set) and MBD dispersion interaction. We will



(A) Atomization Energy distribution (B) HOMO-LUMO gap distribution

FIGURE 2.3: The distribution of Atomization Energy, and HOMO-LUMO gap in the 3 datasets used. The y-axis shows the number of molecules in the respective dataset corresponding to a particular interval of energies in the x-axis. The values for both the properties are in eV.

here use three subsets of the QM7-X dataset: the first subset contains ≈42k highly distorted molecules, the second subset contains ≈42k equilibrium molecules, and the last one is a bigger dataset of ≈ 200k slightly distorted molecules.

2.2.2 Properties

In this thesis, we have used levels with two different levels of theory. The first set of properties is the target properties, which are the properties we attempt to predict, and the second set is of the properties used for calculating the electronic descriptors.

Target Properties: Our work considers two target properties that we attempt at predicting. This includes the atomization energy E_{AT} and the HOMO-LUMO gap E_{gap}. These properties were computed with PBE0+MBD level of theory, and are highly accurate. For the three subsets of the QM7-X dataset used, the distribution of atomization energy and HOMO-LUMO gap are shown in Figure 2.3.

Atomization Energy is used frequently as the first property checked for a prospective ML model predicting physicochemical properties. As it is relatively simple to measure atomization energy, there is an existence of experimental data on the use of different descriptors to predict Atomization Energy. E_{AT} is the measure of energy needed to break down a molecule into its constituent atoms. Hence, we can see in the Figure 2.3(A) that the atomization energy is higher for the equilibrium molecules, than the distorted molecules.

The second property predicted is the HOMO-LUMO gap. E_{gap} is an intensive property, as it depends on both the composition and configuration of the molecule, and not on the system size. E_{gap} is a measure of the difference in energies of the Highest Occupied Molecular Orbital

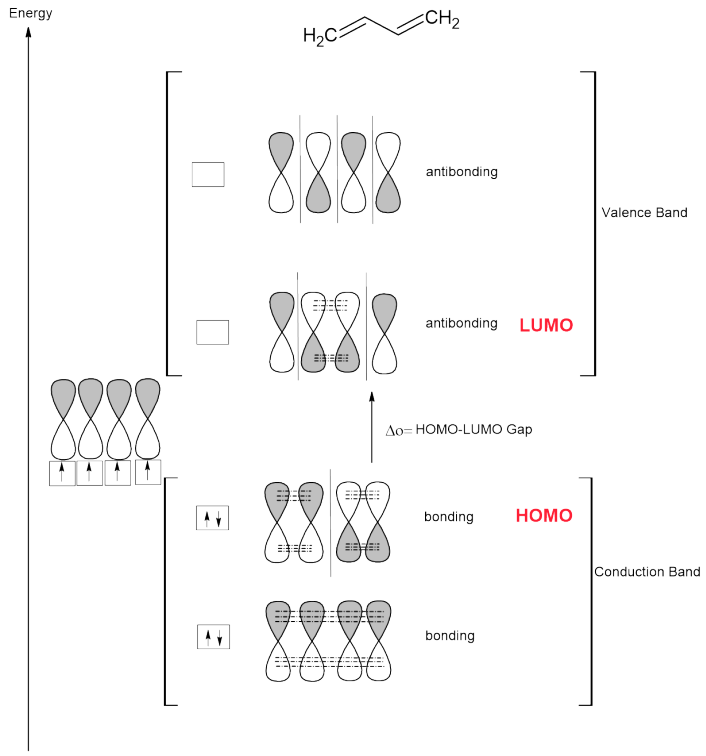


FIGURE 2.4: Molecular Orbital Diagram for 1,3-butadiene. The highest energy orbital of the conduction band is called HOMO, and the lowest energy orbital of the valence band is LUMO. The difference in energy of LUMO and HOMO is denoted by Δ_o, or E_{gap}

(HOMO), and the Lowest Unoccupied Molecular Orbital (LUMO). The HOMO and LUMO are collectively called as frontier orbitals. The highest occupied molecular orbital is obtained through constructive interference of the atomic orbitals (AO). As the name suggests, this is the last orbital that is occupied by an electron in a molecule's ground state. The LUMO, on the other hand is the Lowest Unoccupied Molecular Orbital, which is formed by destructive interference of atomic orbitals. When any molecular excitation happens, electrons move from the HOMO, to the LUMO. Hence the HOMO to LUMO energy gap is an important factor to know about the atomic overlaps, and can be used to predict the strength and stability of molecules.

Electronic Descriptors: The second set of properties were used for computing the electronic descriptors, and were computed using semi-empirical methods. Here, we have made use of DFTB approximations for generating the electronic descriptors, using the DFTB+ [34] package, and these properties have lower accuracy. A total of 11 physicochemical properties from the dataset were used to define the electronic descriptors. The 11 properties used are: Fermi Energy, Band Energy, Number of Electrons, 0th Energy, scc Energy, 3^{rd} energy, repulsive energy, many-body-dispersion energy, Dipole Moment, Eigen Values, Atomic Charges. More information about these properties is given in the table 2.1, and a distribution of some of these properties in the dataset of strongly distorted molecules is provided in the Figure 2.5

Symbol	Property	Unit
E_{fermi}	Fermi Energy	eV
E_{band}	Band Energy	eV
N_{elec}	Number of Electrons	_
E_0	0th order DFTB Energy	eV
E_{scc}	2nd order DFTB Energy	eV
E_{3rd}	3rd order DFTB Energy	eV
E_{MBD}	Many Body Dispersion Energy	eV
E_{rep}	Repulsive Energy	eV
TBdip	Tight Binding Dipole Moment	Cm
TBeig	Tight Binding Eigen values	_
TBchg	Atomic Charges	_

TABLE 2.1: List of the physicochemical properties used as inputs for property prediction.

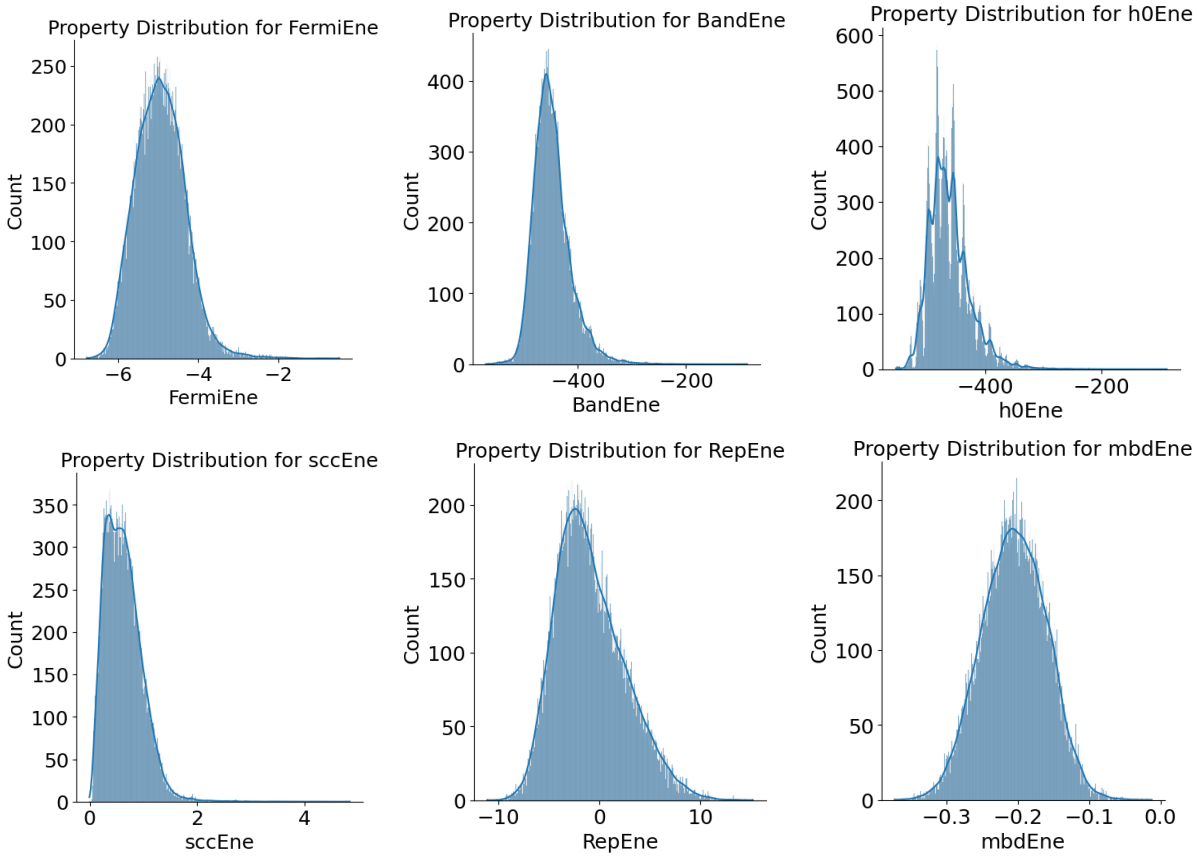
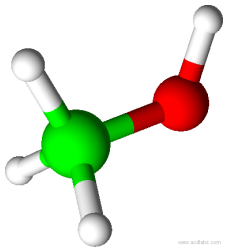


FIGURE 2.5: Distribution of some of the properties taken into consideration for generating the electronic descriptors.

2.3 Machine Learning

Electronic descriptors of molecules will be defined using the information obtained from DFTB outputs, as mentioned before. In this section, we explain the geometric descriptors that will be



(A) 3d structure of methanol molecule

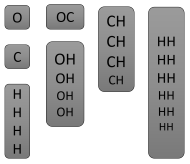
$$C_{ij} = \begin{cases} 0.5\, Z_i^{2.4} & \forall i = j \\ \frac{Z_i Z_j}{|\mathbf{R}_i - \mathbf{R}_j|} & \forall i \neq j. \end{cases}$$

	C	O	H	H	H	H
C	36.9	33.7	5.5	3.1	5.5	5.5
O	33.7	73.5	4.0	8.2	3.8	3.8
H	5.5	4.0	0.5	0.35	0.56	0.56
H	3.1	8.2	0.35	0.5	0.43	0.43
H	5.5	3.8	0.56	0.43	0.5	0.56
H	5.5	3.8	0.56	0.43	0.56	0.5

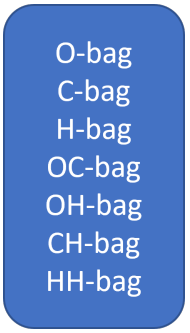
(B) Coulomb Matrix Representation of methanol

	O	C	H	H	H	H
O	O	OC	OH	OH	OH	OH
C	OC	C	CC	CH	CH	CH
H	OH	CH	H	HH	HH	HH
H	OH	CH	HH	H	HH	HH
H	OH	CH	HH	HH	H	HH
H	OH	CH	HH	HH	HH	H

(C) BOB Representation of methanol



(D) Sorting of Bags



(E) Concatenation of bags

FIGURE 2.6: CM and BOB representation of methanol

considered in the present work to train ML model. Then, the concept of neural networks and their related hyperparameters are briefly described.

2.3.1 Geometric Representations

In order to train a ML model, information about molecules needs to encoded in a mathematical form to input to the model. The mathematical representation of the 3D structure of a molecule is called geometric descriptor. We have here used three well-known geometric descriptors: Coulomb Matrix (CM) [14], Bag-of-Bonds (BoB) [22], and SLATM [23].

Coulomb Matrix : The mathematical representation of methanol molecule in the form of coulomb matrix is shown in the Figure 2.6. The elements in the coulomb matrix, C_{ij}, are defined using the atomic numbers of the atoms, Z_i and Z_j, and the interatomic distances R_{ij} between the atoms i and j.

$$C_{ij}^{\text{Coulomb}} = \begin{cases} 0.5 Z_i^{2.4} & \text{for } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{for } i \neq j \end{cases} \tag{2.13}$$

The mathematical formula for calculation of coulomb matrix is given in equation 2.13. The non-diagonal elements of the coulomb matrix represent the coulombic repulsions of the atoms, while the diagonal elements encode a polynomial fit of atomic energies to nuclear charge [14].

The resultant coulomb matrix is a square symmetric matrix, of size (N,N), where N is the number of atoms in the molecule. It can be flattened into a 1D vector while removing the duplicate entries, making the total size of the vector to be $\frac{N^2+N}{2}$. CM representation is invariant to molecular translations and rotations.

Bag of Bonds: The Bag-of-Bonds representation is based on Coulomb Matrices, and uses the principles of the bag-of-words representation used for natural language processing. Here, each molecule is represented as a vector composed of bags, and a bag represents a particular bond type, i.e., a pair of elements. For each combination, the interaction is computed by the formula for coulomb matrix. All the off-diagonal CM entries are sorted and assigned to a "bond" bag. The bags are then concatenated and zero-padding is done to make the bags of the same size. This representation is also invariant under molecular rotations and translations, and permutational invariance is enforced by the sorting step. Compared to CM, BoB provides more information about the molecule and has shown to provide better property prediction results, but at the same time, it's computationally more expensive. However, BoB still lacks spatial information about the molecule, for example dihedral angles, which is a major difference between this and more complex descriptors.

SLATM: Spectrum of London and Axilrod-Teller-Muto (ATM) potential, or the SLATM representation [23] makes use of the charge density of the system, and is much more complex than CM and BOB representations. It is invariant to translation, rotation, and permutation of the system. CM, BOB and SLATM were compared in ref: [23] on the datasets QM7b and QM9 for prediction of total molecular potential energy using Kernel Ridge Regression methods, and SLATM was seen to outperform CM and BOB by a big margin. SLATM is composed of two- and three-body potentials, which are derived from the atomic coordinates and contain most of the relevant information to predict molecular properties [36].

For the SLATM representation, we start by choosing the charge density distribution of the system as an ensemble of electrons partitioned onto different atoms [23]. $\rho(\mathbf{r}) = \sum_i \rho^i(\mathbf{r})$, and $\rho^i(\mathbf{r})$ is approximated as:

$$\rho^i(\mathbf{r}) = Z_i \left(\delta(r - R_i) + \frac{1}{2} \sum_{j \neq i} Z_j \delta(r - R_j) g(\mathbf{r} - R_j) \right)$$

Here, δ is set to the normalized Gaussian function, $\delta(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma}}$, and g(r) is chosen similar to the london potential, $g(R) = 1/R^6$.
The charge density contribution for the i^{th} atom from the j^{th} atom is same as the contribution for the j^{th} atom from the i^{th} atom, hence a factor of 1/2 is added before the summation term to remove the double counting.
The charge density so generated contains unnecessary degrees of freedom, as it is translation and rotation dependent. Hence, $\rho^i(r)$ is projected onto different internal degrees of freedom, associated with well-known many-body terms.
For one-body term, the projection is simply the nuclear charge, Z_i. Whereas, the two-body term is

$$\frac{1}{2}\sum_{j\neq i} Z_j \delta(r - R_{ij}) g(r)$$

The 3-body term is:

$$\frac{1}{3}\sum_{j\neq k\neq i} Z_j Z_k \delta(\theta - \theta_{ijk}) h(\theta, R_{ij}, R_{ik})$$

θ is the angle spanned by the vectors R_{ij} and R_{ik}. The 3-body contribution is $)h(\theta, R_ij, R_ik)$ and is chosen to correspond to the Axilrod-Teller-Muto (ATM) vdW potential[37]:

$$h(\theta, R_{ij}, R_{ik}) = \frac{1 + \cos\theta \cos\theta_{jki} \cos\theta_{kij}}{(R_{ij} R_{ik} R_{kj})^3}$$

The charge density ensemble representation is in essence the concatenation of the different many-body potential spectra. The resultant representation is called as Spectrum of London and ATM potential (SLATM) representation.

So, although SLATM provides more information about the molecule, it is difficult to use it because of it's high computational cost. So, there is a tradeoff between the computational cost and the effiency of the geometric descriptor, and we need to find an appropriate middleground to achieve high accuracy with lower resources.

2.3.2 Neural Networks

A Neural Network (NN) is a collection of artificial neurons, which collectively work as a function to map a set of input features to the output. These artificial neural networks work in a manner similar to the transport of information in the neurons of our body's biological neural network. A basic structure of a multilayer artificial neural network is given in the Figure 2.7. As visible in the Figure 2.7, a set of input features {I_1, I_2, ..., I_n} are mapped to the outputs {O_1, O_2, ..., O_w}, through certain hidden layers. The circles shown in the figure are the neurons, and each

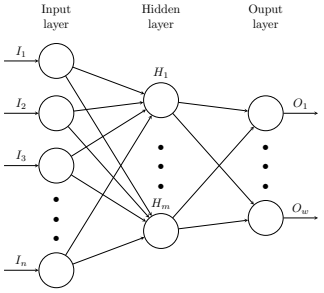


FIGURE 2.7: This figure denotes a sequential multi-layered neural network. The circles denote the neurons of a layer, and the lines denote connections between 2 neurons. This is a fully-connected neural network, which means that each neuron of 2 adjacent layers are connected to each other.

line connecting the neurons is assigned a weight and each neuron has an associated bias. The output of j^{th} neuron of the hidden layer will be given by:

$$H_j = f(\sum_{i=1}^{n}(w_{ij} \cdot I_i) + b_j) \tag{2.14}$$

Here, w_{ij} is the weight associated with connection of the i^{th} neuron of the input layer and the j^{th} neuron of the hidden layer. b_j is the bias, or an offset associated with the j^{th} neuron of the hidden layer, and "f" is an activation function employed to remove the linearity from the model, which will be discussed below.

We discuss a few components of a neural network as follows:

Dense Layers : Figure 2.7 shows a simple *sequential NN* architecture, also called as multi-layered perceptron. A sequential NN is composed of layers of neurons, called as the *Dense Layers*. The neurons of these layers are fully connected as discussed previously. These neural networks work sequentially, layer-by-layer, transferring the information from the input layer to the hidden layers, followed by the output layer. The layers are called dense, because the neurons of adjacent layers are fully connected, and data propagation happens only in 1 direction. This different from other architectures like convolutional layers, where the neurons are not fully connected, or the recurrent networks, where data is propagated in multiple directions. Dense layered sequential networks are the most basic types of neural networks, and are the most relevant to this project.

Activation Functions The function 'f' used in the Equation 2.14 is an activation function, which is used to remove linearity from a neural network model. Typical activation functions are tanh, relu or sigmoid. The presence of the activation function allows the neural networks to handle non-linear data and decode complex patterns and relationships between the features. A

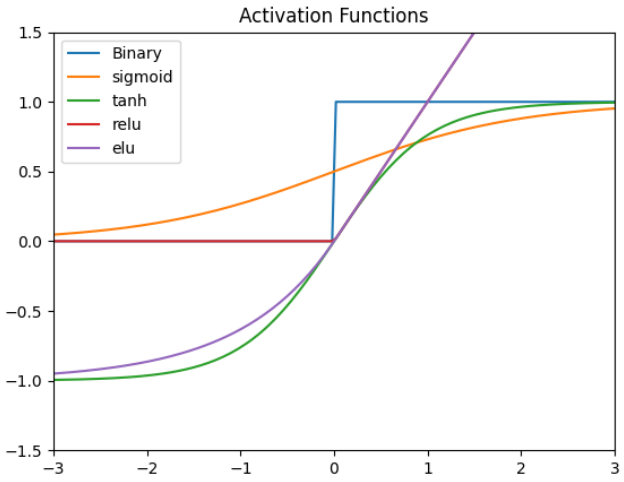


FIGURE 2.8: Comparison for various common activation functions.

graphical representation of different activation functions is shown in the Figure 2.8. One can here see that elu, or exponential linear unit, is based on relu and contains a parameter called α which denotes the smoothness of the curve when the input is negative.

The tunable parameters like the choice of the activation function, number of neurons in each layer and total number of layers are collectively called as hyperparameters.

Training neural networks Initially, random weights and biases are assigned to the network, and these are then improved by using a trial-and-error approach called as backpropagation. This step is referred to as training of the neural network. During this propagation and backpropagation of information in the neural network, the model tries to change the weights and biases in a manner that minimizes the error. To do this, a "loss function" is defined based on some statistical parameters. The change of weights and biases for minimizing the loss is governed by the "optimizer" method. A simplest example of an optimiser is the Gradient descent, which changes the weights based on a "learning-rate" and the slope of the loss function. If the number of hidden layers are increased, the training method is referred to as "deep-learning". The model tries to reach a minima of the loss function based on the slope of the loss function. While changing the weights and biases in backpropagation, the gradient of the loss function is typically multiplied by a *learning rate* which determines the size of the step to take. Figure 2.9 shows the impact of learning rate on the training efficiency. A very small learning rate will take a long time to converge, and will have high chances to converge to a local minima. A higher learning rate, on the other hand, will be too chaotic and cause drastic updates which will not be able to converge to a minima.

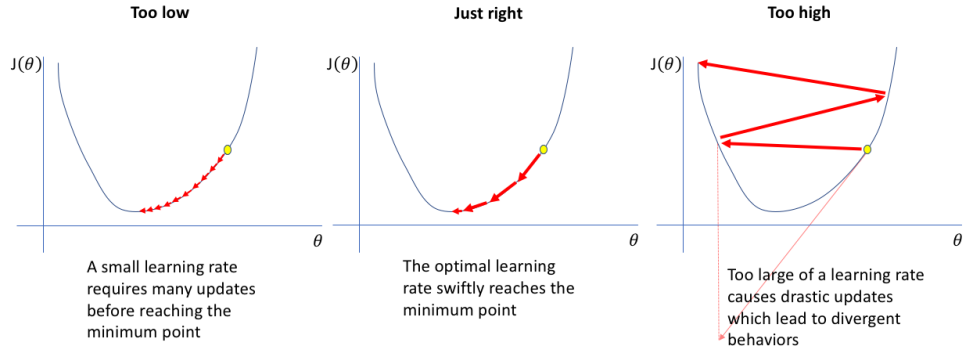


FIGURE 2.9: Importance of an optimum learning rate. (Figure taken from [38]). J(θ) is the loss function calculated for a given parameter θ. The minimization of loss is done via adjustment of weights and biases by the optimiser method.

Typically the training procedure requires a large number of iterations (called as "epochs" in machine learning). Hence, the more the training data is available to the model, the more accurate weights and biases would be, and the better the model would perform. Errors in the data are usually called outliers, and these outliers cause problems in the training process. So, the quality of data is important for the training of the model. Another drawback of neural networks is that the curves for the loss function are usually not as smooth as given in Figure 2.9, and contain multiple local minimas. The model might converge to a particular local minima rather than the global minima, and since the dependence is on multiple parameters, the optimum training depends on an experienced-choice of hyperparameters.

k-fold Cross Validation For efficient training of a neural network model, the input data is split into training and testing sets, so the model can be trained on the training set, and the efficiency of the model can be measured on the testing set. Another way to efficiently validate your model is the cross-validation method. In k-fold cross validation method, the input data is split into k equal parts, and (k-1) sets are used for training the model, and the last set is used for testing the model. Then the entire process is repeated 'k' times, such that each subset has been used at least once as the testing set. The testing error for each case is taken and the final cross-validation error is reported to be the average of each error. An illustration of the process is shown in the Figure 2.10. The number of folds 'k' are chosen with respect to the the computational resources available, and also on the available data points.

Overfitting Another problem in the use of neural networks is "overfitting", where the model starts to memorize the training data and hence fails to perform well on the testing data. One of the causes of overfitting is when there are large number of trainable parameters available to the network. In this case, the model would be able to perform well on the training data because of the ability to detect patterns, but it wouldn't be able to learn the trend of the data, and hence

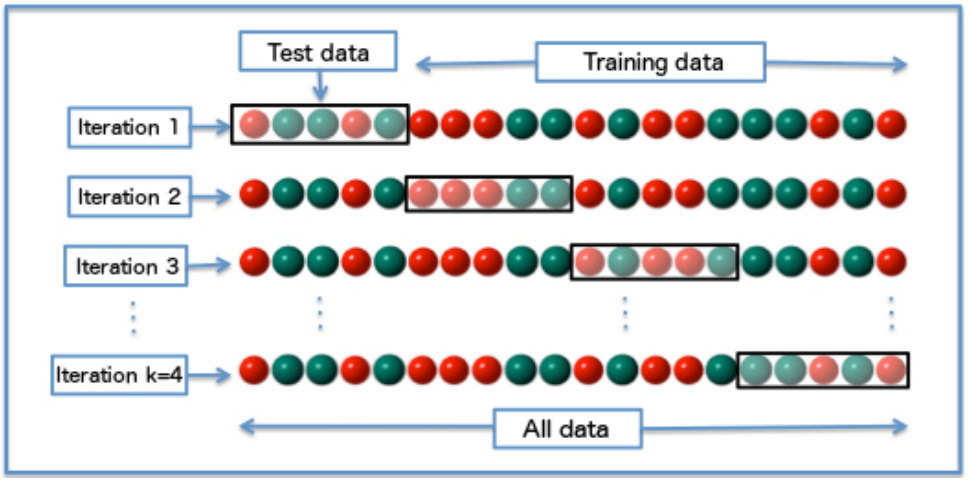


FIGURE 2.10: Illustration of the k-fold cross validation method. Figure taken from Wikimedia [39]

fail on the test set. Another cause of overfitting is when training is performed on smaller training set for too long. Feeding the same data multiple times to the model will allow it to learn the output, but with unseen data (test set), the model will fail. Overfitting can be prevented by penalizing the network during the training process thereby increasing the training error, but lowering down the testing error. This process enforces the network to not memorize the patterns in the training data. It can be done via methods like dropout [40], regularization, etc.

2.3.3 Convolutional Neural Networks

Convolutional Neural Network, or CNN [41] is a deep-learning algorithm that makes use of a specific layer known as the convolutional layer. The name of convolutional layers comes from the use of a mathematical operation called convolution. Mathematically, convolution operation works on two given functions as:

$$s(t) = \int x(a)w(t-a)\,dx$$

.

This operation can be denoted by an asterick (*) as: $s(t) = (x * w)(t)$. For the case of the CNNs, $w(t)$ is the trainable kernel that will be "convolved" with the data in the neural network ($x(t)$) to produce output data which will propagate through the network. An example of 2-dimensional convolutional operation is given in Figure 2.11.

Similar to the approach used for simple sequential neural networks, convolutional neural networks also make use of neurons, and the network works by assigning weights and biases. The difference

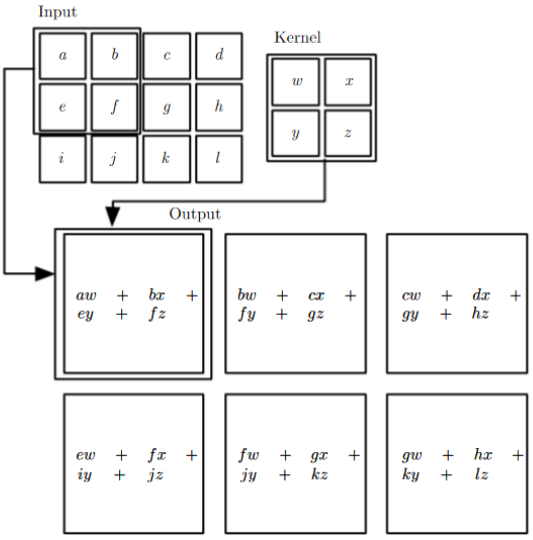


FIGURE 2.11: An example of 2D convolution operation. (Figure taken from [42]).

is, that for convolutional layers, instead of having a weight associated with each neural connection, there is a kernel matrix that convolves with the inputs to the layer to produce the output, which is propagated forwards. In traditional neural networks, the weights of any layer can be represented as a matrix, and output of the layer can be calculated by taking the dot product of the input matrix with the weights matrix. In CNNs, on the other hand, the output of the convolutional layer is calculated by taking the convolution of the input matrix with the kernel filter. Since the kernel matrix used in practice is usually smaller in size compared to the input matrix, CNNs are called sparse-connected networks, because there is no direct connection between each neuron of the layers. Accordingly, CNNs store fewer number of parameters for each convolutional layer, reducing the computational cost, while improving the model's statistical efficiency. The kernel matrix used is smaller in size, so that it can detect small features of the input, (e.g., a particular edge in the image of a huge building).

Owing to the nature of the convolution operation, the input matrix needs to be padded along the borders so that the border values are not lost in the convolution operation. Usually, the matrices are zero-padded, i.e., they are padded with 0s on the borders. Another hyperparameter in CNNs is stride, which denotes the number of pixels of the input matrix that the kernel filter will move on each iteration. By default, stride is taken as 1 in the convolution operation, which is shown in the Figure 2.11, if the stride is taken as 2, the kernel matrix will move 2 pixels on each iteration, both across the row, and down the column.

If the input vector is of size, 'W', the filter kernel is a vector of size 'F', the amount of zero padding is 'P', and the stride is 'S', then the output vector of the convolutional layer will have size, $O = \frac{W-F+2P}{S} + 1$. This increases the size of the vector, and at the same time, convolutional

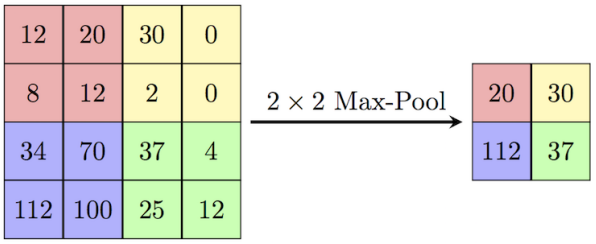


FIGURE 2.12: An example of 2x2 max pooling operation.

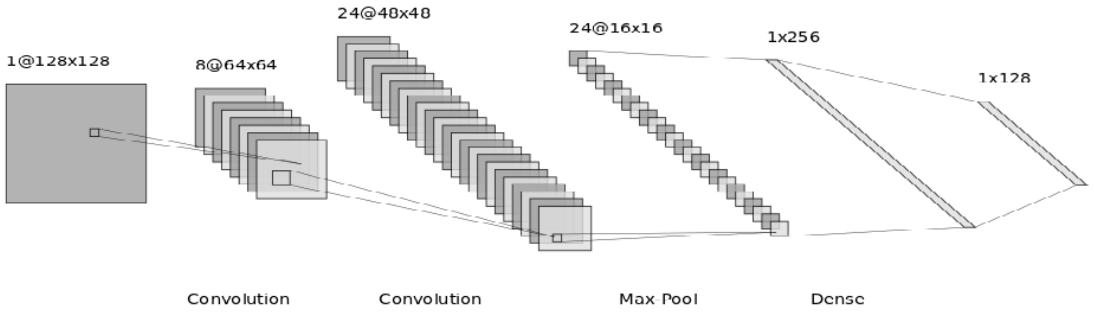


FIGURE 2.13: A typical CNN architecture showing the convolutional layer, pooling layer

layers also increase the number of channels of the input, where each channel works on identifying a different feature of the input data. Thus, the size of the data propagating through the network increases, which increases the computational load. To tackle this problem, CNNs make use of another type of layer called as the pooling layer, which helps in downsampling the data efficiently. The Figure 2.13 shows examples of convolutional and pooling layers in a typical CNN architecture. Here, the pooling layer is represented by 'Max Pool' denoting the Max Pooling operation, which is shown in the figure 2.12

2.4 SchNetPack

SchNetPack [43] is an end-to-end deep-learning framework to develop NN model of atomistic systems. It includes a complex NN architecture successfully used for the prediction of a range of physicochemical properties of diverse molecular systems across the chemical space. Schnetpack considers the implementation for the SchNet [44] NN architecture which follows the deep tensor neural network (DTNN) framework [45] based on continuous-filter convolutions. As shown in the Figure 2.14, any atomistic system with 'n' atoms is initially represented by the set of nuclear charges and the position vectors for each atom. Through the layers of the SchNet architecture, the atoms are described by a tuple of features $X^l = (\boldsymbol{x_1^l}, ..., \boldsymbol{x_n^l})$, where 'l' is the current layer.

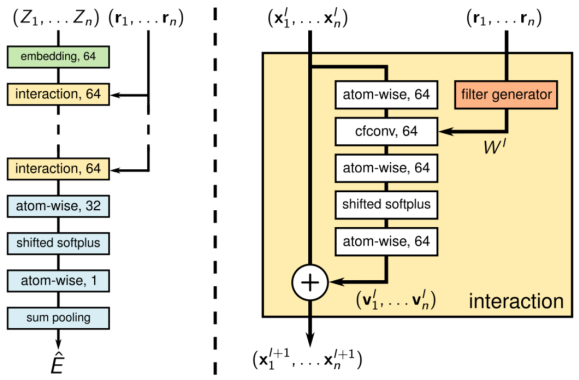


FIGURE 2.14: Illustration of the SchNet architecture (left). The right part shows the interaction block.

These are initialized randomly and optimized during the training of the network. By definition, these representations only include information about the atoms individually, discarding any information about the environment. The Figure 2.14 shows 'atom-wise' layers which are dense layers used for training and for each atom 'i', the output of layers is given as: $\boldsymbol{x_i^{l+1}} = W^l \boldsymbol{x}_i^l + \boldsymbol{b}^l$.

As it is shown in right part of Figure 2.14, the interaction blocks of Schnet make use of 'cfconv' layers, which are the continuous filter convolutional layers. This is similar to the convolutional layers explained above, but instead of having a particular kernel filter, schnet models the filters *continuously* with a separate filter-generating neural network. These interaction blocks model the pairwise interaction of a given atom with its environment via the cfconv layer. For a given atom 'i' and layer 'l', the interaction of the atom with all the surrounding atoms is given by:

$$\boldsymbol{x}_i^{l+1} = (X^l * W^l)_i = \sum_{j=1}^{n_{atoms}} \boldsymbol{x}_j^l \circ W^l(\boldsymbol{r}_j - \boldsymbol{r}_i)$$

. Here, * represents the convolution operation and X^l denotes the atomwise representations at positions R. $\circ$ represents element-wise multiplication. The filter-generator is a fully connected neural network which models the continuous filter W^l used by the cfconv layer.

Schnet architecture has been widely used for different studies, e.g., the prediction of total energy U_o of ≈131k small organic molecules with up to 9 heavy atoms considered in QM9 dataset. The results are shown in the Figure 2.15. This figure shows the learning curves, i.e., the variation of the mean absolute error (mae), in eV on the y-axis, as a function of the training set size (number of molecules used for training), on the x-axis. 'T' represents the total number of interaction blocks used to train the NN model. As we can see, after increasing the number of interaction blocks, MAE for the model reduces and all the Schnet models shown in the figure performed better than the best DTNN models.

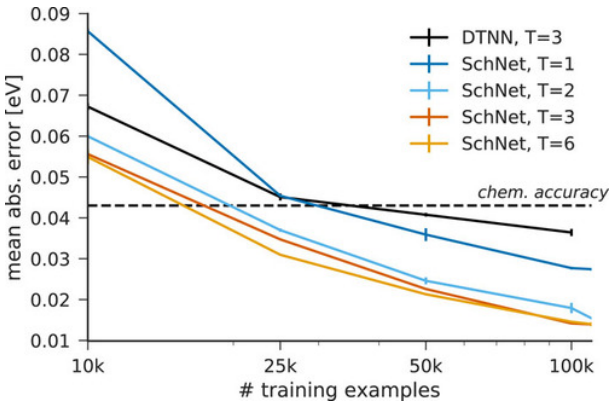


FIGURE 2.15: MAE of energy predictions by Schnet for different number of interactions compared with the best performing DTNN model. Results from the paper [44]

Schnet has been shown to be useful for not only property predictions, but also the predictions of potential energy surfaces and energy-conserving force fields. It follows rotational, translational and permutational invariances, and is able to perform fast and accurate predictions. The architecture is currently state-of-the-art. The results from the NN models generated using the novel geometric/electronic descriptors in this thesis are compared with the obtained using SchNet models.

Chapter 3

Results and Discussions

3.1 Molecular Property Space

As mentioned in the previous chapter, the goal of the present thesis is to define efficient molecular representations by combining geometric and electronic descriptors, and compare the representations by training efficient neural network (NN) models. The atomic coordinates and (extensive/intensive) physicochemical properties of diverse organic molecules used for training and testing the NN models were taken from subsets of the QM7-X dataset. The first subset contains 41,537 strongly distorted (non-equilibrium) molecules and the second one is a larger dataset of 207,685 slightly distorted molecules. The last subset is of equilibrium molecules and has a size of 41,537. On each of these datasets, separate computations were done using third-order density functional tight-binding (DFTB3) method supplemented with a many-body dispersion (MBD) interaction, and eleven electronic properties were stored to form the electronic descriptors in a later step. DFTB+ package [34] was used for the calculation of the DFTB properties.

Extensive properties are strongly correlated with the system size compared to intensive properties, and therefore easier to learn by using only a geometric descriptor. For instance, atomization energy E_{AT} being an extensive property is more likely to be predicted better by geometric descriptors than electronic descriptors. These properties are also less impacted by the structural distortion of the molecular conformation. The distortion of the bonds adds tension to the molecules and therefore makes them easier to break, which results in a lower atomisation energy. The difference in atomization energy for two conformations of C4NH7 is shown in the Figure 3.1. On the other hand, intensive properties such as E_{gap} do not depend on the system size, but strongly correlate with the atomic constituents, and molecular conformation. As seen in Figure 3.1, there is a 54% reduction in the value of E_{gap} from the equilibrium conformation (Figure 3.1(b)) to the distorted one (3.1(a)).

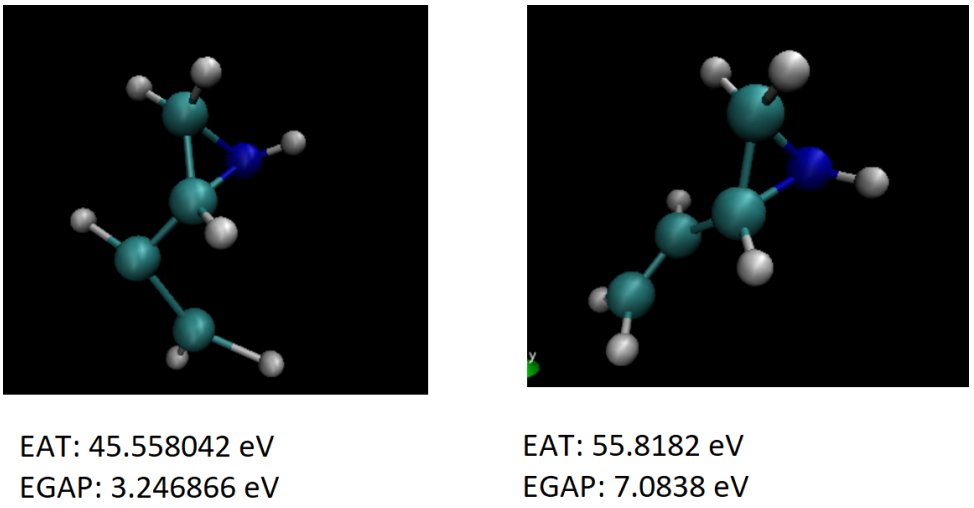


FIGURE 3.1: EAT and EGAP for equilibrium and distorted conformation of C4NH7. The light blue balls denote carbon atoms, dark blue ball is Nitrogen atom, and white balls are hydrogen atoms.

The 3D structure of molecules were extracted from QM7-X dataset and saved in 'xyz' file format. The graphical 3D geometries were generated using Visual Molecular Dynamics(VMD) software [46].

3.2 Neural Networks Implementation

The neural network architectures were developed using Keras [47] library in python. The computations for various geometric descriptors was done using QML code [48]. The Coulomb Matrix generated was flattened from the square symmetric representation, to a 1D numpy array [49] containing only the lower triangle of the matrix. This reduced the size of the input and accelerated the calculations. For all the neural networks, Adam optimizer [50] was used with learning rate varying from 10^{-4} to 10^{-6}. Mean Absolute Error (MAE) was used as a metric for training, and all the weights and biases were initialized with HeNormal initializers [51]. Various activation functions like tanh, relu, and sigmoid were tested, and the best performance was obtained in most architectures using 'elu' activation function [52].

The results of the neural network architectures were compared with that of kernel ridge regression, which were obtained using the QML package. The σ and λ parameters in the kernel regression were optimized using principal component analysis algorithm [53]. All results presented in this project have been obtained using QML and a supercomputer located at the Technical University of Dresden in Germany.

3.2.1 Neural Network Architectures

In this section, we will define the distinct neural network architectures developed in the project. In each of the architectures mentioned, 3 types of geometric descriptors were used: Coulomb Matrix (CM), Bag-of-Bonds (BoB), and SLATM. In the dataset, the largest molecule contains 23 atoms, consequently, for this dataset, the size of CM generated is 276, size of BOB is 528, and size of SLATM is 17,895. Thus, the computational cost increases as we move from CM to BOB or to SLATM. In order to ensure a representative sample for the training and test sets, a random generator was used to supply non-repeating indices for molecules within the dataset, with molecules from those indices being extracted to form the desired set. The data is split into training, validation, and testing sets. As explained in the prevoius chapter, the *training set* is used for the modification of weights and biases in the backpropagation process. The model is validated on the *validation set*, based on which the learning rate is modified. We have used a patience of 100 epochs, which means that if the model fails to show much improvement in the validation loss over 100 epochs, the learning rate is reduced by a factor of 0.5. This process of training and validation is repeated for 20,000 epochs, and then the model is finally tested on the *test set*. To ensure the results are comparable, the validation and test set used for different training set sizes were always the same - this was ensured by supplying the same number as a seed to the random generator.

In the first architecture, a simple sequential neural network was trained. The network contained a total of 3 hidden layers, and the number of neurons in each layer were tuned using Hyperparameter tuning methods. The architecture is shown in the Figure 3.2. This simple dense NN architecture used only geometric descriptors as input. In the Figure 3.2, the input layer shows size of 276, for Coulomb Matrix. For BOB, it would be 528, and for SLATM, 17895. The activation function used here was ELU (Exponential Linear Units). To avoid confusion, this architecture is referred to as the Sequential Architecture 1.

The second architecture is similar to the first one, but instead of having only the geometric representation in the input features, the geometric and electronic representations are concatenated in the data pre-processing step and then provided as input. The hyperparameters for this architecture are same as that of the previous one. Both of these architectures were used for prediction of both atomization energy E_{AT} and HOMO-LUMO gap E_{gap}. This NN architecture is referred to as the Sequential Architecture 2 in the next section. The second architecture was also trained making use of data standardization of the DFTB properties in the pre-processing step.

As a 3rd architecture, the geometric and electronic descriptors were trained separately for a few layers, and then the layers were concatenated together and the training proceeded in a sequential manner. The architecture is shown in Figure 3.3. The Figure here shows training

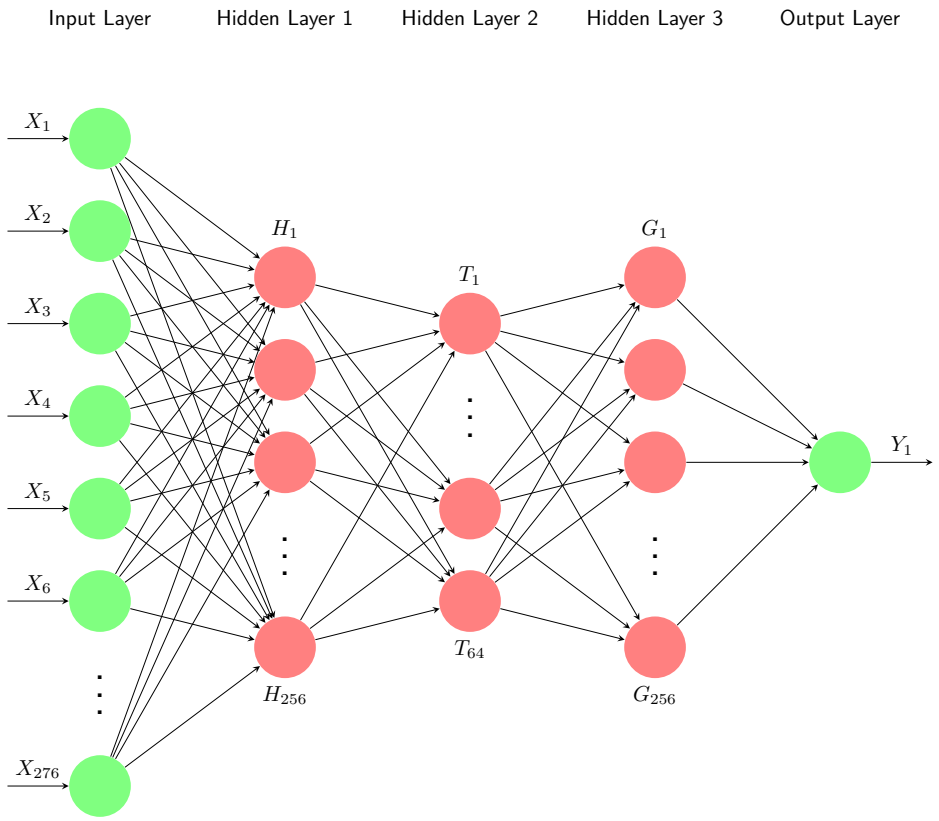


FIGURE 3.2: The first neural network architecture making use of only the geometric descriptors

of the geometric and electronic descriptors separately for 3 and 1 hidden layers, respectively, followed by a concatenation of the layers. After the concatenation, the network is trained for another layer, followed by the output layer. As the size of the electronic descriptor is lower than that of the geometric descriptors, the number of hidden layers before concatenation are lower for the electronic descriptors, to avoid loss of data. This architecture is called as the Sequential Architecture 3 in the next section.

Similar to the approach used with the three sequential networks, two Convolutional Neural Network Architectures were developed. In the first CNN, the geometric and electronic descriptors were concatenated and passed to the network. The number of channels used by the convolutional layer were tuned using Random Search method. In the second CNN architecture, the geometric and electronic descriptors were treated separately for a few layers, followed by their concatenation. Due to the increased number of channels in the convolutional layer, the data propagated through the network becomes 3-dimensional. Hence, for the concatenation operation to work, 2 of the 3 dimensions need to match for the 2 layers being concatenated. Hence, the kernel size had to be chosen accordingly to allow the size of the layer outputs to match, as shown in the Figure 3.4b.

All the mentioned neural network models were trained on all the three datasets, i.e., strongly

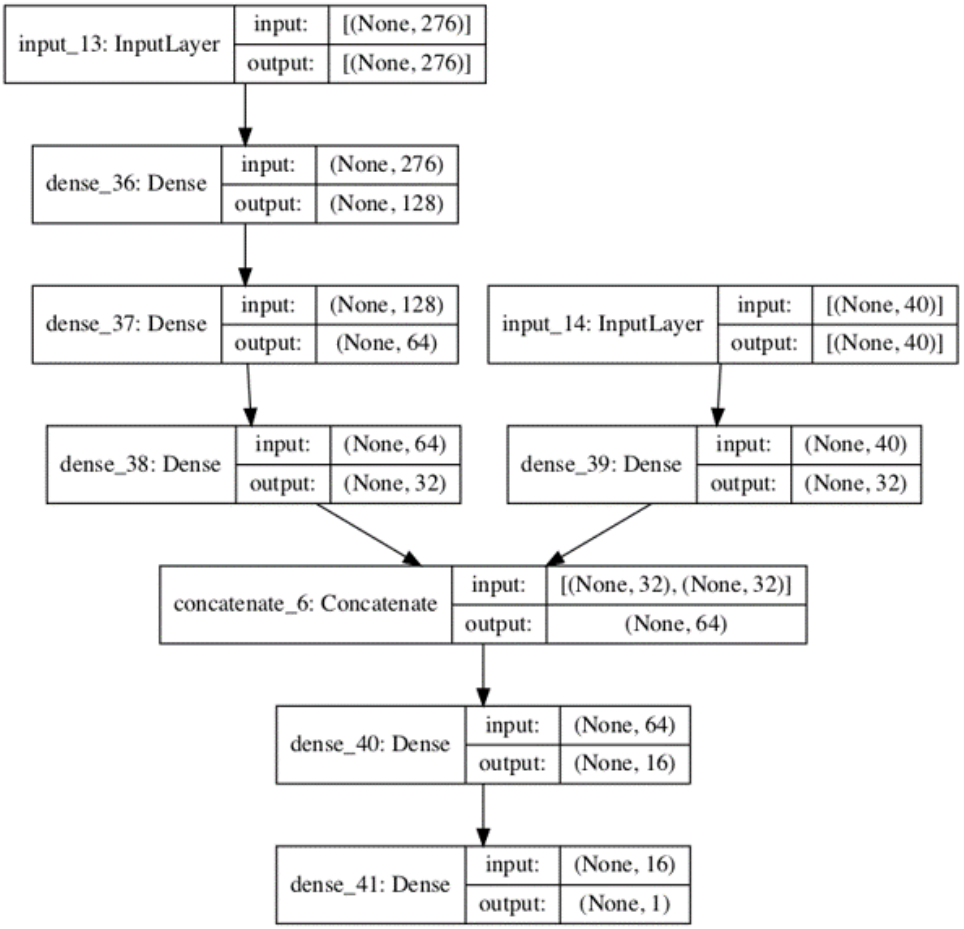
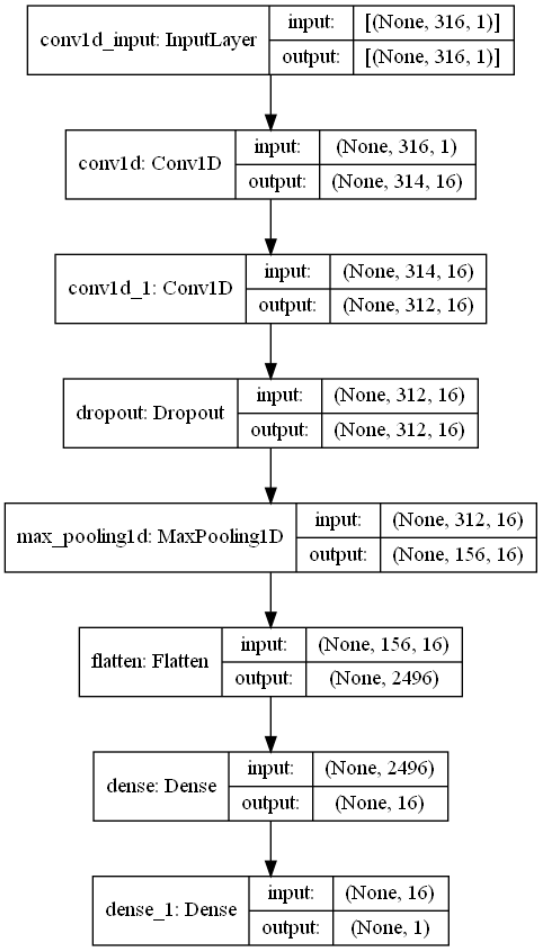


FIGURE 3.3: The third neural network architecture training the geometric and electronic descriptors separately followed by their concatenation.

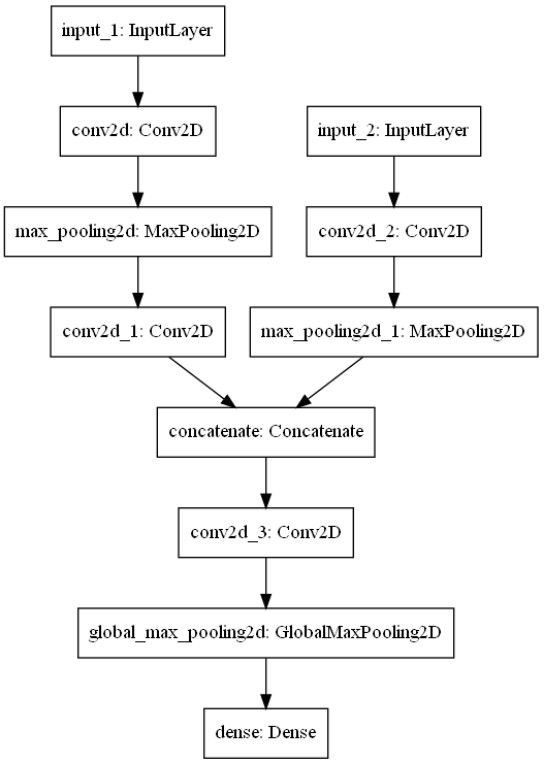
distorted molecules, slightly distorted molecules and the equilibrium molecules. For the distorted and equilibrium molecules, the training size is varied from 1k to 30k molecules, validation size is taken as 2000 molecules, and test set is 10k molecules. However, the dataset of slightly distorted molecules is a larger dataset containing 207,685 molecules, and for this, the training size was varied from 1k to 50k, the validation size was of 5000, and the remaining molecules were used for testing.

3.2.2 Comparison of the models

The usual way of tracking the performance of a NN model is through a learning curve. In a learning curve, the error of the NN model is plotted with respect to the training set size, i.e. with respect to the total number of molecules used for training the model. As we have seen, if more data is available to a neural network, the model will be able to learn more about the chemical space represented by the dataset, and the error is expected to be reduced. However, we have seen that due to overfitting, or due to improperly tuned hyperparameters, the error



(A) CNN architecture using combined electronic and geometric descriptors as input



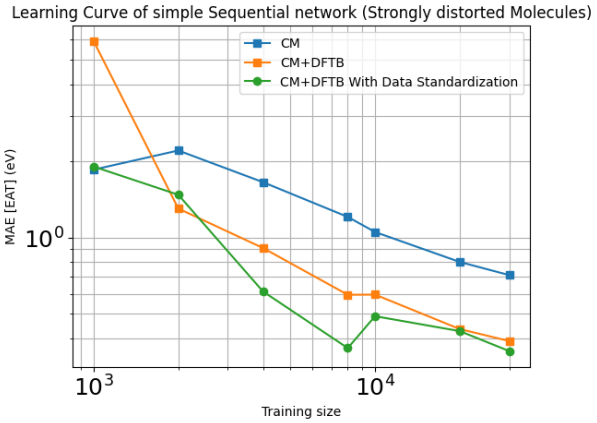
(B) CNN architecture for handling the electronic and geometric descriptors separately for a few layers before their concatenation.

FIGURE 3.4: Convolutional Neural Network Architectures using the electronic and the geometric descriptors. (A) shows the Electronic descriptors combined with the coulomb matrix representations making the input size of 316, while (B) shows the geometric and electronic descriptors handled separately.

sometimes increases with the increasing training size. These learning curves will help us to identify the performance and the errors of the models.

There are various metrics that can be used as error representation while plotting the learning curve. More oftenly, mean absolute error (MAE) or mean squared error (MSE) are used as metric for performance measurement. MAE can be defined as $MAE = (\frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i|$; where y_i and x_i are the true and predicted values of the model, and 'n' is the total size. Notice that using the plot of MAE vs training set size as the sole metric can sometimes be misleading while comparing multiple NN models. In that case, we make a histogram of the difference between the true and the predicted values for a particular training set size, to know the distribution of errors. The performance of the models for different geometric descriptors has been analysed in this section using learning curves and histograms. All this analyses are initially done for the strongly distorted molecules, because the physicochemical properties of these molecular structures have

(A) Error Distribution (MAE) of the sequential neural network architectures, for the training size of 30k molecules



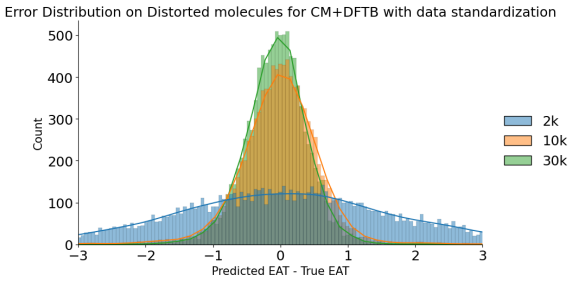
(B) Learning Curve for the sequential networks in loglog scale.

FIGURE 3.5: Error distribution of Sequential Networks for prediction of E_{at}, using Coulomb Matrix as the geometric descriptor on the dataset of strongly distorted molecules.

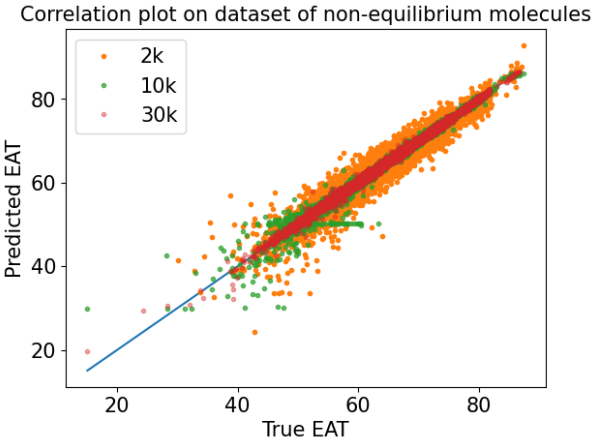
been shown to be a challenge for current ML methods. After this, the results are also verified on the equilibrium dataset and the larger dataset of slightly distorted molecules.

For the sequential architecture 1 and 2 described in the previous section, the learning curve and histogram of error distribution for E_{AT} prediction are shown in the Figure 3.5. From the learning curve in Figure 3.5b, we can see that for a small training size of 1k molecules, coulomb matrix is able to predict atomization energy better than when using it along with electronic descriptors. But for higher training set sizes, the sequential architecture 2, which makes use of both the electronic and geometric descriptors, has performed better than the architecture 1. The Figure 3.5a shows that error distribution is more 0-centred when data standardization is used for the DFTB properties. The histogram also shows that the prediction error by architecture 2 for most of the molecules is close to 0, and mostly lies between -1 to +1 eV. Whereas, in the case of architecture 1, there are more outliers, and the error distribution lies between -3 to +3 eV. This shows that neural network is able to extract almost 2-times more information about the atomization energy when tight-binding properties are also considered. The increase in error from training set size of 8000 to training size of 10,000 in Figure 3.5b denotes an error due to overfitting.

A comparison of the errors obtained on the second sequential network using CM+DFTB with data standardization is provided in the Figure 3.6. This allows us to visualise the distribution of error, and we can observe that for a smalling training set size, the variance of error is higher. Similar results were obtained in the prediction of HOMO-LUMO gap, which is shown in the Figure 3.7. For further discussions, the DFTB properties are always standardized while pre-processing before passing to the neural network.

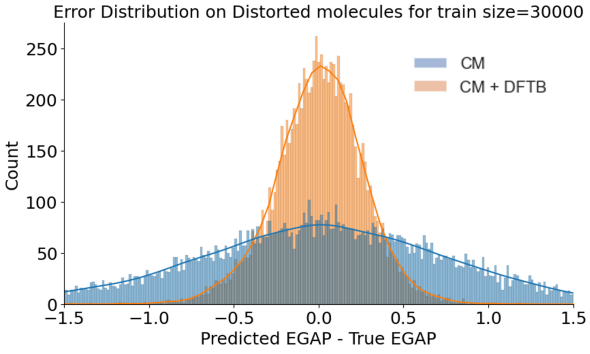


(A) Error Distribution (MAE) for the second sequential network using CM+DFTB with standardization of properties for 3 different training set sizes.

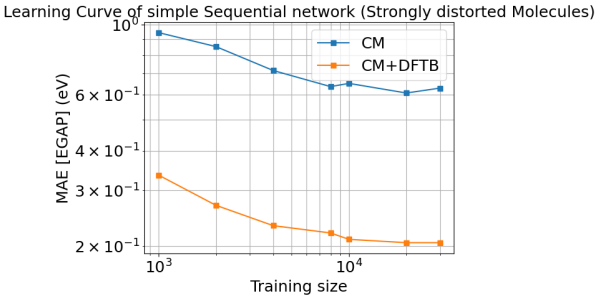


(B) Correlation plot for 3 different training set sizes for the second sequential network

FIGURE 3.6: Error distribution and correlation plot of Sequential Networks for prediction of E_{at}, using Coulomb Matrix + DFTB properties with data standardization on the dataset of strongly distorted molecules.



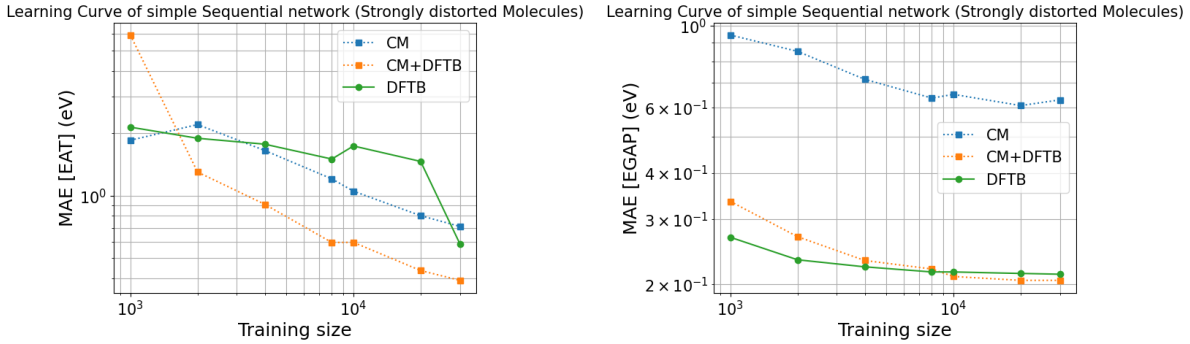
(A) Error Distribution (MAE) of the sequential neural network architectures, for the training size of 30k molecules for HOMO-LUMO gap



(B) Learning Curve for the sequential networks in loglog scale.

FIGURE 3.7: Error distribution of Sequential Networks for prediction of E_{gap}, using Coulomb Matrix as the geometric descriptor on the dataset of strongly distorted molecules. DFTB properties were standardized while preprocessing.

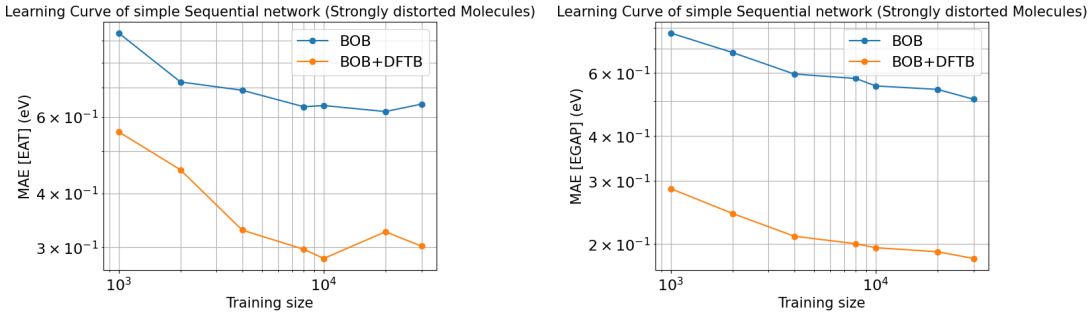
As discussed in the previous section, for an extensive property like E_{AT}, geometric descriptors are important for property prediction because they encode the structural information of the molecules. When the models were trained for property prediction of E_{AT} and E_{gap} using only the DFTB properties, without the geometric descriptors, the results are shown in the Figure 3.8. As expected, the DFTB properties are able to predict the HOMO-LUMO gap properly without any geometric descriptor, with almost the same error as the sequential architecture 2. However, for the prediction of atomization energy, using only DFTB properties fails to provide good results and the overall error for training size of 30k is 0.586 eV, while the error with the sequential architecture 2 is 0.357eV. This highlights the difference in approach needed for the prediction of extensive and intensive properties.



(A) Learning curve for the prediction of atomization energy (in loglog scale)

(B) Learning Curve for the prediction of HOMO-LUMO gap (in loglog scale)

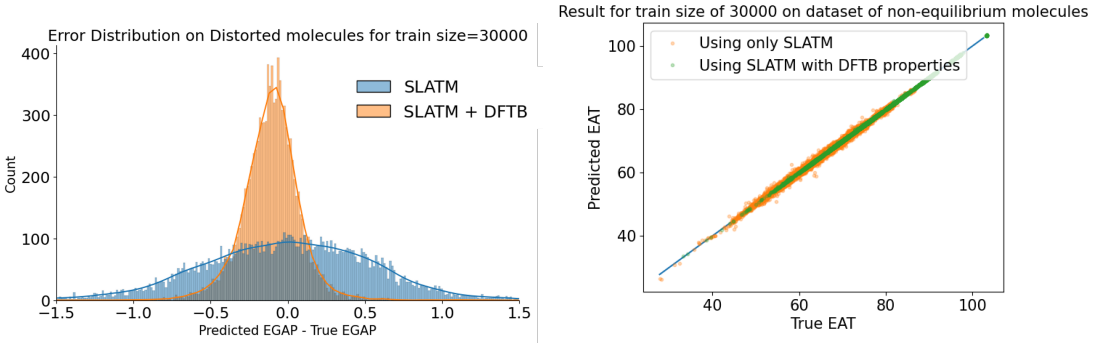
FIGURE 3.8: Error distribution of Sequential Networks for prediction of E_{gap}, using Coulomb Matrix as the geometric descriptor on the dataset of strongly distorted molecules. The DFTB properties are standardized in both cases.



(A) Learning curve for the prediction of Atomization energy gap using Bag-of-bonds (in loglog scale)

(B) Learning Curve for the prediction of HOMO-LUMO gap using Bag-of-bonds (in loglog scale)

FIGURE 3.9: Learning Curve for the Sequential networks for predicting EGAP and EAT using Bag-of-Bonds as the geometric descriptor.



(A) Histogram for prediction of atomization energy, using SLATM.

(B) Scatter plot for comparison of models using SLATM, with and without the use of DFTB properties.

FIGURE 3.10: Learning Curve for the Sequential networks for predicting EAT using SLATM as the geometric descriptor.

While using the sequential architecture 3 as shown in Figure 3.3, the final error obtained on a training set size of 30k molecules was 0.4132 eV, which is slightly more than 0.357 eV which was obtained on sequential architecture 2.

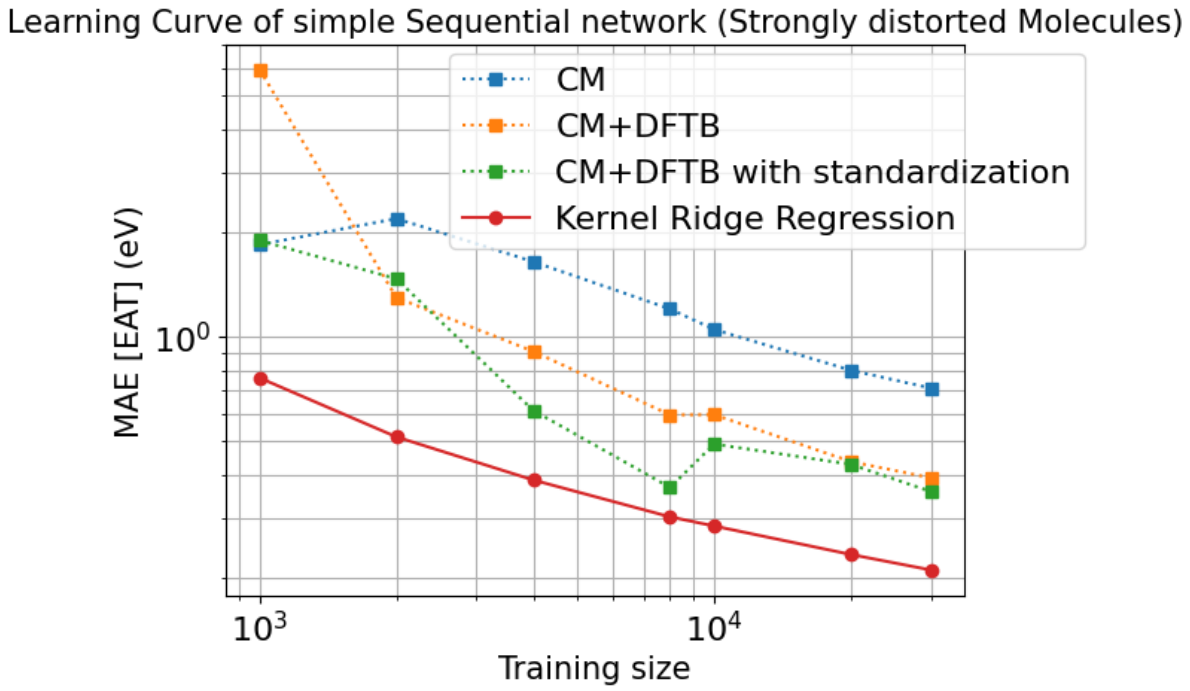


FIGURE 3.11: Comparison of the 2 sequential NN architectures with the results obtained from kernel ridge regression. All the learning curves make use of Coulomb Matrix as the geometric descriptor.

The results for the predictions of E_{at} and E_{gap} while using BOB, and SLATM as the geometric descriptor are shown in the Figures 3.9 and 3.10, respectively. When comparing the results of the sequential NN architectures with those corresponding to KRR method, we find the learning curve as given in Figure 3.11. The kernel was computed using the same input vector as used by the sequential architecture 2, i.e. by concatenating coulomb matrix and the set of 11 DFTB properties together in a 1D vector. The σ and λ parameters for KRR were optimized using PCA. As visible from the learning curve, the kernel method is able to outperform all the NN models, and the learning curve follows a linear trend, showing the presence of overfitting in the NN architectures.

3.2.2.1 SchNet Results

After the comparison of the deep learning models implemented, we compare our results with a benchmark state-of-the-art SchNet implementation. As discussed in section 2.4, SchNet is a continuous filter neural network implementation which makes use interaction blocks to compute the atomic interactions in a molecule. The comparison of MAE of all the models is shown in the Table 3.1 The overall comparison of the three geometric descriptors: CM, BOB, and SLATM for prediction of atomization energy with the sequential architecture 2 is given in Figure 3.12 for equilibrium molecules.

Prediction method	MAE[EAT](eV)
KRR using CM and DFTB	0.211
Arch 1 using CM	0.7114
Arch 1 using BOB	0.642
Arch 1 using SLATM	0.4467
Arch 2 with CM	0.357
Arch 2 with BOB	0.302
Arch 2 with SLATM	0.0498
Arch 3 with CM	0.4132
SchNet with 3 interaction blocks	0.2657
SchNet with 1 interaction block	0.3046

TABLE 3.1: Mean Absolute Error for prediction of Atomization energy using the different Neural Network approaches compared to state-of-the-art model of SchNet and Kernel Ridge Regression methods for strongly distorted molecules.

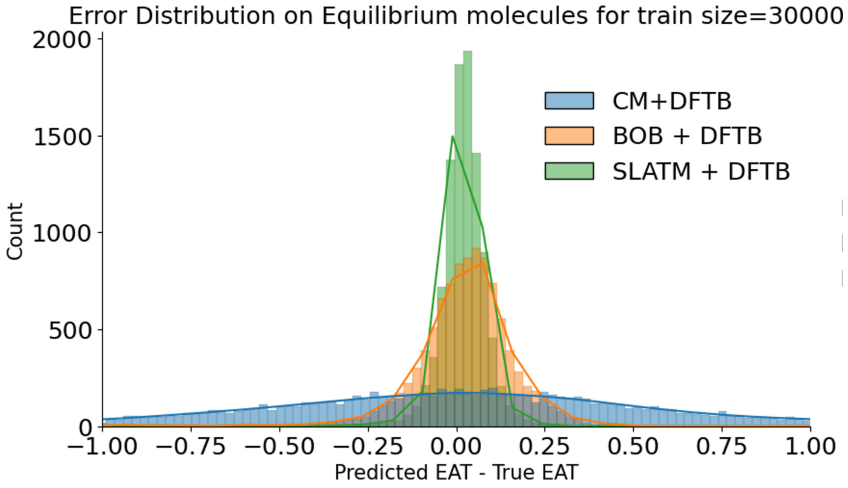
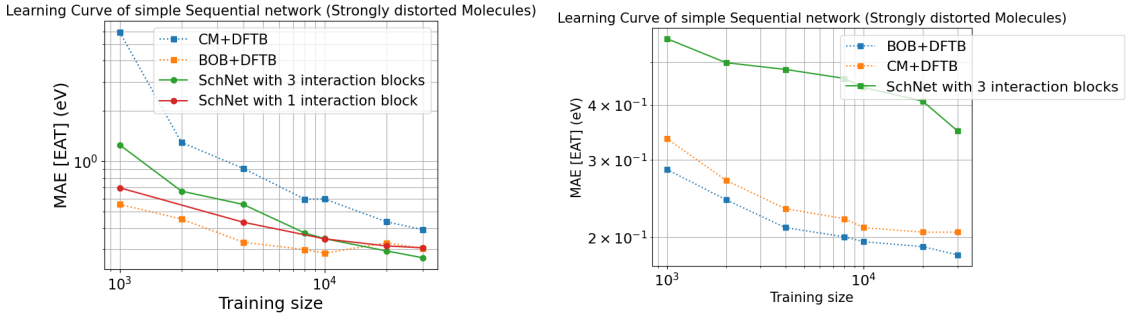


FIGURE 3.12: Comparison of the 3 geometric descriptors for the prediction of atomization energy with the 2nd Sequential architecture.

The learning curves for the comparison is given in Figure 3.13 Figure 3.13 shows that for the prediction of atomization energy, SchNet outperforms the CM+DFTB approach with both one and three interaction blocks. Whereas bag-of-bonds representation is able achieve results comparable to SchNet with a single interaction block. This shows the importance of the interaction blocks implemented in SchNet. These interaction blocks are able to capture the geometric information about the molecules which is of crucial importance for the prediction of extensive properties. On the other hand, for the prediction results of HOMO-LUMO gap, both CM and BOB representations in combination with the electronic descriptors produced better results than SchNet with 3 interaction blocks. These results again show that, for prediction of intensive property like E_{gap}, the electronic descriptors are sufficient and the interaction block approach of SchNet is not needed.



(A) Comparison of the sequential architectures with SchNet for Atomization Energy prediction

(B) Comparison of Sequential architectures with SchNet for HOMO-LUMO gap prediction

FIGURE 3.13: Learning Curves for comparing the prediction results of deep learning models developed with the results obtained from SchNet.

3.3 Feature Reduction

From the results discussed above, we have seen that addition of electronic descriptors allows the neural network to better capture the intensive and extensive properties of the dataset. The set of electronic properties used in training contains a total of 11 properties, of which 2 are multidimensional, which produce a total size of 40 components for the electronic descriptors. The geometric descriptors, on the other hand, have higher dimensionality (276 for CM, 528 for BOB, and 17895 for SLATM). So, electronic descriptor space of size 40 is able to reduce the testing error by a factor of 0.5 in comparison to high-dimensional geometric descriptors.

For efficient training of any NN model, it is important to keep the most relevant features in the input and discard the irrelevant ones, as they will not only slow down the computation, but will increase the training and testing error by moving the model away from the minima. Any machine learning model will use information from irrelevant variables for new data, leading to poor generalization [54]. So far, all the 11 DFTB properties were used for training, and to improve the performance, the most relevant properties will be chosen. Initially, a low-level feature correlation within the properties was computed. A heatmap of feature correlation is shown in the Figure 3.14, where the last row represents the target property, which is atomization energy for this case. This Figure is the graphical representation of correlation matrix computed in python. We can observe that there is a strong negative correlation of atomization energy with properties like Repulsive Energy and Many-Body dispersion energy. "Good features" are defined as those which have high correlation with the target variable, yet low-correlation with each other. This is because of the idea, that if two variables have strong correlation with each other, then they would essentially provide similar information to the model, and having both of them would be redundant.

To get a deeper idea about the correlation, we compute the correlation score of features using ANOVA (ANalysis Of VAriance) F-test [55] values. The formula for the F-test statistic is given

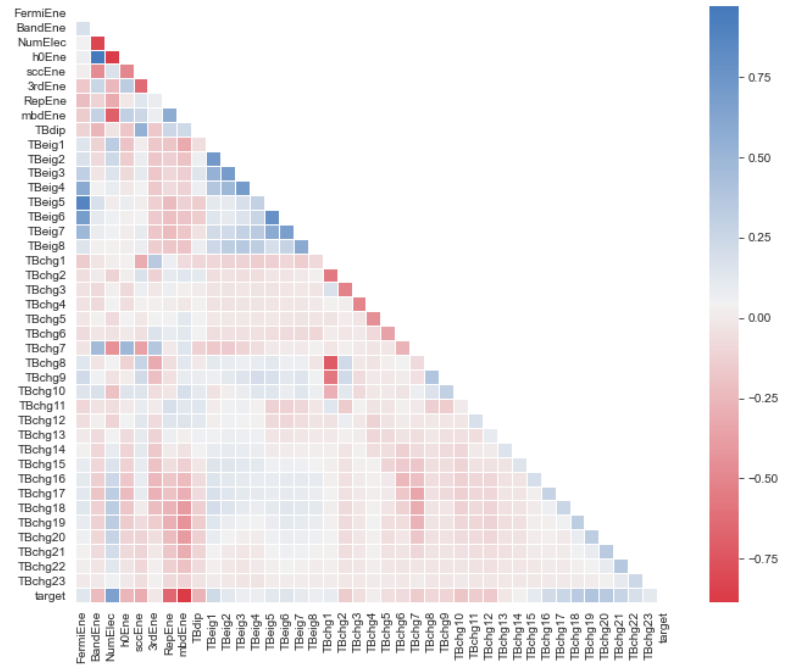


FIGURE 3.14: Heatmap for correlation of various semi-empirical properties with each other, and with the target property, i.e. Atomization Energy.

as

$$F = \frac{ExplainedVariance}{UnexplainedVariance}$$

$$ExplainedVariance = \sum_{i=1}^{K} n_i(\bar{Y}_i - \bar{Y})^2/(K-1)$$

Here, $\bar{Y}_i$ is the sample mean for the i^{th} feature, n_i is the number of observations in the i^{th} feature. $\bar{Y}$ is the overall mean of the data, K is the total number of features.

$$UnexplainedVariance = \sum_{i=1}^{K}\sum_{j=1}^{n_i}(Y_{ij} - \bar{Y}_i)^2/(N-K)$$

Y_{ij} is the j^{th} observation of the i_{th} group out of the K groups, and N is the overall sample size.

Using the F-value as a function to find the best features correlated with the Atomization energy using python library sklearn [56], we found the following scores given to each feature, which is given in the Table 3.2.

This analysis provides the least score to Dipole Moment, and Fermi Energy, and the maximum score to many-body dispersion energy. The techniques for feature reduction discussed above

Property	Feature Score
Fermi Energy	1.728
Band Energy	2.039
Number of Electrons	3.92
0th order Energy	2.353
Second order energy	2.37
3rd order	2.002
Repulsive Energy	4.796
Many-body Dispersion Energy	8.813
Tight-binding Dipole Moment	1.561

TABLE 3.2: Feature Scores given to different electronic properties based on the f-value classifier.

use ranking of features based on correlation with the target property. The methods using such techniques are called filter methods. These methods are simple to use, and do not require much computational resources. These methods work as good low-level feature detection algorithms, using which we can discard the features with really low scores.

Apart from filter methods, another set of methods used fore feature reduction are called *wrapper methods*. These methods use a predictor as a black box and the predictor performance is used to evaluate the features. An example of this approach is Recursive Feature Elimination (RFE). In this technique, recursively features are eliminated, and based on reduced input space, neural network models are trained and the testing error is checked.

For some preliminary results, RFE was used with keras neural network model as a black box predictor, using only 4000 molecules and training them for only 4000 epochs. When this RFE method was run on the dataset of distorted molecules, the results were removal of only 1 property, that is the dipole moment. These results are coherent with the results obtained from the filter methods, but an accurate feature selection will require a better predictor, trained for more number of epochs than the current one.

Chapter 4

Summary and Outlook

4.1 Synopsis

In the present master thesis, we have generated efficient molecular representations by combining geometric and electronic descriptors to predict extensive/intensive physicochemical properties of small organic molecules. The focus here was to use low-cost geometric descriptors and to improve the accuracy in property prediction by adding a set of electronic properties obtained from a well-established semi-empirical method such as density functional tight-binding (DFTB).

The Coulomb Matrix, Bag-of-Bonds, and SLATM geometric descriptors have been used along with DFTB electronic properties to predict PBE0 atomization energy and HOMO-LUMO gap of small molecules considered in QM7-X dataset. In order to test the molecular representations and get good property prediction accuracy, multiple neural network architectures were generated and trained. Due to the availability of large amount of training data in recent chemical databases, it was possible to train efficient neural network architectures on varying training sizes. The training and testing were done on three subsets of the QM7-X dataset, including strongly distorted, slightly distorted, and equilibrium molecular structures. Initially, all the analyses were done on the subset of strongly distorted molecules, and then the results were compared with the other subsets. When comparing the molecular representations, it was observed that for the intensive properties like HOMO-LUMO gap, electronic properties were able to achieve better prediction results as compared to the geometric descriptors. Whereas for the extensive properties, the addition of electronic and geometric descriptors was able to reduce the prediction error as compared to the use of only the geometric descriptors. The best results were obtained using the SLATM geometric descriptor, which is the most costly of the three descriptors used. On comparing geometric descriptors with smaller dimension (Coulomb Matrix, and Bag-of-Bonds), it was observed that BoB provided much better results with only slightly more increase in the training time.

These results for extensive properties were not able to outperform state-of-the-art methods like SchNet and Kernel Ridge Regression, whereas for the intensive property HOMO-LUMO gap, using only the DFTB properties provided better results than SchNet with three interaction blocks. The learning curve for KRR was linear, as compared to the results from the neural networks indicating the presence of overfitting in the NN architectures, because of the inefficiency in the hyperparameters. The effects of outliers proved to be quite small for HOMO-LUMO gap, as compared to the extensive property atomization energy. In order to improve the hyperparameters, hyperparameter tuning was done to know the efficient number of neurons, activation functions, and the batch sizes using keras-tuner [57], and talos [58]. It was observed that smaller architectures with lower number of neurons were able to provide good results compared to bigger architectures. This observation is explained by an interpretation of the Occam's razor, which states that given all other things being equal, the best accuracy will be achieved by a model with the lowest trainable parameters. In other words, if a model's complexity is increased beyond need, the performance will decrease.

Different feature selection techniques were used to identify the most relevant electronic properties suitable for the task at hand. First, filter methods were used to get a low-level understanding and analysis of the properties being used, then some preliminary results were obtained by using wrapper methods like recursive feature elimination. The aim of this technique was to remove the dependent variables and thereby reducing the amount of input data, leading to an improvement in the performance of the model.

4.2 Perspective and future work

A major limitation in the current work is the existence of overfitting in the neural network models reported, which acts as a barrier to efficient neural network training. Practical feature reduction techniques need to be implemented, obtaining the best molecular representation required for prediction of atomization energy and HOMO-LUMO gap. Using the effective representation, a more exhaustive hyperparameter tuning can be done, on not just the number of neurons, but also the batch-sizes and activation functions.

Moreover, as a part of future work, we propose the analysis of local and global electronic descriptors, and splitting them separately for training of the neural networks. Another possible aspect is the use of neural network geometric representations together with electronic properties to develop machine learning force fields. This can also make use of better loss functions involving both energy and forces similar to the one shown by [44].

As we have seen, the output of machine learning models depend a lot on the input data. In this project only small organic molecules were considered (up to 23 atoms), however, it is important

to be able to handle larger molecular systems for many real-life applications. Availability of better data is a crucial aspect for improving the scalability of machine learning models for molecular property prediction.

Bibliography

(1) Keith, J. A.; Vassilev-Galindo, V.; Cheng, B.; Chmiela, S.; Gastegger, M.; Müller, K.-R.; Tkatchenko, A. *Chemical Reviews* **2021**, *121*, 9816–9872.

(2) Von Lilienfeld, O. A.; Burke, K. *Nature Communications* **2020**, *11*, 4895.

(3) Tkatchenko, A. *Nature Communications* **2020**, *11*, 4125.

(4) Von Lilienfeld, O. A.; Burke, K. *Nature Communications* **2020**.

(5) Vamathevan, J.; Clark, D.; Czodrowski, P.; Dunham, I.; Ferran, E.; Lee, G.; Li, B.; Madabhushi, A.; Shah, P.; Spitzer, M.; Zhao, S. **2019**.

(6) Unke, O. T.; Chmiela, S.; Sauceda, H. E.; Gastegger, M.; Poltavsky, I.; Schütt, K. T.; Tkatchenko, A.; Müller, K.-R. *ACS Publications* **2021**.

(7) Chmiela, S.; Sauceda, H. E.; Müller, K.-R.; Tkatchenko, A. *Nature Communications* **2018**, *9*, 3887.

(8) Chmiela, S.; Sauceda, H. E.; Poltavsky, I.; Müller, K.-R.; Tkatchenko, A. *Computer Physics Communications* **2019**, *240*, 38–45.

(9) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. *Scientific Data* **2014**, *1*, 140022.

(10) Smith, J. S.; Isayev, O.; Roitberg, A. E. *Scientific Data* **2017**, *4*, 170193.

(11) Hoja, J.; Medrano Sandonas, L.; Ernst, B. G.; Vazquez-Mayagoitia, A.; DiStasio Jr., R. A.; Tkatchenko, A. *Scientific Data* **2021**, *8*, 43.

(12) Montavon, G.; Rupp, M.; Gobre, V.; Vazquez-Mayagoitia, A.; Hansen, K.; Tkatchenko, A.; MÃ$\frac{1}{4}$ller, K.-R.; von Lilienfeld, O. A. *New Journal of Physics* **2013**, *15*, 095003.

(13) Hansen, K.; Montavon, G.; Biegler, F.; Fazli, S.; Rupp, M.; Scheffler, M.; von Lilienfeld, O. A.; Tkatchenko, A.; Müller, K.-R. *Journal of Chemical Theory and Computation* **2013**, *9*, 3404–3419.

(14) Rupp, M.; Tkatchenko, A.; Müller, K.-R.; von Lilienfeld, O. A. *Phys. Rev. Lett.* **2012**, *108*, 058301.

(15) Blum, L. C.; Reymond, J.-L. *Journal of the American Chemical Society* **2009**, *131*, 8732–8733.

(16) Ramakrishnan, R.; Hartmann, M.; Tapavicza, E.; von Lilienfeld, O. A. *Journal of Chemical Physics* **2015**, *143*, 84111–84111.

(17) Ruddigkeit, L.; van Deursen, R.; Blum, L. C.; Reymond, J.-L. *Journal of Chemical Information and Modeling* **2012**, *52*, 2864–2875.

(18) Ramakrishnan, R.; Dral, P. O.; Rupp, M.; von Lilienfeld, O. A. *Scientific Data* **2014**, *1*.

(19) Schneider, G. *Nature Reviews Drug Discovery* **2010**, *9*, 273–276.

(20) Todeschini, R.; Consonni, V., *Handbook of molecular descriptors*; Todeschini, R, Consonni, V., Eds.; Methods and Principles in Medicinal Chemistry; Wiley-VCH Verlag: Weinheim, Germany, 2000.

(21) Weininger, D. *Journal of Chemical Information and Computer Sciences* **1988**, *28*, 31–36.

(22) Hansen, K.; Biegler, F.; Ramakrishnan, R.; Pronobis, W.; von Lilienfeld, O. A.; Müller, K.-R.; Tkatchenko, A. *The Journal of Physical Chemistry Letters* **2015**, *6*, 2326–2331.

(23) Huang, B.; von Lilienfeld, O. A. *Nature Chemistry* **2020**, *12*, 945–951.

(24) Puleva, M. Reliability of Machine Learning Models for Molecular Property Prediction, MA thesis, University of Luxembourg, 2021.

(25) Pronobis, W.; Schütt, K. T.; Tkatchenko, A.; Müller, K.-R. *The European Physical Journal B* **2018**, *91*, 178.

(26) Porezag, D.; Frauenheim, T.; Köhler, T.; Seifert, G.; Kaschner, R. *Physical Review B* **1995**, *51*, 12947–12957.

(27) Hohenberg, P.; Kohn, W. *Physical Review* **1964**, *136*, B864–B871.

(28) Kohn, W.; Sham, L. J. *Phys. Rev.* **1965**, *140*, A1133–A1138.

(29) Spiegelman, F.; Tarrat, N.; Cuny, J.; Dontot, L.; Posenitskiy, E.; MartÃ, C.; Simon, A.; Rapacioli, M. *Advances in Physics: X* **2020**, *5*, PMID: 33154977, 1710252.

(30) Koskinen, P.; MÃ¤kinen, V. *Computational Materials Science* **2009**, *47*, 237–253.

(31) Fonseca Guerra, C.; Snijders, J. G.; te Velde, G.; Baerends, E. J. *Theoretical Chemistry Accounts* **1998**, *99*, 391–403.

(32) Frisch, M. J. et al. Gaussian09 Revision E.01, Gaussian Inc. Wallingford CT 2009.

(33) Calaminici, P.; Janetzko, F.; Köster, A. M.; Mejia-Olvera, R.; Zuniga-Gutierrez, B. *Journal of Chemical Physics* **2007**, *126*, 44108–44108.

(34) Hourahine, B. et al. *The Journal of Chemical Physics* **2020**, *152*, 124101.

(35) Gaus, M.; Lu, X.; Elstner, M.; Cui, Q. *Journal of Chemical Theory and Computation* **2014**, *10*, 1518–1537.

(36) Gallarati, S.; Fabregat, R.; Laplaza, R.; Bhattacharjee, S.; Wodrich, M. D.; Corminboeuf, C. *Chemical Science* **2021**, *12*, 6879–6889.

(37) Axilrod, B. M.; Teller, E. *Journal of Chemical Physics* **1943**, *11*, 299–300.

(38) Jordan, J. Setting the learning rate of your neural network. 2020.

(39) Commons, W. K-fold cross validation EN `https://commons.wikimedia.org/w/index.php?curid=82298768`.

(40) Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. *CoRR* **2012**, *abs/1207.0580*.

(41) LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D. *Neural Computation* **1989**, *1*, 541–551.

(42) Goodfellow, I.; Bengio, Y.; Courville, A., *Deep Learning*, `http://www.deeplearningbook.org`; MIT Press: 2016.

(43) Schütt, K. T.; Kessel, P.; Gastegger, M.; Nicoli, K. A.; Tkatchenko, A.; Müller, K.-R. *Journal of Chemical Theory and Computation* **2019**, *15*, 448–455.

(44) Schütt, K. T.; Kindermans, P.-J.; Sauceda, H. E.; Chmiela, S.; Tkatchenko, A.; Müller, K.-R. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions, 2017.

(45) Schütt, K. T.; Arbabzadah, F.; Chmiela, S.; Müller, K. R.; Tkatchenko, A. *Nature Communications* **2017**, *8*, DOI: `10.1038/ncomms13890`.

(46) Humphrey, W.; Dalke, A.; Schulten, K. *Journal of Molecular Graphics* **1996**, *14*, 33–38.

(47) Chollet, F. et al. Keras, `https://keras.io`, 2015.

(48) Christensen, A.; Faber, F.; Huang, B; Bratholm, L.; Tkatchenko, A; Muller, K.; von Lilienfeld, O. QML: A Python Toolkit for Quantum Machine Learning.

(49) Harris, C. R. et al. *Nature* **2020**, *585*, 357–362.

(50) Kingma, D. P.; Ba, J. Adam: A Method for Stochastic Optimization, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015, 2014.

(51) He, K.; Zhang, X.; Ren, S.; Sun, J. In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp 1026–1034.

(52) Clevert, D.-A.; Unterthiner, T.; Hochreiter, S. *arXiv preprint arXiv:1511.07289* **2015**.

(53) Vidal, R.; Ma, Y.; Sastry, S. S. **2016**, 25–62.

(54) Chandrashekar, G.; Sahin, F. *Computers Electrical Engineering* **2014**, *40*, 40th-year commemorative issue, 16–28.

(55) Lomax, R. G., *An Introduction to Statistical Concepts*, 2nd ed.; Routledge Academic: New York, NY, 2007.

(56) Pedregosa, F. et al. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.

(57) O'Malley, T.; Bursztein, E.; Long, J.; Chollet, F.; Jin, H.; Invernizzi, L., et al. KerasTuner, `https://github.com/keras-team/keras-tuner`, 2019.

(58) Autonomio Talos [Computer software], 2020.