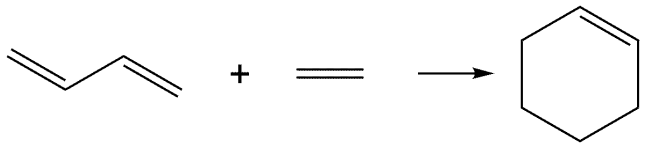
Chapter 1

Introduction

1.1 Preface

Scientists share chemical information through the language of molecules. On the news we hear about the impact of CO_2 on the climate, we look at the back of the ibuprofen container and read that it has a $C_{13}H_{18}O_2$ composition, in organic chemistry we learn that the formation of cyclohexene (C_6H_{10}) from 1,3-butadiene (C_4H_6) and ethylene (C_2H_4) occurs via the Diels-Alder reaction,



and in physical chemistry we discover the molecular orbitals of benzene (C_6H_6) and naphthalene ($C_{10}H_8$)–which explain their unique properties. In these examples, the information about these molecules is passed to the receiver with the use of molecular formulas, structural formulas, or even by introducing aspects of molecular electronic structure.

The digital revolution (or third industrial revolution) introduced automation and digitization in many aspects of the sciences, manufacturing, and daily life through the use of electronics, computers, and the internet. We are now living in the fourth industrial revolution which is dominated by personal connected devices, artificial intelligence technologies, data analytics, and digital transformations. Those require efficient communication between different participants and online objects. Like every aspect of daily life, communication, manufacturing, and sciences, the field of chemistry is currently experiencing a transformation by the rise of digitalization and the era of Big Data. However, the common language used for teaching chemistry or sharing results and announcing discoveries is not adequate for the flow of chemical information between a user and a digital devise. This becomes more evident when such communication involves

databases of billions or even trillions of molecular entries.

Effective methodologies for transferring chemical information between humans and machines have been developed since the early days of chemoinformatics (1970s). The field of molecular representations has been further developed in previous years to digitalize molecular information in order to apply tools from artificial intelligence and machine learning to chemical exploration. In this book, we present the most common "languages" used by scientists to communicate chemical information with computers and algorithms. These methodologies have allowed the systematic exploration of chemical space that goes beyond traditional experimental or computational approaches. Thus, we are now in a unique position to explore large chemical databases for the extraction of patterns (e.g. structure-functionality relations) and to discover molecules and materials with enhanced properties.

The first chapter includes a historical background on how philosophers and alchemists represented mater, and how the Chemical Revolution (1770 – 1790) lead to the standard chemical nomenclature that is used till today. We move then to the motivation behind the preparation of a book that introduces molecular representations to chemistry students. The first chapter ends with a short discussion on important properties that molecular representations should have.

1.2 Historical Background

A long standing question, for scientists and alchemists alike, is how to represent molecules. Since atoms are the building blocks of molecular structures, we need to go back to Ancient Greece and to the origins of atom theory to find answers to this question.

1.2.1 The Atom Theory

Leucippus (480 – 420 BC) and his disciple, Democritus (460 – 370 BC), introduced the hypothesis that the universe is composed by atoms and voids. Democritus proposed a basic theory for atoms in which all apparent changes in matter result from changes in the groupings of atoms. He suggested that they can be distinguished from each other by their properties including weight, form, size, position, and arrangement. Asclepiades of Prusa (124 – 40 BC) hypothesized that atoms interact with each other and form clusters. In parallel to the atom theory, Empedocles (490 – 430 BC) introduced the four "fundamental elements" of fire, earth, air, and water, and a fifth element ("aether") was later added to complete the early interpretation of the matter that is around us.

The Roman philosopher and poet Lucretius (99 – 55 BC) introduced the ideas of Asclepiades to Empedocles' theory and argued that the four elements are connected to each other with hooks and sockets, which is an early view of chemical bonding. During the dark medieval times, alchemists acquired the element theory, and the ideas of Democritus were abandoned, especially since the influential philosophers Aristotle and Plato rejected the atom theory based

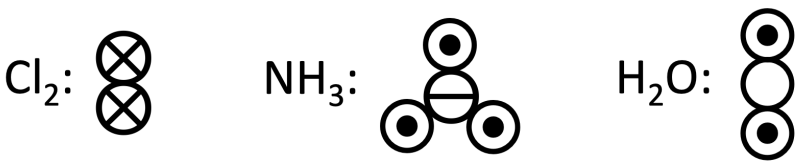
on philosophical grounds. It took almost 2000 years to embrace Democritus ideas with scientific observations and reasoning.

During the 17^{th} century, the atom theory was revived primarily by the work of Pierre Gassendi (1592 - 1655) and Isaac Newton (1643 - 1727), who acknowledged that particles are connected with hooks and that they attract each other via forces which, in close proximity, perform chemical operations. A more concrete concept was given in 1661 by Robert Boyle (1627 – 1691), who showed that the four elements were not fundamental elements as they could be broken down into simpler particles. He also stated that matter is composed of clusters of particles and chemical operations are performed by their rearrangements. Étienne François Geoffroy (1672 - 1731) proposed the theory of chemical affinity to explain the forces between particles of various salts, and introduced the Affinity Table, a visual representation of alchemical elements organized based on "displacement reactions", an early view of chemical reactivity.

In 1803, John Dalton (1766 - 1844) provided the foundations of modern atomic theory based on experimental observations. Among other justified scientific principles, he stated that each elemental atom is unique, he rationalized that the weight of atom determines its character, he defined the atomic weight of the lightest element (hydrogen) as unity, and that chemical reactions will occur based on atom-to-atom ratios. He realized that the old alchemical symbols could not fit his new theory, so he introduced a new set of elemental symbols. Based on this notation, he represented molecules as atom combinations. For example, water ⊙○ is represented as the combination of oxygen ○ and hydrogen ⊙. A few years later, Jöns Berzelius (1779 - 1848) argued that is easier to memorize letters than symbols, so he introduced a simplified atom notation based on the first one or two letters of the element's name, which is the notation we still use today. For example, oxygen (or "Oxygenium") was represented by the letter "O", hydrogen (or "Hydrogenium") by the letter "H", aluminum by "Al", and iron ("Ferrum") by "Fe". Of course, this element terminology has been the basis for the molecular nomenclature.

1.2.2 How to Represent a Molecule?

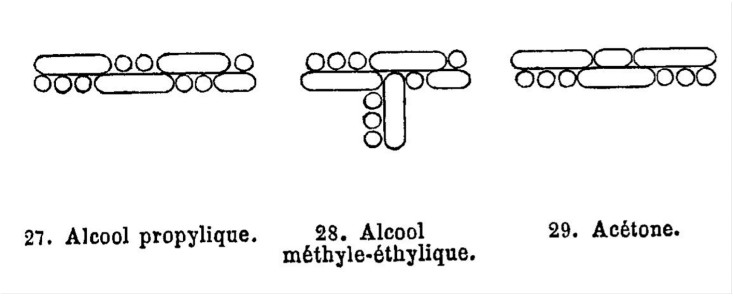
The concept of "molecules" was first proposed by Amedeo Avogadro (1776 – 1856) based on Dalton's laws of atom proportions. In 1833, Marc Antoine Auguste Gaudin (1804 - 1880) introduced "volume diagrams" which represented molecules based on correct molecular formulas. For example, Cl_2, NH_3, and H_2O were represented as:



However, water was given as a linear molecule ...

A few years later, Friedrich August Kekulé (1829 – 1896) developed a theory for explaining how organic molecules are formed. He proposed that carbon is tetravalent and can form bonds with other carbons as well. He also represented simple organic molecules with "sausage models".

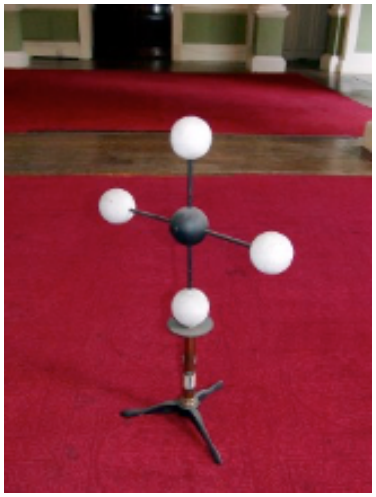
Figure 1.1: Kekulé's "sausage" representation of molecules. From left to right: 1-propanol, 2-propanol, and acetone.



Inspired by the work of Kekulé, the Scottish chemist Archibald Couper (1831 – 1892) developed the idea of molecular structures where bonds are represented with straight lines between circles (atoms), and Alexander Crum Brown (1838 - 1922) was the first who introduced the atom label inside the circle. These are the well-known structural formulas that are still used extensively today. In 1861 a high-school teacher Johan Josef Loschmidt (1821 – 1895) self-published a booklet titled *Chemische Studien*, in which he included two-dimensional molecular representations. He introduced double lines to denote double bonds and triple lines for triple bonds, which are easily recognizable to modern-day chemists.

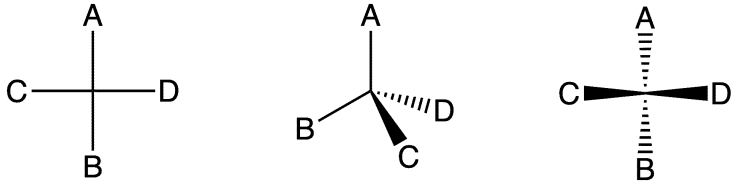
These ideas were further explored by the German chemist August Wilhelm von Hofmann (1818 – 1892), who was the first to create three-dimensional stick-and-ball molecular models for his lectures (Figure 1.2).

Figure 1.2: Physical representation of methane introduced by Hofmann.



Surprisingly, Hofmann's color scheme is still in use today, where, for example, oxygen is represented with a red sphere, nitrogen with a blue sphere, and hydrogen with a white sphere. However, Hofmann's spheres did not reflect the correct proportions of atoms and he also represented carbon as planar. The latter was considered by Emil Fischer (1852 – 1919), who introduced a projection technique for representing three-dimensional molecules in two-dimensions. An example of tetravalent carbon is shown in Figure 1.3 for a planar, tetrahedral, and Fischer projection representation.

Figure 1.3: From the erroneous planar tetravalent carbon (left) to a tetrahedral geometry (center) and from there to the Fischer projection (right).



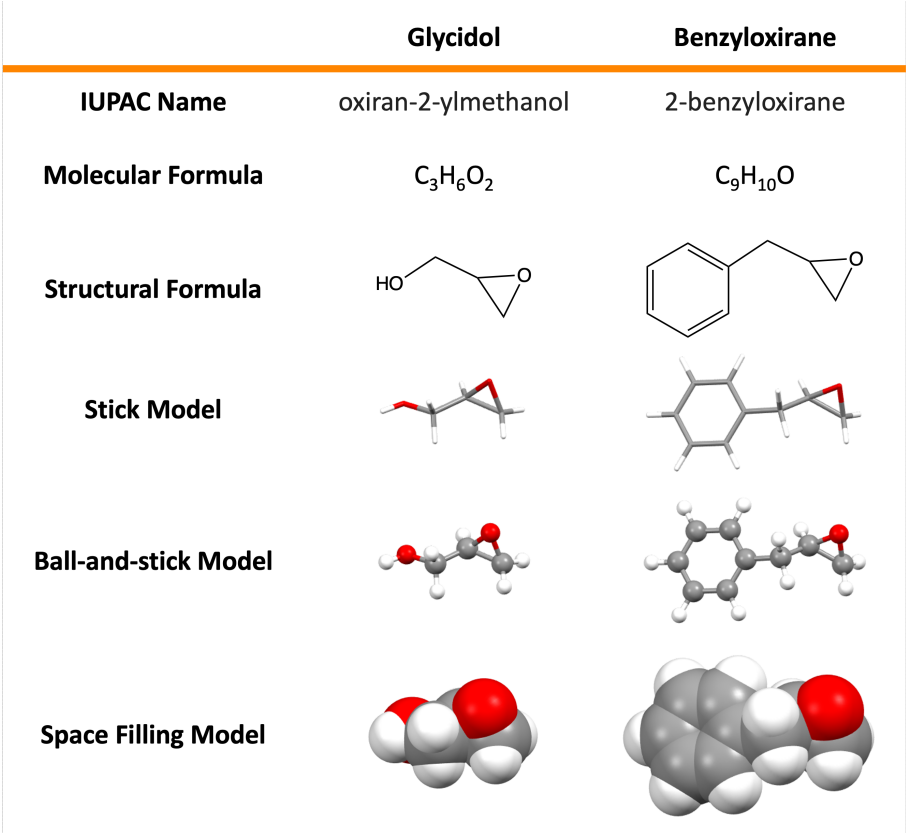
In the early 20$^{\text{th}}$ century, the American chemist Gilbert N. Lewis (1875 – 1946) introduced dots as a representation of electrons around atoms, a concept that he later generalized to represent the covalent, electron-pair bond as two "dots" or a straight line. In 1927, the physicists Fritz London (1900 – 1954) and Walter Heitler (1904 – 1981) applied the emerging theory of quantum mechanics to explain the chemical bond of the hydrogen molecule. Their important work influenced Linus Pauling (1901 – 1994) who further applied quantum mechanics to calculate geometries and properties of molecules, as well as provide the foundations of valence-bond theory. Ultimately, the molecular wave function, the

function that holds all the pertinent information with respect to the electron and nuclear positions, can be considered as the true, most complete molecular representation since it includes all intrinsic information of a molecule.

1.3 Motivation

It is evident that conventional chemical nomenclature, molecular or structural formulas and 3D molecular models (Figure 1.4), that are often used in a lecture hall, in chemistry textbooks, or in scientific publications are difficult to input into a computer program in an effective manner such that the algorithm can obtain useful chemical information. For example, the introduction of a molecular formula to a computer is straightforward since it is composed of letters and numbers. It provides the atomic composition of a molecule, but it does not distinguish between different isomers and does not include any information about the molecular structure. The ball-and-stick model gives information related to the isomer in question, its stereochemistry, and its geometry, but since it is a graphical representation ("image"), its conversion to a machine-readable form becomes cumbersome. This becomes even more important when we consider chemical data representation for machine learning applications, where the target is to describe a molecular structure in a format that can maximize the ability of the model to effectively "learn" the underlying chemical properties and functionalities when it is exposed to hundred of thousands or even millions of molecules. Thus, it becomes evident that an efficient "digitalization" or "encoding" of chemical information is vital for the successful training of data-driven models, especially when a small number of data points are available or when transferability between different families of molecular structures is desired.

Figure 1.4: Glycidol and benzyloxarine are two molecules that are used as examples throughout this book. Their IUPAC name and (condensed) molecular formula are provided with a few common three-dimensional representations.



When we think of molecules, we often think of chemical formulas such as H_2O for water or CO_2 for carbon dioxide. These chemical formulas are based on the invention of *line representations* in the 1860s pioneered by A. Kekulé, H. Debus, H. L. Buff, E. Erlenmeyer, E. Frankland, and B. F. Duppa.[1] In the twentieth century, with the rise of computers, chemist sought novel ways to apply line representations for the exploration of chemical properties. One of the first broadly applied line representations was the *Wiswesser Line-Formula Notation* (WLN), which was introduced in 1949 as a concise line representation that seeks to closely mimic molecular diagrams.[2] While WLN has limited use nowadays, fingerprinting methods that soon followed after, such as *Morgan fingerprints*[3], are commonly applied. The success of Morgan fingerprints lie in their algorithm for sorting connectivity tables, which was introduced to solve the molecular isomorphism problem related to the permutational invariance of atom ordering in molecules.[3] More modern representations, like the *Simplified Molecular Input Line Entry System* (SMILES) discussed in Section 2.7.1,

build on features found in WLN. These features include the use single letter symbols for common atomic groups and numerical notation for alkyl chains.[4] Overall, line notations are compact, less costly than two- or three-dimensional representations,[5] but as we will see in this and in the following chapters, they lack important geometrical, conformational and electronic structural information.

The aim of this book is to provide a non-expert with a comprehensive introduction to molecular representations that are used in the field of chemoinformatics and machine learning applications in chemistry. Our philosophy is to present each representation's theoretical foundations and the logical steps that lead to their development and applicability, together with representative examples and Python code snippets. Since (molecular) graph theory has been the basis of numerous representations that are used nowadays extensively in chemical machine learning and data sciences, we begin this book with the fundamentals of graph theory and the description of methodologies that are directly connected to molecular graphs (Chapter 2).

One of the main driving forces for the development of novel molecular representations has been the field of chemoinformatics. According to Engel and Gasteiger, chemoinformatics is defined as "the application of informatics methods to solve chemical problems". One of the main targets behind Chemoinformatics is to understand the *a priori* physical, chemical, and biological effects of chemicals and chemical reactions.[6] To facilitate this goal, databases to query these properties such as the Chemical Abstract Service (CAS)[7] and PubChem[8] and software packages such as RDKit[9], Open Babel[10], ChemML[11], and DScribe[12] have been developed. Our intention is to include in this book established and widely applicable methodologies for encoding chemical information together with novel and emerging approaches that have recently captured the attention of the chemical community.

Within this context, many common molecular representations developed within the chemoinformatics community have formed the basis for modern chemical data sciences. Since the majority of these methodologies that are originated from the mature scientific field of chemoinformatics are based on molecular graphs, we have included them in Chapter 2. Some representative examples are of such popular methods are the Simplified Molecular Input Line Entry System (SMILES) and the MACCS fingerprints. The second category included in this book explores molecular topology from an algebraic topology perspective such as persistent homology (Chapter 3). The last category includes the emerging field of molecular vectorizations based on the underlying molecular physics (Chapter 4). Such physics-based or physics-informed representations consider principles from the electronic structure of molecules, such as Coulomb interactions, atom-based functions, or even descriptors from electron correlations.

It is also important to remember that the choice of a particular molecular representation should be in based on the chemical application of interest. As we will see in the next Chapters, each molecular representation has been developed on specific chemical, physical, or biological principles and sometimes, by having a specific property in mind (e.g. molecular similarity, drug-like activity, chemical

reactivity or catalysis, conformational search, etc.).

We should also mention that this eBook does not describe the standard machine learning models that are used extensively, nor their mathematical formulation. We refer the reader to excellent textbooks that can provide such background (see the end of this chapter for a collection of useful resources). In this eBook, we use the term "algorithm" or "learner" to describe any machine learning algorithm that is trained based on data such as neural networks, random forest, kernel-ridge regression, or support-vector machines, while the term "model" includes the molecular representation, and external hyperparameters together with the machine learning algorithm.

The chemical encoding (i.e. molecular representation) and the amount of information that is passed to the learner is vital for the successful development and training of a machine learning model with predictive power. Thus, it has become nowadays a common practice to test a variety of different molecular representations and/or learners when we develop a new model. Measuring the model performance for each encoding/learner combination requires proper model training and hyperparameter search, and contributes to the evaluation of the developed machine learning model.

Finally, and in order to enhance the readability of this eBook, we have selected two molecules, glycidol and 2-benzyloxirane (Figure 1.4), to serve as examples of the most widely used representations that are discussed here. These molecules have been selected as each of them has two distinct molecular functional groups. Both contain an epoxide, a three-member ring, but glycidol has a hydroxo group, whereas benzyloxirane has a phenyl group.

1.4 Properties of Molecular Representations

Before we delve into the different categories of representations, it is important to discuss the pertinent properties a method should have in order to be considered an effective molecular representation. As we will see in the next chapters, those properties are desired but not mandatory for the training and application of a machine learning model. First, let us consider a molecule in Euclidean space. The position of each atom is specified by its Cartesian coordinates, and the list of all atomic coordinates generates a simple and general description of the molecular structure. Usually, the atomic order in the list of coordinates is arbitrary, and permutations of atoms can generate multiple lists representing the same molecule. The same holds when rotations, translations, or reflections are performed on a particular molecular structure, where new lists are generated which describe the same molecule structure. Therefore, Cartesian coordinates are not a suitable format for a molecular representation for many data-driven and/or computational methodologies. A good representation should be *invariant with respect to permutation, rotational, reflection, and translation symmetries*. Other desirable properties of molecular representations are:

- *Uniqueness*: Each molecular structure should be uniquely encoded so that a unique one-to-one mapping exists from the input structure to the output

representation. While this requirement is met for most representations, the inverse will not always be true.

- *Completeness*: The injectivity of a molecular representation should be guaranteed by the molecular representation [13].
- *Descriptive*: A molecular representation should adequately describe the desired molecule with respect to the machine learning application. This is an attractive feature for many chemical applications and can enhance the learning power of a model, as a representation can include electronic structure or topological information of a molecule. Numerous examples are provided in the next Sections.
- *Simple*: Do not forget Occam's razor, simplicity is an art. Cumbersome representations add unneeded complexity and are undesirable when hundreds to millions of molecules are in the dataset. Simple representations offer portability of the representation and of the model, especially when we are in the realm of Big Data.

The uniqueness and completeness of a molecular representation tend to be complementary properties and in general, if you have one you tend to have the other. Whereas for the last two properties, the desire for a simple, yet descriptive representation is often the main challenge when designing new molecular representations. If all four properties are satisfied, the molecular representation will offer a reliable input for the machine learning task at hand.

1.5 That's a Wrap

In this Chapter, we have briefly discussed how chemists represent molecules the evolution of chemical nomenclature together with some fundamentals of molecular representations for machine learning. The rest of the book covers established but also emerging methods that allow the encoding of chemical information in formats that can be used and manipulated by computers and algorithms, while we have prioritize representations that are used extensively in chemical applications of machine learning. We have chosen to organize these methods into three main categories. The first category includes methods that are based on molecular graphs, which have both historical and application importance (Chapter 2). In Chapter 3, we extend the ideas of molecular graphs by introducing representations that aim to capture the topology of molecules or molecular groups. Finally, Chapter 4 covers molecular representations that are based on the physical properties of atoms and molecules and, in particular, in their electronic structure. These methods have become the past years the topic of active research since they have a vital role on the use of artificial intelligence and machine learning for chemical applications. Some excellent, introductory resources on machine learning fundamentals are listed below. The first is a compact collection of topics related to chemical applications, the latter is a more detailed introductory book with hands-on examples.

- Machine Learning in Chemistry, Jon Paul Janet, Heather J. Kulik, ACS In Focus, **2020**.
- Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow, Aurélien Géron, **2019**, O'Reilly.

Chapter 2

Graph-based Representations

2.1 Introduction

Graph-based representations serve as one of the more intuitive molecular representations. One of the first ways we learn to visualize molecules is with atoms as nodes/vertices and bonds as edges between nodes, effectively forming a graph. Graph theory is a branch of mathematics that explores how objects are connected and some of the properties of these objects and connections. The central object in graph theory is a graph, which can be considered as a collection of elements that create a set together with the relations between the elements of the set. A chemical graph is an effective representation of an abstract chemical system, such as a molecule or a complex reaction network. As we will see in this Chapter, a molecular structure is a constructed, organized graph itself! Graph theory has found many applications in chemistry including chemoinformatics, chemical topology, chemical reactivity, group theory, and electronic structure theory. In this Chapter, we will construct molecular graphs and analyze these graphs with tools from graph theory. We will also explore related graph-based concepts such as topological indices, autocorrelation functions, structural keys, the Simplified Molecular Input Line Entry System (SMILES), and the International Chemical Identifier (InChI).

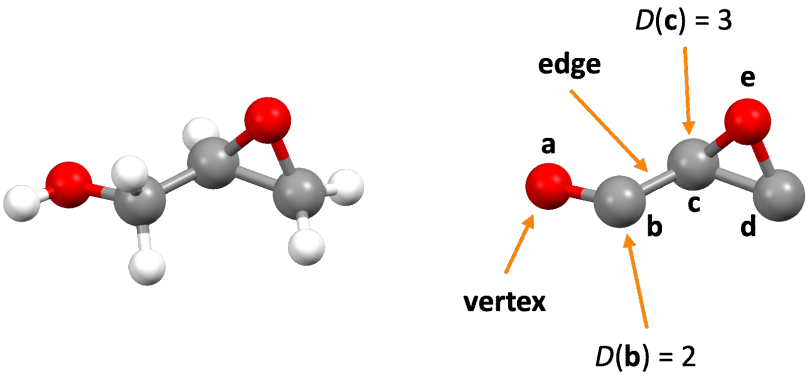
2.2 What is a Molecular Graph

In order to introduce a unified notation, we will now provide some necessary definitions fundamental to graph theory. A graph is an ordered pair, $G = (V, E)$, of a vertex set, $V = \{1, \ldots, n\}$, and an edge set of unordered pairs, $E = \{e_1, \ldots, e_m\}$ where $e_k = v_i v_j = v_j v_i$. Two vertices v_i and v_j are adjacent if they are incident in a common edge e_k. A *labeled graph* is a graph with

labels assigned to the vertices and the edges. *Molecular graphs* are labeled graphs where the vertices are labeled with the type of atom (hydrogen, carbon, nitrogen, etc.) and the edges are the bonds that connect the atoms. We can label the edges with information like the bond order (single, double, or triple) or the bond length. Often, hydrogen atoms are left off of the molecular graph to form the *hydrogen-suppressed molecular graph*, as their presence is implied by the valency of the remaining atoms. Molecular graphs provide information about the elemental composition, connectivity, and bond type of a molecule, and as such have remained a popular format for molecular representations.

To better understand these definitions, we will consider the molecular structure of glycidol ($C_3H_6O_2$, Figure 2.1, left) and we will connect it to a molecular graph (Figure 2.1, right). For sake of simplicity, we consider a hydrogen-suppressed graph. This graph has 5 vertices labeled **a** - **e**, and 5 edges based on the bonding connectivity between the non-hydrogen atoms. The type of atom is given by the color; carbons are grey and oxygens are red. The degree $D(i)$ of a vertex i is the number of edges incident with the vertex i. For example, the carbon labeled as **b** or $C_{\mathbf{b}}$ has $D(\mathbf{b}) = 2$ since it is bonded to two atoms, the oxygen $O_{\mathbf{a}}$ of the hydroxo group and the carbon $C_{\mathbf{c}}$ of the three-member ring (ethylene oxide). Similarly, the degrees of the other vertices are $D(\mathbf{a}) = 1$, $D(\mathbf{c}) = 3$, $D(\mathbf{d}) = 2$, and $D(\mathbf{e}) = 2$.

Figure 2.1: The molecular structure (left) and the molecular graph (right) of glycidol. Note, the color of the vertex denotes the type of atom. White corresponds to hydrogen, grey to carbon, and red to oxygen.

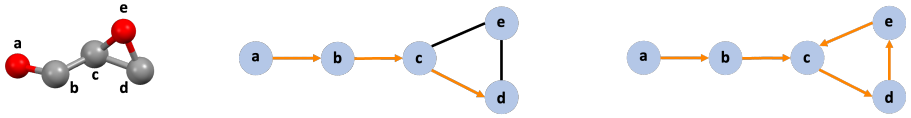


A typical molecular graph is a simplified two-dimensional format that does not hold any geometrical information with respect to bond distances, angles, or dihedrals (see Figure 2.2). Based on this simplified notation, we can provide additional definitions with respect to the properties of the graphs. A walk is a sequence of vertices and edges of a graph. A walk in which neither vertices nor edges are repeated is called a path. If the starting and ending vertices of a walk or path are different, then we will call this as an *open* walk/path. A traversal

is a walk that visits each vertex at least once. Typically, we want to minimize the traversal to cut down on repeated vertices as much as possible.

Two examples of open paths are given in Figure 2.2. The first has an **a** > **b** > **c** > **d** sequence (center) and the second has an **a** > **b** > **c** > **d** > **e** > **c** sequence (right), as which are shown in orange edges. On the contrary, a *closed* walk/path starts and ends on the same vertex. Length is the number of edges included in a walk or path. For the two examples of Figure 2.2, the lengths of the paths are 3 and 5, respectively. *Distance* is the length of the shortest path between two vertices. For example, the distance between vertices **b** and **d** is 2 (**b** > **c** > **d**). A *complete graph* is a fully connected graph where the vertex is connected with all the other vertices ($D = n - 1$, where n is the number of vertices). A *cycle* is a graph where each vertex has $D = 2$, e.g. a benzene ring is a cycle graph. Further categorization of graphs include *connected/disconnected* graphs and *simple*, *multigraph*, and *general* graphs.

Figure 2.2: The molecular graph of glycidol (left). Two examples of open paths are given in (center) and (right). The center path has length 3 and the path on the right has length 5.



These features provide insight to the underlying chemical structure. For example, atom adjacency and the graph distance between two atoms both give connectivity representations for molecules. Capturing this information in a way that is easy to parse and feed into computers can be done via matrices, as we will see in Section 2.3.

2.3 Graphs and Matrices

If a graph is adequately labeled, it can be associated with several matrices. Matrices generated from molecular graphs often serve as a useful representation for data science and chemoinformatics applications, since such "vectorized" molecular structures can be a direct input for machine learning models. Two common "graph-theoretical" matrices are the adjacency matrix and the distance matrix, which are often referred to as topological matrices. The word *topology* is used in mathematics to denote geometric properties that are preserved regardless of how you twist, stretch, or bend an object.

2.3.1 Adjacency Matrix

An important matrix representation of a graph G, that has been used in numerous chemical applications, is the *vertex-adjacency matrix* or simply, the

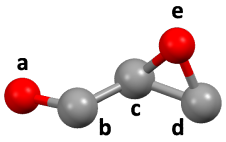
adjacency matrix. The adjacency matrix $\mathbf{A} = \mathbf{A}(G)$ of a graph G, with n vertices, is a square $n \times n$ matrix which contains information about the internal connectivity of the vertices. Its elements are defined as:

$$\mathrm{A}_{ij} = \begin{cases} 1, & \text{if vertices } v_i \text{ and } v_j \text{ are adjacent} \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

$$\mathrm{A}_{ii} = 0 \tag{2.2}$$

Figure 2.3 shows the adjacency matrix of glycidol. In this example, we have followed the same labeling as in Figure 2.2. Since the matrix ordering depends on the vertices labeling and enumeration, the adjacency matrix is not permutationally invariant.

Figure 2.3: The adjacency matrix of glycidol.



$$\mathbf{A}(G) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

The *bond matrix* is an extension of the adjacency matrix that encodes additional molecular information related to the chemical bonds of the molecule. Instead of including the atom connectivity as a binary matrix element, the bond matrix includes the bond order of each connected atom pair. Addition of the free valence electrons of an atom on the diagonal gives rise to the *bond-electron matrix*.

2.3.2 Distance Matrix

The *distance matrix*, $\mathbf{D} = \mathbf{D}(G)$, of a labeled connected graph G, with n vertices, is a square $n \times n$ matrix whose elements are defined as:

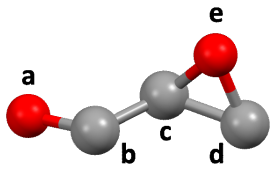
$$\mathrm{D}_{ij} = \begin{cases} d_{ij}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{2.3}$$

where d_{ij} is the distance, such as the topological distance, between two vertices v_i and v_j. The topological distance is measured based on the length of the shortest path (i.e. the minimum number of edges) between these two vertices. As an example, the topological distance matrix of glycidol is shown in Figure 2.4 (left). In graph theory, the conversion of an adjacency matrix into a distance matrix is known as the *all pairs shortest path problem*.

An alternative to the topological distance is the geometric distance (i.e. the atom distance in the Euclidean space). In that case, all matrix elements are

given in distance units (e.g. in Å or pm). Figure 2.4 (right) shows the distance matrix of glycidol with geometric distances.

Figure 2.4: Labeled glycidol molecule (top) with the corresponding distance matrices; topological distances (left) and geometric distances (right).



$$\mathbf{D}(G) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 & 3 \\ 1 & 0 & 1 & 2 & 2 \\ 2 & 1 & 0 & 1 & 1 \\ 3 & 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 1 & 0 \end{bmatrix} \end{array} \qquad \mathbf{D}(G) = \begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 1.422 & 2.376 & 3.764 & 3.549 \\ 1.422 & 0 & 1.491 & 2.610 & 2.462 \\ 2.376 & 1.491 & 0 & 1.508 & 1.437 \\ 3.764 & 2.610 & 1.508 & 0 & 1.431 \\ 3.549 & 2.462 & 1.437 & 1.431 & 0 \end{bmatrix} \end{array}$$

This idea of topologically-invariant features, in other words, features that do not change based on the orientation of the molecule, provide important structural information. In the next Section, we will look at topological indices, a tool used to summarize the connectivity of a molecule.

2.3.3 Weighted Graphs

Weighted graph, a class of graphs commonly applied as input for graph neural networks, include "weights," or numerical values, assigned to nodes and edges. In molecular applications, the nodes of weighted molecular graphs often include the atomic weight, the atomic number (Z), or some other numerical value to represent the atom type and its environment. In machine learning applications, the atomic number Z is often used before being transformed into some unique, "learned" atom embedding. A more specific example is discussed in more detail in Section 4.3.6. The adjacency matrix of a weighted molecular graph differs from the unweighted adjacency matrix since it represents the edges between nodes as weights, instead of using zeroes or ones.

Figure 2.5: Molecular graphs can be modeled as unweighted or weighted graphs, depending on the particular application. For our example molecule, glycidol, we have three toy examples: an unweighted graph (left), a graph with node weights (middle), and a graph with node and edge weights (right). Unweighted nodes are black and weighted nodes are denoted by color. Dashed lines denote an unweighted edge and bold lines denote weighted edges.

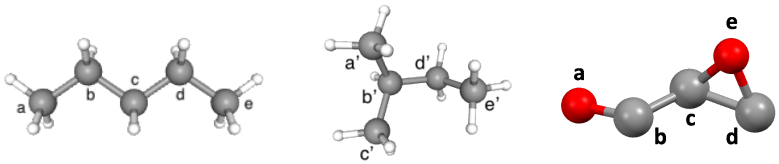
These weighted graphs can replace molecular graphs in molecular descriptors, like the Wiener index, which are discussed next in Section 2.4.

2.4 Topological Indices

Topological indices are *molecular descriptors* based on their molecular graphs and they have been used extensively in the field of chemoinformatics. They can provide insights into the underlying structure of the graph and several of them are graphically invariant. They were pioneered by Harry Wiener, who argued that molecular species can be characterized by such mathematical descriptors as effectively as from other physicochemical properties. Below, we will explore a few different indices, while we will delve into other types of molecular topology in Chapter 3.

Pentane, isopentane, and glycidol will serve as our three examples, pictured in Figure 2.6. However, we should add that such indices are important metrics for structure searching, but they have limited applicability in modern chemical data sciences.

Figure 2.6: Left to Right: Labeled pentane, isopentane, and glycidol.



For the rest of this chapter, we will use the following notation: $\sum_{a \in A} a$ denotes a sum over all elements in a given set A. For example, if $A = \{1, 2, 3, 4\}$, then

$$\sum_{a \in \{1,2,3,4\}} a = 1 + 2 + 3 + 4.$$

2.4.1 The Wiener Index and the Hyper Wiener Index

Introduced in 1947 by Harry Wiener, the *Wiener index* is one of the most common topological indices.[14] Given a molecular graph, the Wiener index is the sum of the shortest paths between each pair of non-hydrogen atoms or in mathematical notation,

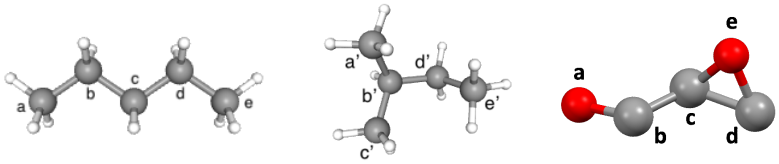
$$\sum_{v,w \in V(G)} d(v, w), \tag{2.4}$$

where v and w run over all possible pairs of vertices in the vertex set and where d denotes the topological distance between v and w. In other words, how many bonds lie between v and w on the shortest path.

Let us consider pentane as an example. The topological distance between carbon C_a (labeled as "a" in the pentane molecule shown in Figure 2.7) and

carbon C_b is 1. Similarly, the topological distances between C_a and the remaining three carbon atoms C_c, C_d and C_e are 2, 3 and 4, respectively. The distances between C_b and atoms C_c, C_d and C_e are 1, 2 and 3, respectively. The distances between C_c and atoms C_d, and C_e are 1, and 2, respectively, while the last distance between C_d, and C_e is 1. Summation of all the pairwise terms gives a Wiener index of 20. By following the same process, we can find that the Wiener indices of isopentane and glycidol are 18 and 17, respectively (Figure 2.7). Even though these molecules consist of the same number of non-hydrogen atoms, their Wiener indices are different, which shows how the different structures are captured by the Wiener index.

Figure 2.7: Top, Left to Right: Labeled pentane, isopentane, and glycidol. Bottom: The Wiener and Hyper-Wiener indices for the three example molecules. The calculation of these indices is shown explicitly for pentane. For the other two molecules, the index calculation is left as an exercise.



Index	Pentane	Isopentane	Glycidol
Wiener:	1+2+3+4+1+2+3+1+2+1=20	18	17
Hyper-Wiener:	$\frac{1}{2}(2+6+12+20+2+6+12+2+6+2) = 35$	28	26

The *Hyper-Wiener index* is a generalization of the Wiener index and is given by

$$\frac{1}{2}\sum_{v,w\in V(G)}(d(v,w)+d^2(v,w)). \tag{2.5}$$

Introduced by Randić in 1993, this modified index captures slightly more structural information due to the added squared term [15]. Randić also developed indices of his own, which are discussed in the next section.

2.4.2 The Randić Index

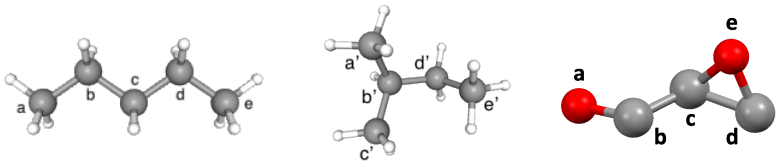
Another commonly used molecular index is the *Randić index*, which was first introduced in 1975 by Milan Randić.[16, 17] The Randić index takes the sum of bond contributions, where the bond contributions are given by $1/\sqrt{(D(v)D(w))}$, where $D(v)$ and $D(w)$ are the degrees of vertices v and w that make up a bond vw (see Figure 2.1).

In shorthand, if $E(V)$ are the bonds in the molecular graph, the Randić index is given by

$$\sum_{vw\in E(V)}(D(v)D(w))^{-1/2}. \tag{2.6}$$

Similar to the Wiener index, hydrogen atoms are not included in the calculation of the Randić index. We calculate the Randić index for our three example molecules in Figure 2.8. For example, there are four edges (bonds) between the carbon atoms of pentane. Two of them are between a terminal carbon atom (C_a or C_e) which have a degree of vertices equal to $D(a) = D(e) = 1$ and a carbon with a degree of $D(b) = D(d) = 2$. The other two edges (bonds) are between vertices with $D(b) = D(c) = D(d) = 2$. Thus, the Randić index for pentane is 2.414 (see Figure 2.8). The Randić index also served as the foundation for higher order connectivity indices, such as the Zagreb indices.

Figure 2.8: Top, Left to Right: Labeled pentane, isopentane, and glycidol. Bottom: The Randić index for pentane, isopentane, and glycidol, respectively. The calculation of the Randić index is given explicitly for pentane.



Index	Pentane	Isopentane	Glycidol
Randić:	$2(\frac{1}{(1\cdot2)^{1/2}}) + 2(\frac{1}{(2\cdot2)^{1/2}}) = 2.414$	2.270	2.432

2.4.3 Zagreb Indices

The first and second *Zagreb indices* were introduced by Gutman and Trinajstić in 1972.[18, 19] Let V be the vertex set and E be the edge set of some molecular graph G. Define the *first Zagreb index*, $M_1(G)$, and the *second Zagreb index*, $M_2(G)$, to be

$$M_1(G) = \sum_{v \in V} (D(v))^2 \tag{2.7}$$

and

$$M_2(G) = \sum_{vw \in E} D(v)D(w), \tag{2.8}$$

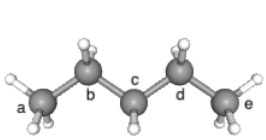
where $D(v)$ is the degree of the given vertex. According to Das, these indices can be used to show the amount of branching a molecule exhibits.[19] As the branching increases, the two Zagreb indices also increase.

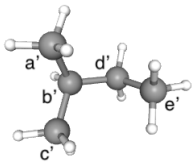
Later, Furtula, Graovac, and Vukičević introduced the *augmented Zagreb index*, as an extension of the atom-bond connectivity (ABC) index, outlined in Ref. [20] and shown in Table 2.1. The augmented Zagreb index is given by,

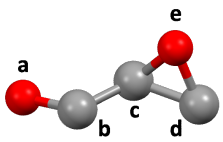
$$AZI(G) = \sum_{vw \in E(G)} \left(\frac{D(v)D(w)}{D(v) + D(w) - 2} \right)^3. \tag{2.9}$$

The calculated first, second, and augmented Zagreb indices for pentane, isopentane, and glycidol molecules are shown in Figure 2.9.

Figure 2.9: Top, Left to Right: Labeled pentane, isopentane, and glycidol. Bottom: A table of the three different Zagreb indices for each molecule. The calculation of these indices is given explicitly for pentane.







Index	Pentane	Isopentane	Glycidol
1st Zagreb:	$1^2 + 2^2 + 2^2 + 2^2 + 1^2 = 14$	16	22
2nd Zagreb:	$1 \cdot 2 + 2 \cdot 2 + 2 \cdot 2 + 2 \cdot 1 = 12$	14	24
Aug. Zagreb	$2(\frac{1 \cdot 2}{1+2-2})^3) + 2(\frac{2 \cdot 2}{2+2-2})^3) = 32$	22.75	40

2.4.4 Other Common Topological Indices

Finally, we summarize some other common topological indices in Table 2.1. Please see the table caption for variable definition and explanation.

Table 2.1: Let G be a molecular graph, V be the set of vertices with $|V| = n$, and E be the set of edges with $|E| = m$. Let $d(v)$ denote the degree of vertex v and let $d(v, w)$ denote the shortest distance between vertices v and w.

Index	Formula
ABC [21]	$ABC(G) = \sum_{vw \in E} \sqrt{\frac{D(v)+D(w)-2}{D(v)D(w)}}$
Hosoya [22]	# of matchings (a matching is a set of pairwise non-adjacent edges, none of which are loops)
Estrada* [23]	$EE(G) = \sum_{i=1}^{n} \exp(\lambda_i)$
Szeged [24]	$SZ(G) = \sum_{vw \in E} n_1(vw\|G)\ n_2(vw\|G)$ n_1: # of vert. lying closer to v than w n_2: # of vert. lying closer to w than v
Padmakar–Ivan [25]	$PI(G) = \sum_{e=vw \in E} n_{ev}(vw\|G)\ n_{ew}(vw\|G)$ n_{ev}: # of edges lying closer to v than w n_{ew}: # of edges lying closer to w than v
Gutman [26]	$Gut(G) = \sum_{v \neq w} D(v)D(w)D(v, w)$
Harmonic [27]	$H(G) = \sum_{vw \in E} \frac{2}{D(v)+D(w)}$
Geometric-Arithmetic [28]	$GA = \sum_{vw \in E} \frac{2\sqrt{D(v)D(w)}}{D(v)+D(w)}$

* For the Estrada index, let λ_i be an eigenvalue of the graph's adjacency matrix.

Although these indices provide molecular information, they are not enough to completely characterize the underlying structure of the molecule. For this, we turn to autocorrelation functions.

2.5 Autocorrelation Functions

The topological indices described in Section 2.4 have limited use in chemical applications of machine learning. However, they serve as an excellent example for understanding molecular representations with broader applicability in chemoinformatics and chemical data sciences. One family of such methods are the *autocorrelation functions* or topological autocorrelation vectors and their variants. Autocorrelation functions (ACs) are based on a formulation of the molecular graph, i.e. atoms are vertices, bonds are unweighted edges, and consider all possible subgraphs based on different atoms. ACs are invariant to system size and composition and do not depend on Cartesian or internal coordinates, but only on the connectivity of the molecular graph.

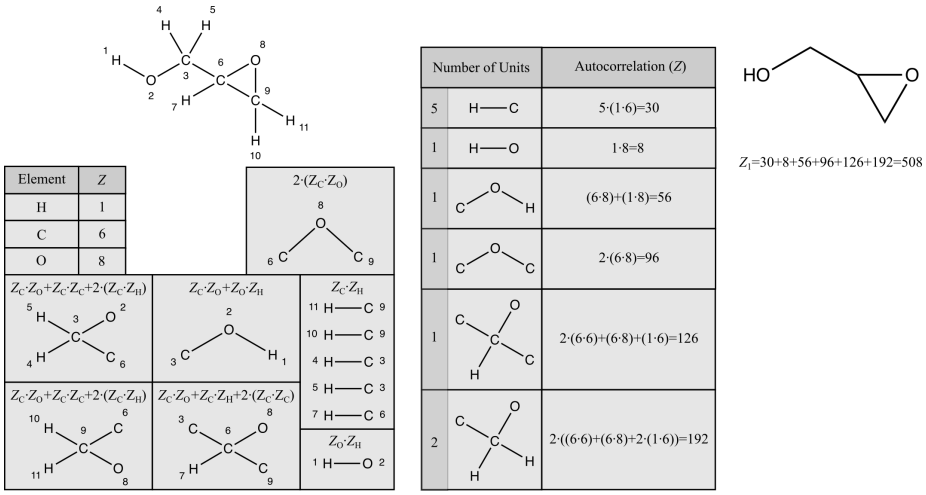
An ACs (P_d) correlates one atom with its neighboring atoms based on a series of atom-based properties such as element type, or electronegativity. For a specific property P of atoms v and w (P_v and P_w, respectively) at the depth d of a subgraph, the AC gets values from the following expression:

$$P_d = \sum_v \sum_w P_v \, P_w \, \delta(d(v,w),d), \tag{2.10}$$

The distance $d(v,w)$ between atoms v and w corresponds to the number of edges, between two atoms in a molecular graph. The Dirac delta $\delta(i,j)$ is a function that is 0 when two values i and j are different, and 1 when $i = j$. For example, for an AC of depth equal to 1, if an atom w is directly bonded to atom v, then $\delta = 1$, otherwise $\delta = 0$.

Figure 2.10 shows schematically how to compute the AC function for glycidol using the atomic number (Z). The values of the atomic properties P_v and P_w are equal to the atomic number of each atom (1 for H, 6 for C, 8 for O). We can label the computed AC as Z_d, where d is the depth of the subgraphs that we will consider. For sake of simplicity, we use a depth of 1 and in the computation of the AC we only show the cases where $\delta = 1$. For $d = 1$, glycidol has 6 unique subgraphs for a total of 11 subgraphs. For example, the H atom with label 1 forms a subgraph that includes only the neighboring atom O (H1-O2), while carbon with label 3 forms a subgraph with the four atoms that are directly bonded to it (O2, H4, H5, C6) After the computation of all nonzero elements, we have a final value of $Z_1 = 508$

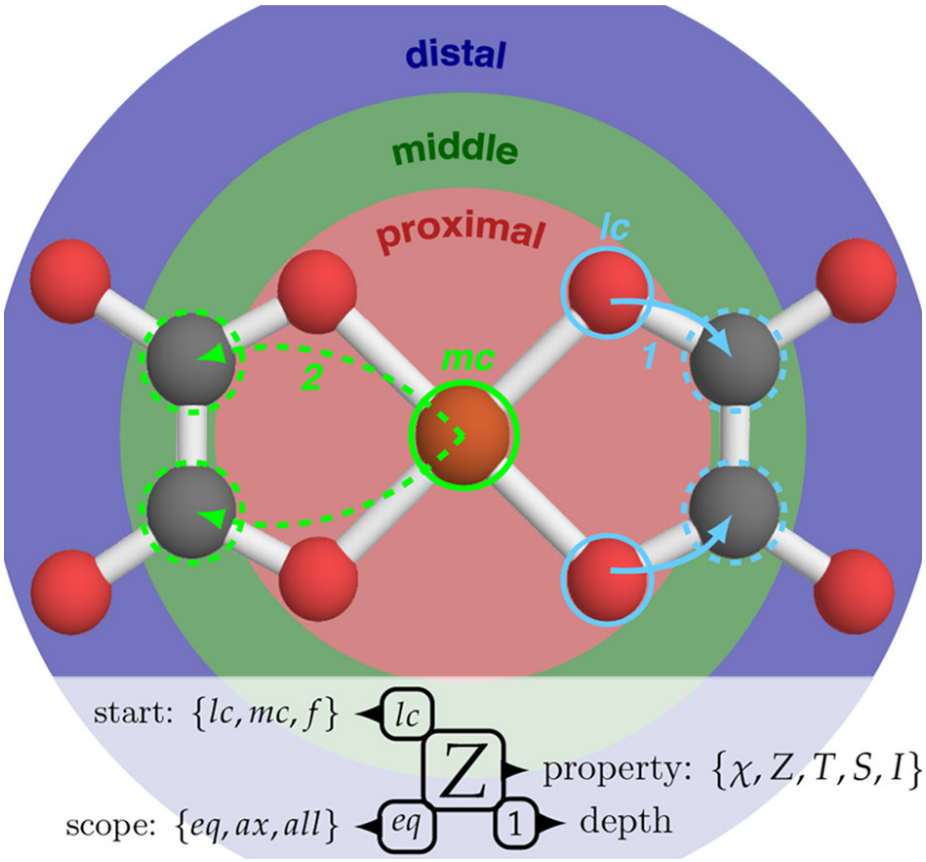
Figure 2.10: An example of the atomic number autocorrelation functions of depth one (Z_1) for glycidol. For sake of simplicity, we only show values where $\delta = 1$. The left part of the figure shows the subgraphs of glycidol with labels to aide in the understanding of the nonzero contributions to the autocorrelation function. The center shows the sum of each contribution and the far right shows the total Z_1 autocorrelation for glycidol.



The Kulik group introduced revised autocorrelation functions (RACs) for the study of inorganic complexes which have been a weak point for traditional ACs. The key new features of RACS are the additional atomic properties, the restricted sums to certain subgraphs, and the introduction of different mathematical operations, such as subtraction between properties, on the molecular graph.[29] Local and global information of a molecule is encoded by systematic, adaptable-resolution heuristics and topological descriptors. The five atomic properties introduced in RACs are nuclear charge (Z), Pauling electronegativity (χ), the atomic coordination number (T), covalent atomic radius (S), and the identity (I), which denotes the number of bonds in a subgraph such that P_v and P_w are equal to one for all bonds.

RACs contain two index terms, *start* and *scope*, which are introduced to account for the importance of the metal and its coordination sphere, and enforce different parts of the molecular graph to be utilized in each RAC. There are three *start* definitions which account for the full molecule (f), the metal-center (mc), and ligand-center (lc). The three *scope* options include all atoms (all), the axial ligand atoms (ax), or the equatorial ligand atoms (eq). The different types of RACs for a square planar complex are shown schematically in Figure 2.11 and are analyzed in the next paragraphs. Note that the traditional AC of Equation 2.10 uses the full molecular graph and all atoms and thus, it can be given as ${}^{\text{f}}_{\text{all}}P_d$ in the RACs notation.

Figure 2.11: Schematic representation of revised autocorrelation functions (RACs) applied to an square planar metal complex with equatorial oxalate ligands. The mc RAC (bright green) is of depth two (${}^{\mathrm{mc}}_{\mathrm{eq}}Z_2$)and the lc RAC (light blue) is of depth one (${}^{\mathrm{lc}}_{\mathrm{ax}}Z_1$). *Reproduced from Jon Paul Janet and Heather J Kulik. Resolving Transition Metal Chemical Space: Feature Selection for Machine Learning and Structure–Property Relationships. The Journal of Physical Chemistry A, 121(46):8939–8954, 2017.*



The first restricted-*scope* ACs incorporate the averaged properties of axial or equatorial ligands:

$$ {}^{\mathrm{f}}_{\mathrm{ax/eq}}P_d = \frac{1}{|\mathrm{ax/eq\ ligands}|} \sum_{v}^{n_{\mathrm{ax/eq}}} \sum_{w}^{n_{\mathrm{ax/eq}}} P_v\, P_w\, \delta(d(v,w),d), \tag{2.11} $$

where $n_{\mathrm{ax/eq}}$ is the number of atoms in either the axial or equatorial ligand.

The next restricted-*scope* ACs introduce the metal-centered descriptors, and

it is defined as:

$$ {}^{\mathrm{mc}}_{\mathrm{all}}P_d = \sum_{v}^{\mathrm{mc}} \sum_{w}^{\mathrm{all}} P_v \, P_w \, \delta(d(v,w),d) \tag{2.12} $$

where v runs over the metal atoms and w denotes all atoms in the complex.

Features of each atom in the first coordination sphere of the metal are incorporated using the restricted-*scope*, ligand-centered sum RAC:

$$ {}^{\mathrm{lc}}_{\mathrm{ax/eq}}P_d = \frac{1}{|\mathrm{ax\ /\ eq\ ligands}|} \frac{1}{|\mathrm{lc}|} \sum_{v}^{\mathrm{lc}} \sum_{w}^{n_{\mathrm{ax/eq}}} P_v \, P_w \, \delta(d(v,w),d). \tag{2.13} $$

To aide in the prediction of electronic properties of metal complexes, an electronegativity difference is considered rather than products of the conventional ACs (see Eq. 2.10):

$$ {}^{\mathrm{lc/mc}}_{\mathrm{ax/eq/all}}P'_d = \sum_{v}^{\mathrm{lc\ or\ mc}} \sum_{w}^{\mathrm{scope}} (P_v - P_w) \, \delta(d(v,w),d). \tag{2.14} $$

In this case, *scope* refers to either axial, equatorial, or all ligands and the *start* must be ligand-centered or metal-centered.

The six types of *start*/*scope* definitions (f/all; mc/all; lc/ax; lc/eq, f/ax; and f/eq) of ACs/RACs are combined for depths from zero, when applicable, to the maximum depth d using both products and differences of the five atomic properties. The RACs descriptor is a continuous vector space with a total of $42d+30$ product or difference RACs, composed of $30d+30$ product ACs/RACs ($6d+6$ descriptors for five atomic properties) and $12d$ difference descriptors. An example of a RACs descriptor for a transition metal is shown in Figure 2.11. The iron complex, where iron denoted by a brown sphere, oxygen is red, and carbon is gray, has two equatorial oxalate ligands and an octahedral geometry. The equatorial plane of the complex is shown with labels denoting the proximal region as the metal and first coordination sphere, the middle region is the second coordination sphere, and the distal region denotes the coordination sphere beyond the second coordination sphere. Two RACS are depicted in the figure: one mc RAC of depth two (${}^{\mathrm{mc}}_{\mathrm{eq}}Z_2$), shown in bright green, and one lc RAC (${}^{\mathrm{lc}}_{\mathrm{ax}}Z_1$), in light blue.

In the following Python example, we calculate the full complex ACs of glycidol, which include ${}^{\mathrm{f}}_{\mathrm{all}}\chi_d$, ${}^{\mathrm{f}}_{\mathrm{all}}Z_d$, ${}^{\mathrm{f}}_{\mathrm{all}}T_d$, ${}^{\mathrm{f}}_{\mathrm{all}}S_d$, and ${}^{\mathrm{f}}_{\mathrm{all}}I_d$, using molSimplify.[30, 31] For $d = 1$, 10 features will be generated: ${}^{\mathrm{f}}_{\mathrm{all}}\chi_0$, ${}^{\mathrm{f}}_{\mathrm{all}}\chi_1$, ${}^{\mathrm{f}}_{\mathrm{all}}Z_0$, ${}^{\mathrm{f}}_{\mathrm{all}}Z_1$, ${}^{\mathrm{f}}_{\mathrm{all}}T_0$, ${}^{\mathrm{f}}_{\mathrm{all}}T_1$, ${}^{\mathrm{f}}_{\mathrm{all}}S_0$, ${}^{\mathrm{f}}_{\mathrm{all}}S_1$, ${}^{\mathrm{f}}_{\mathrm{all}}I_0$, and ${}^{\mathrm{f}}_{\mathrm{all}}I_1$.

Python Example 1: Autocorrelation Functions

```
# An example of Autocorrelation Functions (ACs)
# using MolSimplify
import molSimplify.Informatics.autocorrelation as ac
from molSimplify.Classes.mol3D import mol3D
import pandas as pd

# Create mol object from xyz using the mol3D class
mol = mol3D()
mol.readfromxyz('glycidol.xyz')


# Create the ACs for the full structure with depth 0 to depth 1
#
# chi - electronegativity
# Z - nuclear charge
# I - identiy, i.e. 1 for
# T - atom coordination number
# S - covalent atomic radius
acs = ac.generate_full_complex_autocorrelations(mol,depth=1,
                                                loud=False)
# print dictionary containing acs
print(acs)

# Make a Pandas DataFrame to view the features together with
# the feature name and depth, i.e. featurename-depth
df_feat=pd.DataFrame(zip(sum(acs['colnames'],[]),
                         sum(map(list,acs['results']),[])),
                     columns=['Feature Name','Features'])
print(df_feat)
```

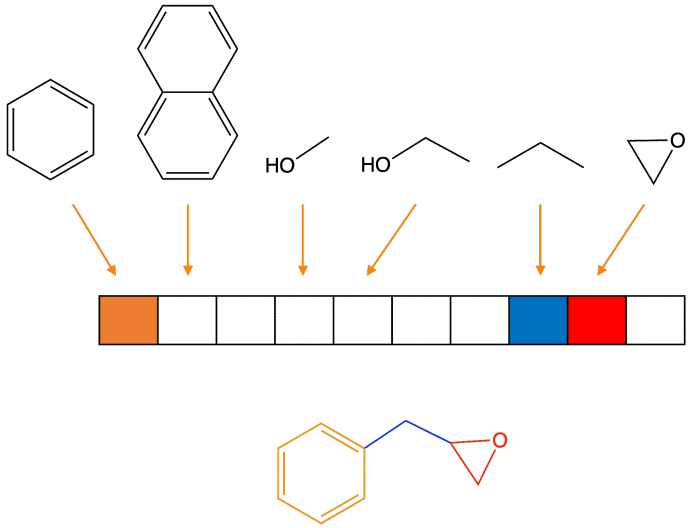
2.6 Structural Keys

Structural keys or *molecular fingerprints* are an alternative approach for developing molecular representations based on molecular graphs. These representations are nothing else but a vector containing information related to the structural features generated by decomposing a molecular structure into fragments. Since they contain information pertinent to molecular patterns, they have been extensively applied for the search of similar compounds in large databases.

Before we describe some common molecular fingerprinting methods, we need to provide definitions of the terms that we will use. A molecular fragment is defined as a structural motif present on molecules that are larger than an atom, and they can be atom pairs, common functional groups (e.g. amino groups, carboxylic groups, etc.), or ring structures. These structures are selected from a list ("dictionary") of molecular fragments. The information of these frag-

ments is stored is added in a vector called structural key or fingerprint. The vector elements can be given as "bits" (fragment present on a given structure or not, denoted as 1 or 0, respectively), as integers (e.g. how many times a specific fragment is present), or as real numbers (computed through mathematical formulas). Let us consider benzyloxirane as an example of a structural key (Figure 2.12). The bit vector has three non-zero elements that correspond to three fragments present in a hypothetical molecular dictionary (benzene, propane, ethylene oxide shown in orange, blue, and red, respectively).

Figure 2.12: Example of a structural key for benzyloxirane, where colors denote the presence of a molecular fragment in the structure key.



Common, well-established structural keys include *circular fingerprints*, such as the *Morgan* and *extended-connectivity fingerprints* (ECFPs), and the Molecular ACCess System (MACCS), which have been broadly applied in chemoinformatics as molecular representations, such as applications in molecular similarity searches.[32, 33]

2.6.1 Circular fingerprints

Circular fingerprints represent the atomic neighborhoods in molecules based on the radial or circular growth of the descriptor.[34] Each circular fingerprint offers a unique interpretation of how the fingerprints are iteratively updated based on the environment of each atom. While circular fingerprints tend to only describe bonds around an atom within a given radius and information about bonding and connectivity of atoms between layers is not available, the overlap of features allows for the implicit connection of disjointed fragments.

The subsequent fingerprint generated by the the Morgan algorithm (Morgan fingerprint) was introduced in 1965 by H. L. Morgan to assign unique labels to

chemical structures for the purposes of computer generated registrations for the Chemical Abstracts Service (CAS).[3] The key innovation in Morgan's method was the proposed algorithm to solve the permutational invariance of identical molecules with different atomic numbering schemes (molecular isomorphism). The Morgan fingerprint utilizes the molecular graph to generate compact connection tables.

In most software packages, such as RDKit and Open Babel, the Morgan fingerprints are often *extended-connectivity fingerprints* (ECFPs) generated based on the Morgan algorithm.[35] Unlike the traditional Morgan algorithm, ECFPs do not require maximum disambiguation to represent a viable representation. The ECFP algorithm terminates after a set number of iterations and the set of all atomic identifiers is retained to generate ECFPs. The ECFP algorithm uses a fast-hashing scheme for computational efficiency and helps avoid assigning the same identifier to two different atomic environments, or bit collisions which are common in this method. Like many other representations, ECFPs are based on molecular graphs and exclude hydrogen atoms and bonds in the fingerprint. Since ECFPs are a method that updates the fingerprint iteratively, it is common to see the method labeled "ECFP_4" or "ECFP_6". The number following the underscore denotes the diameter of the largest feature and is equal to double the number of iterations performed.

ECFPs are generated in three stages: 1) the initial assignment, 2) iterative updating, and 3) duplicate identifier removal. In the initial assignment stage each atom is assigned an integer identifier, which are collected into an initial fingerprint set. Various integer values can be used for the initial atom identifiers, as long as they are independent of atom numbering.

The initial identity of an atom is hashed into a single 32-bit integer based on the Daylight atomic invariants criteria, plus an additional criteria for whether an atom is contained in at least one ring. These criteria are:

- Number of connections
- Number of non-hydrogen bonds
- Atomic number
- Sign of charge
- Absolute charge
- Number of attached hydrogen atoms

It should be noted that during the 0th iteration, the bond set of each atom is an empty set.

The iterative updating stage incorporates the circular neighborhood of an atomic environment and includes information about whether the structural feature is a duplicate. The starting point of each iteration are the previous iterations atomic identifiers. The identifiers of the target and neighboring atoms are collected into an array ordered by the their identifiers and attached bonds

to avoid dependence on the ordering. To reduce this representation down to a single integer identifier, a hash function is applied. For a set number of iterations, each atom identifier is iteratively updated and added to the fingerprint set. Once all atoms have been updated with a new identifier the iteration is ended. For a single iteration the order of operations is as follows:

1. For a given atom, the iteration number and atomic identifier is placed in an array of integers

2. The attached atoms are then sorted based on the bond order: single, double, triplet, and aromatic (based on Hückel's $4n + 2$ formula)

3. The bond order and the identifier for each attachment is added to the array

4. If stereochemical fingerprints are requested and an atom is a possible stereoatom that has not been disambiguated, with different identifiers for each attachment atoms, a flag for stereochemistry is appended to the feature array and marked as disambiguated

5. A new atomic identifier is generated by a hash function as a 32-bit integer

Features which occur more than once are removed and condensed into a single feature after the last iteration is performed. Despite having unique hashed identifiers, structurally identical regions of molecules lead to so called structural duplication, which can introduce redundancies in the fingerprint. Duplicate features are removed based on two criteria. If the redundancies were generated during different iterations, the one from the largest iteration is discarded or if they were generated during the same iteration the one with the larger hashed 32-bit integer is discarded. Since the generation of each identifier is based on the local nature of each iteration, this highlights the power of the Morgan algorithm for the iterative generation of unique fingerprints.

The following code example and figure (Figure 2.13) show an example of generating an ECFP or Morgan fingerprint for 2-benzyloxirane using RDKit. In this example, we use a radius of 2 for simplicity.

Python Example 2: Morgan Fingerprints

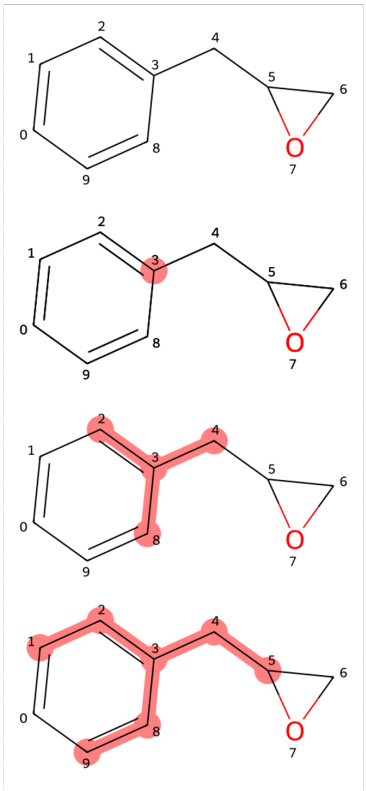
```
# This is an example of Morgan fingerprints as implemented in RDKit
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import MolFromSmiles,MolToSmiles
from rdkit.Chem import Draw
from rdkit.Chem.Draw import rdMolDraw2D
# Glycidol
smi='OC[C@@H]1CO1'
# 2-benzyloxirane
# smi='c1ccc(C[CH]2CO2)cc1'
mol=MolFromSmiles(smi)

# Prepare an interpretable Morgan Fingerprint instance
info={}
fp = AllChem.GetMorganFingerprint(mol,2,bitInfo=info)

# info will be a dictionary with bit id keys and
# values of (atom index, radius)
print(info)
# The first bit, 98513984, is set three times:
#    ((1, 1), (0, 1), (9, 1)).
# This means that it is set by atom 1, 0, and 9 at radius 1.
```

Using the code above we can further examine the iterations of the Morgan fingerprint for 2-benzyloxirane. Using the radius criteria of 2, we see in Figure 2.13 that at each radius iteration up, until the max, a bit is generated to describe the environment highlighted in red. In this case, we would have to use a radius of 3 to capture the full molecular information from atom 3. Luckily, due to the Morgan algorithm, each atom is used as a starting point so the information related to the environment of the full molecular structure is included in this representation.

Figure 2.13: Example of an extended-connectivity fingerprint (ECFP) for benzyloxirane implemented in RDKit with a max radius of 2. The labeled benzyloxirane is shown before iterations begin and for each iteration the associated bit is also included. For iteration 0, the associated bit only includes information about atom 3. Iteration 1 includes information about the first-neighbors of atom 3, the bit includes information about the bonds attached to the neighboring atoms. In iteration 2, the second-nearest neighbors are incorporated. To incorporate all atoms in the molecule a third iteration would be required.



2.6.2 Molecular ACCess Systems Fingerprint (MACCS)

The *Molecular ACCess Systems fingerprint* (MACCS) was introduced in 2002 by MDL Information Systems.[36] The 166 bit MACCS fingerprint, the most commonly applied and implemented version of the MACCS fingerprint, is described in the white paper by BIOVIA (formerly MDL Information Systems and Accelrys).[37] In this section, we refer to the 166 bit MACCS fingerprint as a MACCS fingerprint or key, both of which denote the same thing. Also, we only discuss this version of the MACCS key since it has been implemented in common cheminformatics packages such as RDKit[9], Open Babel[10], and

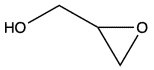
OpenEye Toolkits.[38]

The 166 bits in the MACCS key denote the absence of a key by 0 or the presence of a key using a 1. The keys are defined using a set of symbols to describe the atomic and bonding environments of a molecule.[39, 40] The atomic symbols are: A (any valid element from the periodic table), Q (any non-carbon or non-hydrogen atom), X (F, Cl, Br, and I), and Z (any atom other than H, C, N, O, Si, P, S, F, Cl, Br, or I). Common bonds such as single bonds are denoted by −, double by =, and triple by T or #. Single and double query bonds are denoted by ~ and aromatic query bonds by %. Any bond type, i.e. an unspecified bond type, is denoted by $None$. When a $ appears before a bond type it specifies that it is a bond in a ring and when a ! appears before a bond type it denotes a chain bond. When @ is followed by an integer, it denotes where ring linkage occurs. Aromatic denotes either Kekule or Arom5, where Kekule is a benzene ring and Arom5 is a five-membered ring with a Q, or heteroatom, at the top of the ring.

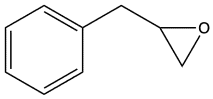
We will give a brief, general overview of the 166 MACCS keys. Keys 1 and 49 denote whether a isotope and charge are present, respectively. Several keys exist, 2, 3, 4, 5, 6, 7, 9, 10, 12, 18, and 35, to related where atoms are located on the periodic table, i.e. what periods and groups atoms belong to. Numerous keys exist to denote the presence of cyclic groups in molecules (11, 19, 22, 36, 57, 96, 101, 121, 137, 162, 163, and 165). The count of various fragments are denoted by 118, 120, 125, 136, 138, 140-142, 145, 146, 149, and 159. The remaining keys (8, 13-17, 20, 21, 23-34, 37-48, 50-56, 58-95, 97-100, 102-117, 119, 122, 123, 124, 126-135, 139, 143, 144, 147, 148, 150-158, 160, 161, 164, 166), denote various fragments that can be found at Refs. [39] and [40].

Since most MACCS implementations do not include documentation on what each bit in the MACCS fingerprint denote, in our code example we show how to generate an interpretable MACCS key for glycidol, shown in Figure 2.14, using the documentation from Refs. [39] and [40]. Since MACCS are often used in similarity searches, this example provides powerful insight into the similarities and dissimilarities of our sample molecules, glycidol and 2-benzyloxirane, in Figure 2.14.

Figure 2.14: An example of the bits in the MACCS keys for glycidol and 2-benzyloxirane that correspond to 1. The tables in the figure show the keys that equal to 1 and what the values correspond to based on the interpretation of MACCS in Refs. [39] and [40].



Key	Value
16	QAA@1
22	3M RING
57	O HETEROCYCLE
72	OAAO
82	ACH2QH
89	OAAAO
90	QHAACH2A
91	QHAAACH2A
109	ACH2O
129	ACH2AACH2A
132	OACH2A
137	HETEROCYCLE
138	QCH2A > 1 (&...)
139	OH
152	OC(C)C
153	QCH2A
155	A!CH2!A
157	C-O
159	O > 1
164	O
165	RING



Key	Value
16	QAA@1
22	3M RING
57	O HETEROCYCLE
109	ACH2O
129	ACH2AACH2A
132	OACH2A
137	HETEROCYCLE
152	OC(C)C
153	QCH2A
155	A!CH2!A
157	C-O
162	AROMATIC
163	6M RING
164	O
165	RING

Python Example 3: MACCS Key

```
# This example includes examples of the MACCS key
# as implemented in RDKit
# This is a SMARTS-based implementation of the 166 public MACCS keys
import pandas as pd
from rdkit.Chem import MACCSkeys
from rdkit.Chem import MolFromSmiles
# glycidol
smi='C1C(O1)CO'
# 2-benzyloxirane
# smi='c1ccc(C[CH]2CO2)cc1'
mol=MolFromSmiles(smi)
MACCS=list(MACCSkeys.GenMACCSKeys(mol))
# Print MACCS key
length=len(list(MACCS))
print(f'Length of MACCS key={length}')
# Print MACCS as a string of consecutive numbers
print(''.join(map(str,list(MACCS))))


# Keys from http://www.mayachemtools.org/index.html
keys=pd.read_excel('MACCS_keys_example.xlsx').drop(
    columns=['Unnamed: 0'])

# Find the keys that appear in the fingerprint
mol_keys=[idx for idx, i in enumerate(MACCS) if i==1]
# Print what the bit corresponds to
print(keys.set_index('Key').loc[mol_keys])
```

2.7 SMILES Notation and its Variants

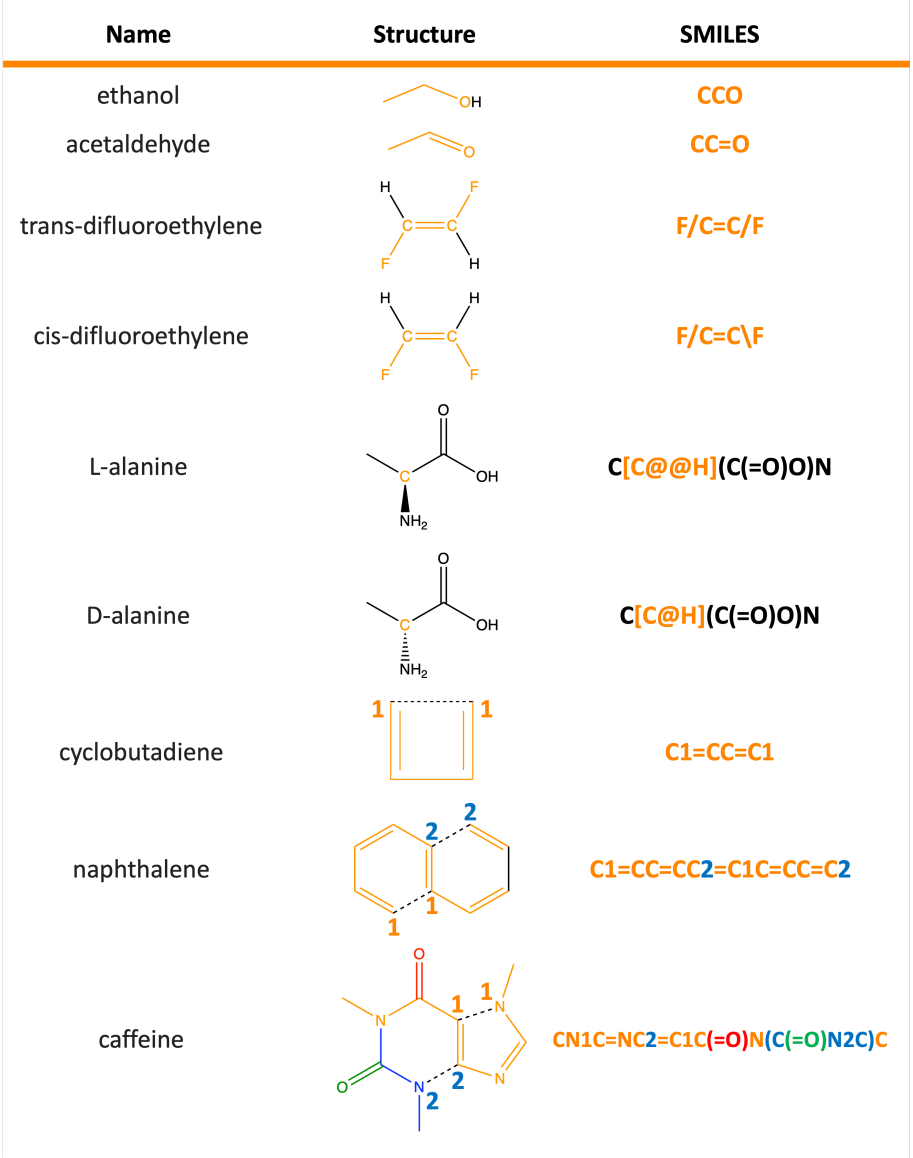
One of the most ubiquitous molecular representations is the *Simplified Molecular Input Line Entry System* (SMILES), first introduced in 1988 by Daylight Chemical Information Systems, Inc. as a representation that was developed with computational efficiency in mind.[41] Daylight also introduced other related line notations such as SMARTS (discussed in Section 2.7.2) and SMIRKS. These line notations have inspired other novel representations built specifically for machine learning such as DeepSMILES and SELFIES. In this section, we will provide examples of SMILES, SMARTS, DeepSMILES, and SELFIES using Python so that the reader can apply these methods on their own.

2.7.1 Simplified Molecular Input Line Entry System (SMILES)

The SMILES notation was proposed as a user and machine friendly, unique graph-based approach which has been broadly applied in machine learning applications such as molecular design and discovery.[42, 43] In the SMILES nota-

tion, the three-dimensional representation is ignored and the SMILES structure is built based on the traditional two-dimensional molecular graph. In SMILES a hydrogen-suppressed graph or hydrogen-complete graph, the graph which includes the hydrogen atoms explicitly, can be used. SMILES represents molecules as a series of characters followed by a space. Atoms, the most fundamental building blocks of a molecule, are represented by their atomic symbols. Hydrogen atoms are usually omitted since they can be obtained from the valencies of the other atoms. For example, methane (CH_4) is represented by "**C**", which implies that the carbon is bonded to 4 hydrogen atoms. The formal SMILES encoding of bonds are "**-**" for a single bond, "**=**" for a double bond, "**#**" for a triple bond, and "**:**" for an aromatic, but single and aromatic bonds are often omitted, as explained below. For example, ethanol (CH_3CH_2OH) is given as **CCO**, while acetaldehyde ($CH_3CH{=}O$) is represented as **CC=O** (see Figure 2.15). Formal charges and additional information on the attached hydrogens can be enclosed in brackets. For example, a proton is represented by **[H+]**, the hydronium cation is given by **[OH3+]**, while an isotope label precedes the atom character (e.g. **[14C]**). Disconnected structures, like an anion with an associated cation, are denoted as two separate SMILES joined by a period, e.g. sodium chloride is given as **[Na+].[Cl-]**.

Figure 2.15: Examples of SMILES strings.



The SMILES notation permits, but does not require, stereochemical information, which can be explicitly introduced with a specific notation. Cis- and trans- stereoisomers can be specified with the help of the "/" and "\" characters. When these two symbols are around a double bonds, they indicate the direction of the single bonds adjacent to the double bond. For example, F/C=C/F

indicates that the first fluorine atom is below the double bond axis, while the second fluorine atom is above and thus, it represents the trans-difluoroethylene (see Figure 2.15). On the contrary, the **F/C=C\F** notation is used for cis-difluoroethylene where both fluorine atoms are on the same side of the double bond. Chiral centers can also be explicitly introduced by considering the order of the bonds as they appear on the SMILES string. The symbol **@@** is used when the four bonded atoms are given in clockwise order, and the **@** when they are in anticlockwise order (since @ is already pointing in an anticlockwise direction!). For example, L-alanine (Figure 2.15) is represented as C**[C@@H]**(C(=O)O)N since we start the clockwise ordering from the first methyl group, then move on to the hydrogen atom that points at the back side of your screen (not included in the structure formula of Figure 2.15), then we continue to the carboxylic acid group (shown as C(=O)O in the SMILES string, *vide infra*), and finally we consider the amino group -NH_2 which points at the front side of your screen. The opposite holds for D-alanine, where connectivity of the chiral center follows an anticlockwise order.

A cyclic structure is represented with a numeric suffix that is attached as a label next to the first and last atoms that close the molecular ring. Thus, cyclobutadiene (**C_4H_4**) is given as **C1=CC=C1**, while naphthalene (**$C_{10}H_8$**) that has two aromatic rings, is given as **C1=CC=CC2=C1C=CC=C2** (Figure 2.15). In the rare cases of cyclic structures having 10 or more rings, the ring closure would be denoted with a percent sign (**%**). Sometimes, aromatic atoms are denoted with lower case symbols, i.e. a carbon atom in benzene would be denoted as "**c**" whereas a carbon atom in cyclohexane would be denoted as "**C**".

When we are constructing SMILES strings, it is important to identify the main atom chain in the molecular graph. Since more than one chain is usually included in a molecular graph, the main chain can be defined based on specific assumptions, such as the starting atom, the longest chain, or the number of bonds chosen to break cycles. Once the main chain has been identified, all the other chains or branches are denoted by parentheses (). For complex structures, it is common to define branches inside branches, which leads to parentheses inside other parentheses. Let us consider caffeine as a representative example, which is shown at the bottom of Figure 2.15. The main chain is denoted in orange. It begins from the methyl group ("**C**") that is next to the five-member ring (upper right of figure), while the nitrogen that is part of that ring has label "**N1**". As we "move" around the five-member ring, we encounter the first atom ("**C2**") that is attached to the six-member ring. A side (blue) chain appears at the first (orange) nitrogen atom of the main chain, which also includes a second (green) side chain. Of course, we could have chosen to add the blue-label atoms into the main chain of the molecule, and treat the methyl group as a side chain. This topic is further discusses in the next paragraph.

Since the rules above can generate numerous non-unique SMILES, canonicalization schemes are required to generate unique structures. While numerous canonicalization schemes exist, including Universal SMILES[44] and RDKit SMILES[45], we will discuss the original SMILES canonicalization scheme called CANGEN. CANGEN was the first proposed SMILES canonicalization

algorithm which consists of two parts: first, CANonicalization (CANON) where each atom is ordered and labeled using molecular graph theory and second, GENerated (GENES) where a unique SMILES is created using the lowest labeled atoms.[46] A ranking system can be devised based on the connectivity and node properties for an undirected molecular graph. The node ranking system utilized in CANGEN can differentiate node environments based on the initial set of node properties. In order to be effective, CANGEN must be able to differentiate nodes that are topologically identical. Hence, unique node orderings are generated by CANGEN via topological symmetry classes. Canonicalization is guaranteed by the utilization of an unambiguous function for structural notations which include labeling, ranking, and unique ordering. For further information about the algorithm, we refer the reader to the original article (Weiningen *et al.*, *J. Chem. Inf. Comput. Sci.,* **1989**, *29*, 97–101).[46].

2.7.2 Popular Variants of SMILES

One of the most popular extensions of SMILES is the *SMARTS* molecular representation. SMARTS build on the SMILES notation by introducing logical operators that allow for a more general specification of atoms and bonds specifically for substructure searches.[47] The main advantages of using SMARTS over SMILES are its' generality, ability to handle reactions, and query molecular structures. It should be noted that SMILES describe molecular graphs and SMARTS describe patterns, so most SMILES are valid SMARTS but most SMARTS are not valid SMILES.

Starting with atomic properties, SMARTS introduces a wildcard symbol, $*$, which can be used to query any atom with certain properties. An example would be [$*$**C2**] which means any atom with exactly two carbons attached. A general aromatic atom is denoted as **a**. Atoms can also be denoted using $\#n$, where n is the atomic number. To search for atoms with <n> total bonds including implicit hydrogens use the connectivity **X<n>** and for the total bonds without implicit hydrogens use the degree flag **D<n>**. For an atom with 3 total bonds this would be denoted as **X3**, where as an atom with 3 explicit bonds would be **D3**. Along this line, the total number of attached hydrogens and the number of implicit hydrogens can be queried using **H<n>** and **h<n>**, respectively. Ring membership can be searched by using **R<n>**, where **[R]** denotes any atom in any ring. The size of a ring be searched using **r<n>**, where **[r6]** would search for any ring of size 6. To find the total number of ring connections **x<n>** is utilized, where **x2** would look for an atom with two ring connections. Bond orders can be searched using the valence flag **v<n>**, which includes implicit hydrogens. To search for an atom with a bond order of four, **v4** would be used. SMARTS have the ability to search for chirality as well. If the chirality is specified **@<c><n>** matches the chirality, where <c> denotes the chirality (**@** for anticlockwise and **@@** for clockwise). By using **@<c><n>?** the chirality is matched if it is specified or will not if it is unspecified.

SMARTS also introduces a similar notation for bonds where a wildcard ~ is applied to denote any bond and @ to denote any ring bond. Like in SMILES /

and \ denote directional bonds but when used with a **?**, **\?** denotes a down or unspecified directional bond and **/?** denotes an up or unspecified bond.

To assist with structure querying, common logical operators, such as NOT, OR, or AND, are introduced. For a general atom or bond e, **!e** denotes NOT e, **e1&e2** denotes e1 AND e2 with high precedence, **e1;e2** denotes e1 AND e2 with low precedence, and **e1,e2** denotes e1 OR e2. Additional querying methods, such as recursive SMARTS, component-level groupings of SMARTS, and reaction queries can be found at Daylight SMARTS.

We will now turn our attention to two more modern methods, developed with machine learning in mind. When used in generative models, syntactically invalid SMILES are often produced. A large fraction of the possible SMILES generated do not correspond to a proper molecular graph or they violate basic chemistry rules, such as maximum valency rules. One proposed method is the *DeepSMILES* representation which was developed to address the issues related to parentheses balancing and the pairing of ring closure symbols in SMILES.[48] DeepSMILES avoids unmatched ring closure digits by using a single digit, instead of two, to denote ring closures. The ring-opening symbol is removed and the ring-closing symbol is replaced by **%N** for double and **%(N)** for triple digit ring size. A postfix notation is used to avoid the use of paired parentheses where branch length is denoted by the number of close parentheses used. An example of this is benzene, which has a SMILES string c1ccccc1, whereas in DeepSMILES the benzene would be denoted as ccccccc6. Examples of more complex molecules are discussed in Subsection 2.7.3, where we discuss glycidol and 2-benzyloxirane. It should also be noted that SMILES and DeepSMILES can be interconverted without loss of information.

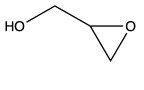
The second proposed method is the *SELF-referecIng Embedded Strings* (SELFIES) method.[49] SELFIES is proposed as a robust method that represents every molecule, even randomly generated strings, with a valid string-based descriptor. SELFIES was developed in order to be paired with machine learning models and as such, they offer a significant advantage over SMILES for molecule generation tasks. Due to a more complex set of rules, every molecule generated by a machine learning model will correspond to a valid molecule that is independent of the machine learning model. SELFIES show the branch length along with the ring size are stored with the corresponding branch and ring identifiers. To enforce the validity of chemical bonds, a formal grammar derived from theoretical computer science is used. For a more in-depth discussion of the SELFIES derivation rules refer to the supporting information document of Ref. [49].

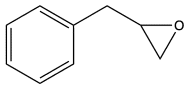
2.7.3 Examples of SMILES, SMARTS, DeepSMILES, and SELFIES

We will now compare and contrast SMILES, SMARTS, DeepSMILES, and SELFIES (Figure 2.16) for our example molecules using the provided code example. Glycidol has a very simple SMILES string where C1CO1 denotes the oxirane group and OC denote the hydroxymethyl group. For our more complex molecule, 2-benzyloxirane, the SMILES string is c1ccc(CC2CO2)cc1,

where c1ccccc1 denotes a benzene ring and CC2CO2 denotes the methyloxirane functional group. The parenthesis denote a substitution at the specific carbon in the benzene ring. The SMARTS for glycidol and 2-benzyloxirane are [#6]1-[#6](-[#8]-1)-[#6]-[#8] and [#6]1:[#6]:[#6]:[#6](-[#6]-[#6H]2-[#6]-[#8]-2):[#6]:[#6]:1, respectively. In this example, the only difference between the SMILES and the SMARTS generated by RDKit are the explicit incorporation of the bond type (i.e. - for single and : for double) and the use of the atomic number instead of the atomic symbol. In the DeepSMILES represenation, the main difference between the two molecules is the presence of the benzene ring in the 2-benzyloxirane. The most complex representation in this example is SELFIES, which is not readily interpretible without referencing the derivation rules.

Figure 2.16: A comparison of SMILES, SMARTS, DeepSMILES, and SELFIES for glycidol and 2-benzyloxirane generated using the Python code below.





SMILES	OCC1CO1	c1ccc(CC2CO2)cc1
SMARTS	[#6]1-[#6](-[#8]-1)-[#6]-[#8]	[#6]1:[#6]:[#6]:[#6](-[#6]-[#6H]2-[#6]-[#8]-2):[#6]:[#6]:1
DeepSMILES	CCO3)CO	ccccC[CH]CO3))))cc6
SELFIES	[C][C][Branch1][Ring2][O][Ring1][Ring1][C][O]	[C][=C][C][=C][Branch1][#Branch1][C][CH1][C][O][Ring1][Ring1][C][=C][Ring1][#Branch2]

Python Example 4: SMILES and Derivatives

```
# This example includes examples of SMARTS and SMILES
# RDKit
from rdkit.Chem import MolFromSmiles, MolToSmarts, MolToSmiles
# Glycidol
# smi='C1C(O1)CO'
# 2-benzyloxirane
smi='c1ccc(C[CH]2CO2)cc1'
mol=MolFromSmiles(smi)
# Print SMILES and SMARTS
print('SMARTS',MolToSmarts(mol))
print('SMILES',MolToSmiles(mol))

# DeepSMILES
import deepsmiles
converter = deepsmiles.Converter(rings=True, branches=True)

# DeepSMILES encoding of 2-benzyloxirane from canonical SMILES
encoded = converter.encode(smi)
# Print the DeepSMILES
print('DeepSMILES', encoded)

# Example of SELFIES
import selfies

# SMILES to SELFIES
try:
    selfie = selfies.encoder(smi)
    # Print SELFIE
    print('SELFIES',selfie)
except selfies.EncoderError:
    pass  # selfies.encoder error!
except selfies.DecoderError:
    pass  # selfies.decoder error!
```

2.8 International Chemical Identifier (InChI)

The International Chemical Identifier (InChI) was first introduced in 2005 by the International Union of Pure and Applied Chemistry (IUPAC) as a non-proprietary molecular representation for the identification of printed and electronic data.[50] Unlike SMILES, InChIs are a unique machine-readable identifier that is not meant to be interpreted by humans. The goal of InChI is to provide a unique inline string representation based on the molecular graph. InChI has a set of rules for normalization and canonicalization based on conventions derived by IUPAC for molecular structures.

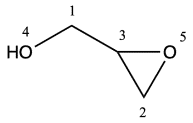
InChI can be consider as a unique structure identifier, similar to a bar code,

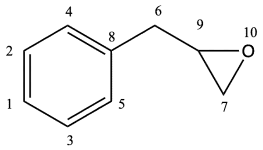
based on the molecular structure for which it was generated. The identifier consists of several layers of structural information, each containing unique information, that can be generated in an automated manner. Each layers adds additional information to the representation that the prior layer did not provide. The layered structures provides different levels of abstraction for identical molecules with varying stereochemistry or tautomerism; if one molecule omits such bonds but another one includes them, the more refined structure will include the description of the less refined structure.

The five layers used for creating InChIs are related to the chemical formula (including no formal bond orders), connectivity (information about disconnected/connected metals, etc.), isotopes, stereochemistry, and tautomerization. Except for the first layer that describes the chemical formula, each layer of the InChI string is separated by a / followed by a lowercase letter. The first position in an InChI corresponds to the InChI version name, which includes an S if it is the standard version, followed by a / with the chemical formula. The connectivity is defined by /c, which excludes terminal hydrogens, and /h, which denotes the locations of terminal and mobile hydrogen attachment positions. Next the charge is included by /q and the proton balance by /p. The parity of tetrahedral centers is denoted by /t and the relative stereo obtained by inverting the parity (inverted equal to 1 and not inverted equal to 0) is denoted by /m. The type of stereo center, whether it is absolute (=1), relative (=2), or racemic (=3), are denoted by /s. The chemical formula of the fixed hydrogen structure, if it is different than the original chemical formula, is denoted by /f. Locations of fixed mobile hydrogens are denoted by /h.

In Figure 2.17 we provide an example of two InChIs generated by RDKit for glycidol and 2-benzyloxirane. In both InChIs, 1S is used to denote that they are standard version 1 InChIs. The connectivity of glycidol and 2-benzyloxirane are /c4-1-3-2-5-3 and /c1-2-4-8(5-3-1)6-9-7-10-9, respectively. Note that in 2-benzyloxirane, 1-2-4-8(5-3-1) is the benzene ring, where atoms 5, 3, and 1 are treated as branches from atom 8. In glycidol, /h3-4H,1-2H2 denotes that there is one hydrogen on atoms 3 and 4 and two hydrogens on atoms 1 and 2. /h1-5,9H,6-7H2 denotes the 10 hydrogen atoms of 2-benzyloxirane, which are assigned as follows: 1-5,9H means that atoms 1-5 and 9 have one hydrogen and 6-7H2 means that atoms 6 and 7 both have two hydrogens. We provide the code used to generate this example in the following Python example.

Figure 2.17: The InChI string for glycidol and 2-benzyloxirane generated using the Python code below.





InChI 1S/C3H6O2/c4-1-3-2-5-3/h3-4H,1-2H2 1S/C9H10O/c1-2-4-8(5-3-1)6-9-7-10-9/h1-5,9H,6-7H2

Python Example 5: InChI

```
# This is an example of InChI and InChI key as implemented in RDKit
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import MolFromSmiles,InchiToInchiKey,MolToInchi
from rdkit.Chem import Draw
from rdkit.Chem.Draw import rdMolDraw2D
# Glycidol
# smi='C1C(O1)CO'
# 2-benzyloxirane
smi='c1ccc(C[CH]2CO2)cc1'
mol=MolFromSmiles(smi)
# RDKit mol object to InChI
print(MolToInchi(mol))
# InChI to InChIKey
print(InchiToInchiKey(MolToInchi(mol)))
```

Available Software Packages

Packages with common representations covered in this Chapter:

- Open Babel
  - Wiswesser Line Notation
- RDKit
  - RDKit fingerprints:
    - Morgan
    - MACCS
    - Hashed topological torsion,
    - Hashed atom pair
  - SMARTS
  - SMILES
  - InChI
- ChemML
  - RDKit fingerprints
  - SMARTS
  - SMILES
  - InChI

2.9 That's a Wrap

- Common graph-based molecular representations include matrices, topological indices, autocorrelation functions, and line notations.
- A molecular graph characterizes atoms as nodes and bonds as edges. Adjacency matrices and distance matrices can be used to capture the underlying connectivity and the molecule's structure.
- Topological indices and autocorrelation functions are two molecular representations that utilize connectivity information to summarize the underlying structure.
- Structural keys, SMILES, and InChI are a few popular line notations which are widely applied vectrorizations in chemoinformatics and chemical machine learning.
- Additional resources related to molecular graphs and graph-based molecular representations can be found in these textbooks and review articles:

- Chemical Graph Theory, Nenad Trinajstić, **1992**, CRC Press.
- Chemoinformatics: Basic Concepts and Methods, Thomas Engel and Johann Gasteiger, **2018**, John Wiley & Sons.
- Daniel S. Wigh, Jonathan M. Goodman, and Alexei A. Lapkin. A Review of Molecular Representation in the Age of Machine Learning. *WIREs Comput. Mol. Sci.*, **2022**, *12*, e1603.
- Laurianne David, Amol Thakkar, Rocò Mercado, and Ola Engkvist. Molecular Representations in AI-Driven Drug Discovery: a Review and Practical Guide. *J. Cheminformatics*, **2020**. *12*, 56.
- Daniel C. Elton, Zois Boukouvalas, Mark D. Fuge, and Peter W. Chung. Deep Learning for Molecular Design - a Review of the State of the Art. *Mol. Syst. Des. Eng.*, **2019**, *4*, 828–849.
- Shampa Raghunathan and U. Deva Priyakumar. Molecular Representations for Machine Learning Applications in Chemistry. *Int. J. Quant. Chem.*, **2022**, *122*, e26870.

Chapter 3

Topology-based Representations

3.1 Introduction

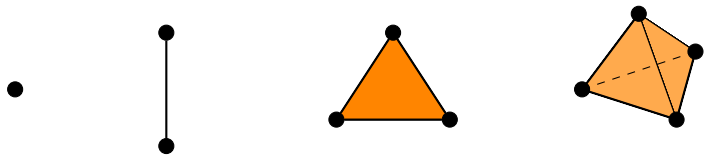
In Chapter 2, we saw the importance of molecular topology for the generation of useful representations. We explored the topology of a chemical structure through matrices, topological indices, functions, molecular fragments, and line notations, all of which centered around the molecular graph. These features and other topological properties are useful chemical representations as they are preserved independently of the spacial orientation or the representation of the molecule. In this Chapter, we will survey persistent homology-based approaches. Persistent homology, a powerful topological tool, offers an alternative methodology for capturing the underlying structure of a molecule, and has been used successfully in numerous applications as an alternative molecular representation. Some recent examples of this emerging methodology in chemoinformatics and chemical data sciences include molecular similarity searches[51, 52], molecular discovery[53], and HOMO-LUMO gap minimization.[54] In this Chapter, we will provide the fundamentals of persistent homology, visualization techniques, and extensions to alternative encodings that capture additional chemical information.

3.2 Simplicial Complexes

To discuss some of the most common topological properties, we first need to introduce simplicial complexes. Simplicial complexes are made up of k-simplices, or k-dimensional triangles. For a few examples of k simplices for different values of k, see Figure 3.1. A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle and a 3-simplex is a solid tetrahedron. Higher dimensional simplicies are the equivalent of higher dimensional triangles. Each simplex is

made up of lower dimensional simplices. For example, the 2-simplex is made up of three 1-simplices and three 0-simplices. These objects will serve as the foundation for our topology-based representations.

Figure 3.1: From left to right: a 0-simplex, a 1-simplex, a 2-simplex, and a 3-simplex.



With k-simplices in mind, we can define a simplicial complex. In this work, we define a simplicial complex as a finite collection of simplices, K, in the real numbers, $\mathbb{R}^n$, such that the following two conditions are satisfied:

C1) If $\sigma \in K$, then every face of σ is in K,

C2) The intersection of any two simplices in K is either empty or a face of each simplex.

To better understand the aforementioned conditions C1-C2, consider the simplicial complex (skeleton or triangularization scheme) in Figure 3.2**B** (see next Section). As an example of C1, each edge in the figure is connected to two vertices, and those vertices are also members of our simplicial complex. We have no edges without endpoints or faces without edges. Condition C2 outlines that all of our simplices either connect via a node, edge, or triangle.

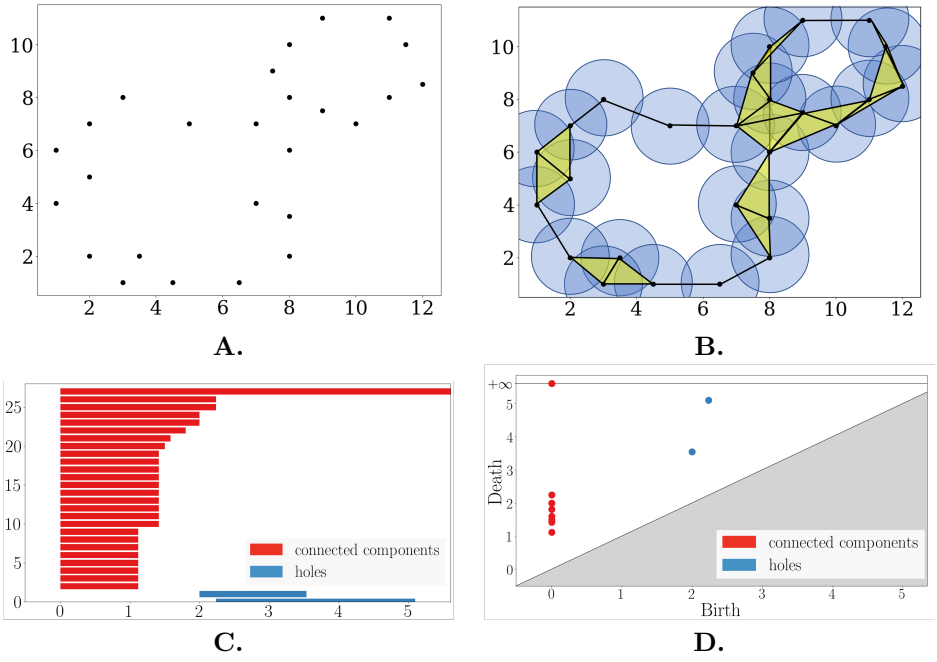
3.3 Persistent Homology

Persistent homology captures the shape of an object across different dimensions and scales. Put simply, homology identifies the number of clusters, or as more commonly called in the topological literature, connected components (0-dimensional homology), holes (1-dimensional homology), voids (2-dimensional homology), etc. of a simplicial complex.

As an example, consider the point cloud data in Figure 3.2**A**. In this instance, the task is to identify the two circular patterns in the dataset. The algorithmic process of generating a simplicial complex starts by considering circles of radius r centered on each data point, and inserting a 1-simplex, i.e. an edge, when two circles overlap. If three circles intersect, we insert a 2-simplex, i.e a triangle. Finally, if k circles overlap, we insert a k-simplex. Indeed, the circles with radius r used in Figure 3.2 capture the structure of our dataset nicely; if we evaluate the homology of this object, we end up with the homology of two circles. But, what if we had chosen a different parameter value, r, for radius? If we chose one too small, the circles would be disconnected and if we chose one too large, there would be no holes in the middle, since all circles would overlap. In other

words, different choices in r values results in different homological features. This is where persistent homology comes in. Instead of choosing a single r value, we look at the homology across a wide range of r values. Hence, we start with $r = 0$ and grow the radius. This allows us to capture the homology and how it changes across various resolutions. Varying, the filtration parameter, the radius in our case, one wants to detect features that persist over a large range of r values. Subsection 3.4 explores ways on how to capture the evolving homology as r varies. It is important to note that the filtration parameter, and its variability in the process, may have physical meaning as it may correspond to atom distances, time, or energy.

Figure 3.2: **A.** A point cloud X in $\mathbb{R}^2$. **B.** The same point cloud with circles of radius r around each ball. Note, we insert a 1-simplex (edge, shown with a thick black line) when two circles overlap and a 2-simplex (triangle, shown as a yellow triangle) when three circles overlap. **C.** The persistence barcode for the point cloud. **D.** The persistence diagram for the point cloud.



3.4 Capturing Persistent Homology

To capture the persistent homology with the help of the filtration parameter, we need tools that allow us to record the birth, death, and dimension of homological features. Persistence barcodes and persistence diagrams are two tools

that capture this information. Each has particular instances that are beneficial for specific applications. Here, we will discuss and analyze both instances and provide examples with respect to chemical applications.

First, we consider persistence barcodes. A persistence barcode assigns a bar to each homological feature. The start of the bar corresponds to the value of the filtration parameter at which the feature is born. The color of the bar denotes the dimension, connected components, and holes. Consider the point cloud consisting of 27 points in the plane given in Figure 3.2**A**. Note, the data seems to fall in two circles to form a shape that looks like a tilted "8", as becomes evident by traversing on the skeleton (edges) in Figure 3.2**B**. We will construct the persistence barcode in Figure 3.2**C** to show how we could capture the underlying structure of the dataset. For this example, our changing filtration parameter is the radius of the circle centered on each point.

At $r = 0$, each data point corresponds to a connected component, resulting in the birth of 27 red bars. Now, as we increase the radius, 0-dimensional bars end whenever two circles merge. To break ties, the latest bar born will die first; hence, the staircase pattern we see on the right-hand side of all of the bars. At around $r = 2$, the circles overlap such that when we add in the 1-simplices, they form a hole (the smaller hole in the upper right corner), which the barcode captures by the birth of a blue bar. At $r = 2.8$, the second circle is born (the larger hole in the bottom left corner). Then, as we keep increasing the radius, the holes are filled in, resulting in their deaths, which naturally terminates the blue bars. Finally, after $r = 6.2$, only one component remains, which we denote with the top bar touching the right-hand side of the figure. Indeed, the only component that survives is this top red bar, which corresponds to the filtration value increasing past the point where all circles overlap.

The length of each bar shows us how long each homological feature *persists*, which is measured by difference between the birth and death of the bar. This means that the longer the bar is, the longer the "lifetime" of a homological feature is. This can be nicely seen in Figure 3.2**C**, where the small, upper right circle has lower persistence, as it is captured by the upper, short blue bar, than the larger circle that is found at the bottom left, which corresponds to a longer bar and thus, to longer persistence.

A persistence diagram captures the same information as a persistence barcode, but displays it in a more concise way. The x-axis corresponds to the birth of the feature and the y-axis corresponds to the death of the feature. It is also common to plot birth vs. persistence. To get how long a feature persists, (shown in the bars by the length of the bar) we simply take the death value and subtract the birth value. For the top bar, which expresses the one single connected component after all circles overlapped, we introduce a line at the top of the diagram and denote it with ∞. Just as in the barcode case, the color of the point denotes the dimension.

For the example in Figure 3.2**A**, we use the persistence barcode in Figure 3.2**C** to construct the persistence diagram. Each point on the persistence diagrams has coordinates $(birth, death)$. For each bar, we plot the birth of the bar on the x-axis and the death of the bar on the y-axis. The length of the

bar translates to (death-birth), which captures how long a feature persists. The persistence can also be computed with respect to the distance from the diagonal. For example, the small circle of Figure 3.2A appears as the blue point on the persistence diagram that is closer to the diagonal. For our semi-infinite bar, we plot a point at $(0, \infty)$ to denote that we have one connected component that lives no matter the scale r. Note, no points can exist below the diagonal as that would mean features would die before they are born.

Below, the code block shows how to produce a persistence diagram and a persistence barcode from a set of points. All barcodes and diagrams were created using GUDHI [55], an open-source C++ library with a documented python interface. Download instructions, documentation, and examples can all be found in [56].

Python Example 6: Generating persistence barcodes and diagrams for a point cloud

```
# Code to generate the persistence barcode and
# persistence diagram for a set of points.

import gudhi
import matplotlib.pyplot as plt

# Import your dataset.
pt_cloud_x = [3,3.5,2,1,2,1,3,4.5,6.5,8,8,7,8,7,
              5,8,8,7.5,9,10,11,12,9,11,11.5]
pt_cloud_y = [1,2,2,4,5,6,8,1,1,2,3.5,4,6,7,7,8,
              10,9,7.5,7,8,8.5,11,11,10]
pt_cloud = zip(pt_cloud_x,pt_cloud_y)

#Use GUDHI to generate the corresponding Rips complex,
rips_complex = gudhi.RipsComplex(points=pt_cloud, max_edge_length=20)
# simplex tree,
simplex_tree = rips_complex.create_simplex_tree(max_dimension=3)
# and persistence data from the point cloud
diag = simplex_tree.persistence(min_persistence=0.4)

# Generate the persistence barcode.
pb = gudhi.plot_persistence_barcode(diag, max_intervals=0,
                                    alpha=1.0,legend=True)
# Generate the persistence diagram.
pd = gudhi.plot_persistence_diagram(diag, max_intervals=0,
                                    alpha=1.0,legend=True)

# Plot the barcode and diagram
plt.show()
```

3.5 Comparing Persistent Homology

A topological summary representation in the form of a persistence diagram allows us to introduce a distance as a measure of shape differences. This idea naturally extends to the distance between two molecular representations.

Suppose you want to compare the shape of two molecules, such as how many holes each molecule possessed. One way to do this would be to calculate the persistent homology of the molecules and compare the two persistence diagrams. The distance between topological features (e.g. two holes) that are found on two different persistence diagrams can be used for comparing the similarity of two molecules. In this section, we will introduce two different ways to compare persistence diagrams, the Wasserstein distance and the Bottleneck distance. These two distance functions are often used due to their stability results, if our molecules have a similar shape, then the persistent homology should be similar. Note, there are other distances, such as the interleaving distance [57], and the d_p^c distance [58, 59] that also can be applied to persistence diagrams.

The *Bottleneck distance* function finds the smallest distance it takes to move the points of one persistence diagram to the points of another persistence diagram. It could be the case that one diagram has more points than the other; if this happens, all leftover points are mapped to the diagonal, the line that corresponds to when *birth* = *death*. This ensures that we have a one-to-one mapping of points for any pairing of persistence diagrams. With these ideas in mind, we define the Bottleneck distance.

Suppose we have two persistence diagrams, call them B and B', and we want to know how close these diagrams are to each other, as shown in Figure 3.3. For computing the Bottleneck distance, consider all possible mappings $\eta: B \to B'$, and record the supremum (the smallest upper bound) of distances of all point pairings for each map η. In other words, record the maximum L_∞ distance between any two points $b \in B$ and $\eta(b) \in B'$. Then, take the infimum (smallest total distance) over all η bijections to calculate the bottleneck distance. For example, Figure 3.3B and Figure 3.3C are two examples of matchings, where B is an optimal matching and C is a less than optimal matching. Mathematically, this corresponds to evaluating

$$W_b(B, B') = \inf_{\eta: B \to B'} \sup_{b \in B} \|b - \eta(b)\|_\infty. \tag{3.1}$$

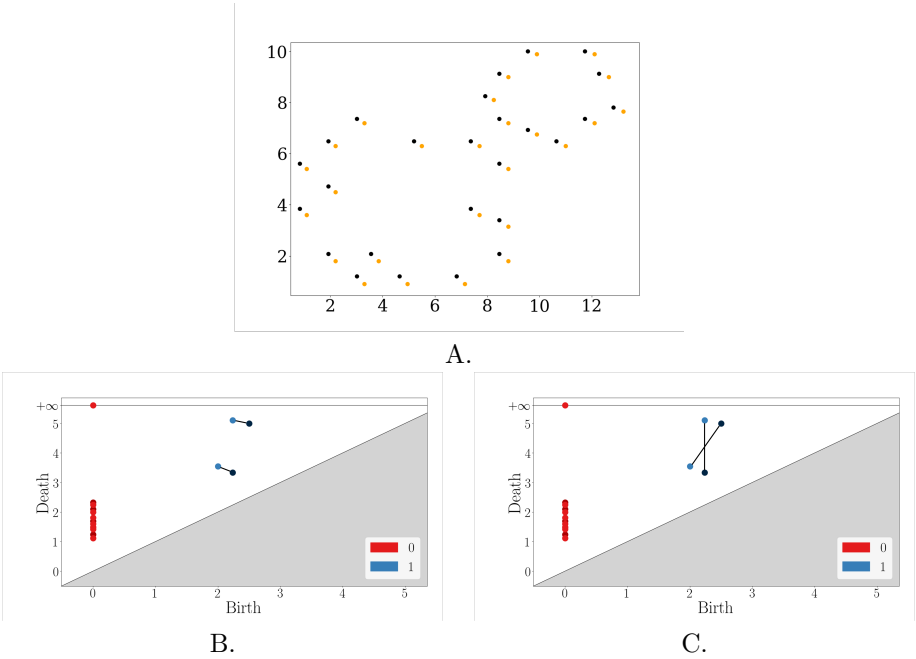
Note, the L_∞ norm, $|v|_\infty$, isolates the maximum magnitude of all vector components. Hence, we get

$$|(v_1, v_2, \ldots v_k)|_\infty = \max(|v_1|, |v_2|, \ldots, |v_k|). \tag{3.2}$$

The bottleneck distance gives a good intuitive understanding of distance. Further, there are stability results for small perturbations of the input data. For the statement and proof of stability, see [60].

As an introduction to the Wasserstein distance, consider the following problem. Suppose you wanted to move a pile of Earth to a different location. The

Figure 3.3: A. The original dataset (Black) and perturbed dataset (Orange). B. and C. The persistence diagram of the original dataset (lighter dots) and the perturbed dataset (darker dots) overlaid. B. An optimal matching of the 1-dimensional features. C. A less than optimal matching of the 1-dimensional features.



amount of effort it would take to move the earth would depend on the amount of earth being moved and the distance it needs to travel. The Wasserstein distance is the mathematical implementation of that concept.

Borrowing notation from the Bottleneck distance, the degree-p Wasserstein distance between two persistence diagrams is given by

$$W_p(B, B') = \left(\sum_{\ell} \inf_{\eta_\ell} \sum_{x} \|x - \eta_\ell(x)\|_\infty^p \right)^{1/p} \tag{3.3}$$

where ℓ is the variable that runs over all homological dimensions, η_ℓ are the bijections between B and B' for the given dimension ℓ, and x is a point in the diagram of B.

3.6 Persistent Homology and Machine Learning

In this section, we focus on methods that will allow us to utilize persistence diagrams as an input to machine learning algorithms for a wide range of applications. To that end, one will need to consider a vectorization of persistence

diagrams. There are various ways to view and vectorize a persistence diagram, from elementary considerations, such as by considering only the most persistent point, to more sophisticated, e.g. probability distributions of persistence diagrams. The reader may refer to [61, 62, 63, 64, 65, 66] and references therein for a gamut of such vectorizations. Below, we will consider persistence images, an up-and-coming method that has performed well on different chemistry applications.

3.6.1 Persistence Images

Persistence images or PIs, developed by Adams et. al.[67], are "a finite-dimensional-vector representation of a persistence diagram." PIs are stable with respect to noise, efficient to compute, understandable with regard to the original persistence diagram, and provide the flexibility to focus on specific types of features, such as medium or high-persisting features. First, we will introduce the mathematics behind PIs and then we will provide examples of their use for molecular representations.[53, 54]

Suppose we have some form of data, such as a point cloud in $\mathbb{R}^n$, and consider its corresponding persistence diagram, B. We break down the PI construction into a series of steps.[67]

1. Convert B from (birth, death) coordinates to (birth, persistence) coordinates.

Let B be the a persistence diagram in terms of birth and death time of a homological feature. Define the function $T(x, y)\colon \mathbb{R}^2 \to \mathbb{R}^2$ such that $T(x, y) = (x, y - x)$. Note, $y - x$ captures the persistence of the feature. Thus, where for each point in B, $T(B)$ gives the birth and persistence coordinates for each feature.

2. Choose a differentiable probability distribution

Consider some differentiable probability distribution $\phi_u\colon \mathbb{R}^2 \to \mathbb{R}$ with mean $u = (u_x, u_y)$. A common choice is to use the Gaussian distribution, given by

$$\phi_u(x, y) = \frac{1}{2\pi\sigma^2} e^{-[(x-u_x)^2+(y-u_y)^2]/2\sigma^2}$$

where σ^2 is the variance. This is used, for example in [67] and [54].

3. Choose a weighting function and construct the persistence surface.

To properly weight the features we want to extract, we must define a function that takes in all points of $T(B)$ and determines each point's significance. This function must be non-negative, zero when $y = 0$, continuous, and piecewise differentiable. We will use this weighting function to define the persistence surface.

First introduced by Donatini et. al [68] and [69] for size functions, the persistence surface of a persistence diagram B is a function $\rho_B\colon \mathbb{R}^2 \to \mathbb{R}$ such

that

$$\rho_B(x,y) = \sum_{u \in T(B)} f(u)\phi_u(x,y),$$

where u is an ordered pair of the birth time and the persistence of $T(B)$.

4. Construct the persistence image.

To obtain a finite-dimensional vector, we discretize the relative portions of the domain and integrate $\rho_B(x,y)$ over each section of the discretization. Thus, compute the double integral

$$\int \int_p \rho_B(x,y)dxdy$$

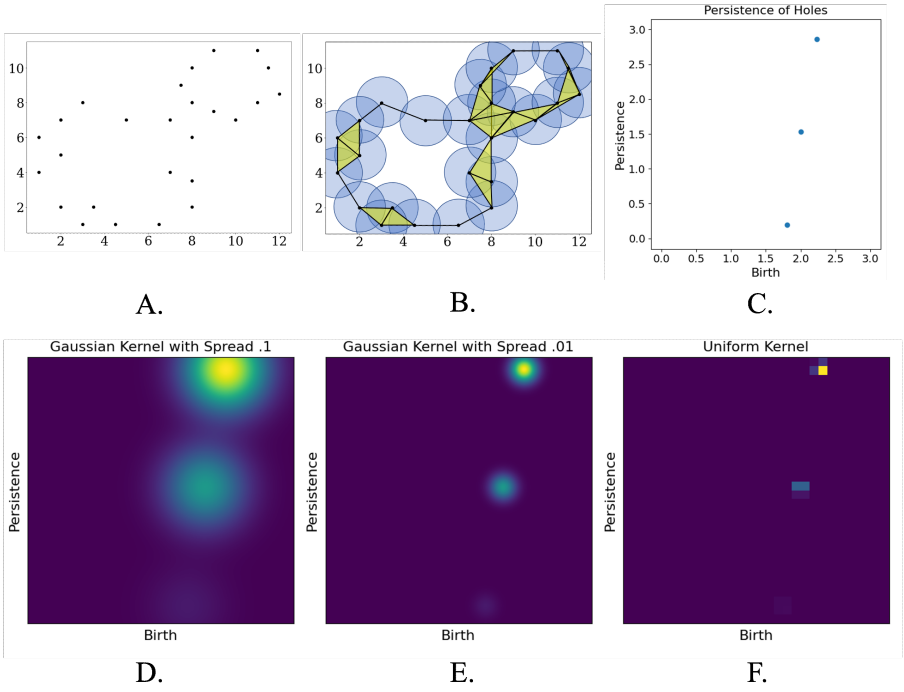
where p is the discretized subdomain. Thus, we get an $m \times n$ grid of values which we transform into a vector by attaching the rows in the natural way,

$$[p_{1,1}, p_{1,2}, \ldots, p_{1,n}, p_{2,1}, p_{2,2}, \ldots, p_{2,n}, \ldots, p_{m,1}, p_{m,2}, \ldots, p_{m,n}].$$

This vector is the persistence image of a given persistence diagram. Note, as it stands, this is the vector for the persistence diagram in one dimension. If there are multiple homological dimensions to consider, simply tack on the other persistence vectors to yield a single vector.

As an example, consider the point cloud introduced in Figure 3.2, shown again in Figure 3.4A. Recall, sublevel set persistence captures the underlying structure of the point cloud by forming simplicial complexes for increasing radius r. Note, Figure 3.4B. contains an example of a simplicial complex for some r. The 1-dimensional persistence diagram in Figure 3.4C. captures the holes in our dataset. From the persistence diagram, we can calculate the persistence image using a variety of probability distributions and variances, see Figure 3.4D.-F. Note, changing the probability distribution changes the persistence image.

Figure 3.4: **A.** The point cloud from Figure 3.2. **B.** An example of a simplicial complex generated via some radius r. **C.** The 1-dimensional persistence diagram in terms of birth and persistence (as opposed to birth and death) for the set of points in A. **D.** The persistence image of the persistence diagram with a Gaussian kernel with spread .1. **E.** The persistence image of the persistence diagram with a Gaussian kernel with spread .01. **F.** The persistence image of the persistence diagram with a uniform kernel.



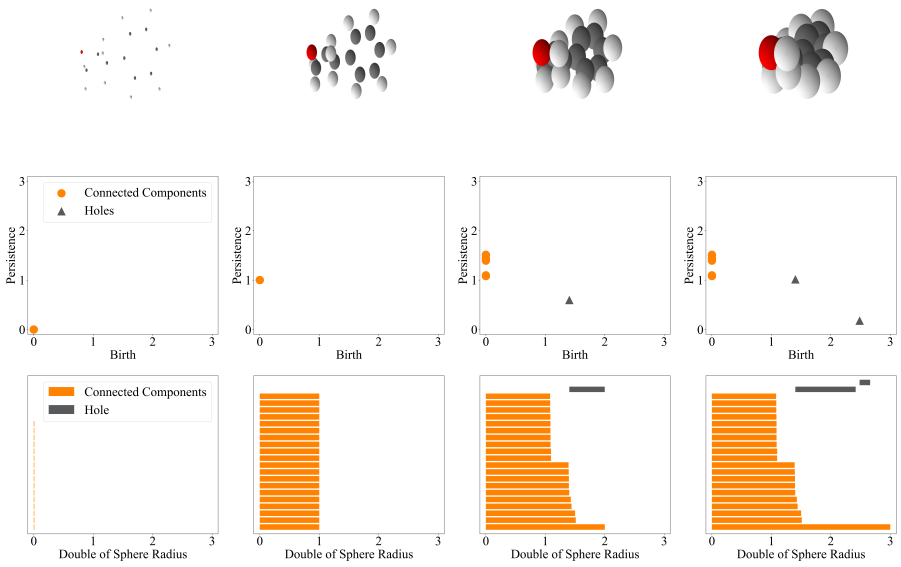
3.6.2 Chemically-driven Persistence Images

In previous sections, we have discussed how persistence diagrams (PDs) and persistence images (PIs) describe topological information of arbitrary points in Euclidean space. Since one aim of molecular representations is to encode the topology of chemical structures, it becomes evident that PDs/PIs can be also employed for that purpose. Indeed, over the past few years, many computational chemists and material scientists have started exploring the unique features of these vectorizations. In this section, we discuss how PDs/PIs can be generated from a molecular structure, and how these vectorizations can be used for machine learning applications. Since PDs are the first step toward generating chemically-driven PIs, we will first explore the properties of a PD for our example molecule, 2-benzyloxirane (Figure 3.5).

First, we will consider each atom as a point in the Euclidean space defined

by its (x, y, z) coordinates, and we will place spheres of radius r centered at each atom. The radius is the filtration parameter and as the radii of each sphere increases with the same rate, they intersect and lead to the evolution of homological features (connected components and holes). In this case, the connected components encode interatomic distances, while holes describe molecular attributes such as ring structures and functional groups.

Figure 3.5: An example of how the persistence diagram (PD) and persistence barcodes for 2-benzyloxirane are generated for a radius of 0, 1, 2, and 3 angstroms (where the algorithm terminates).



Let us see how this is achieved by analyzing the example in Fig. 3.5. The evolution of these homological features is shown in this example (from left to right) as they are plotted in a persistence diagram (middle) and a persistence barcode (bottom). Notice, the y-axis of the PD gives the persistence, which is defined as the difference between death and birth of a homological feature. At radius $r = 0$, we only see zeroth-order homology features at *birth* = persistence = 0, while all bars begin at point 0. As the radii values increase from 0 to 1, the barcodes start to expand and the connected components of the PD *persist*.

Recall, the intersection of two spheres leads to the death of a connected component. This information is introduced in the PD as a point born at *birth*=0 and *persistence*≈1.1. Hence, these features have stopped persisting, as they have merged with other component(s) to form a cluster. For example, in the third column of Figure 3.5, we clearly see points at $(0, 1.1)$ coordinates, which correspond to the overlap between carbon and hydrogen atoms. In other words, C-H

bonds are encoded in the PD, and since there are more than one C-H bonds in 2-benzyloxirane, a "stack" of points is introduced at the $(0, 1.1)$ coordinates. It also becomes evident that the units of the PD are not arbitrary, but they correspond to atom length units (e.g. Å). The same information is also introduced in the persistence barcode since bars end at 1.1 Å. Similarly, information related to C-C and C-O bonds is introduced in both plots.

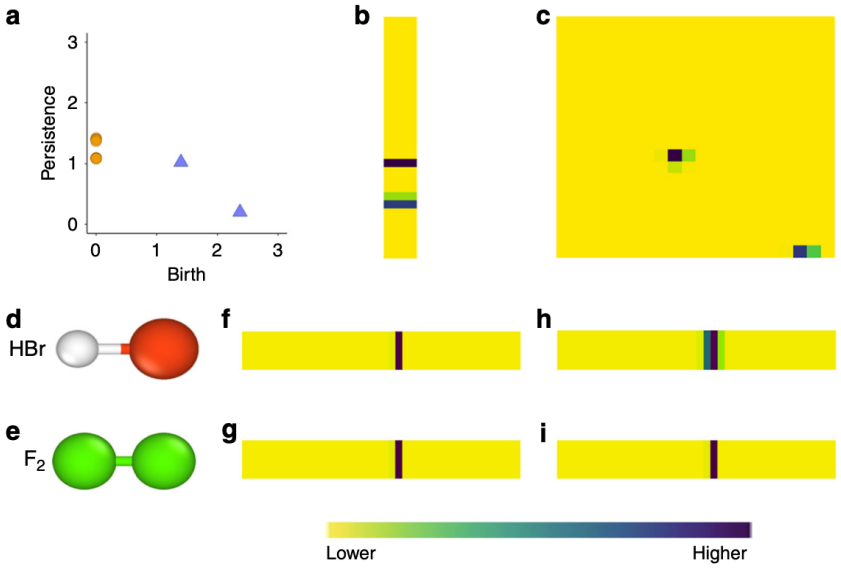
At about r = 1.4Å, we see the birth of the first "hole" (first-order homological feature), which is an outcome of the overlap of the six spheres placed on the carbon atoms of the phenyl group (shown as a triangle in Figure 3.5). As the sphere's radii continue to increase, the first-order hole continues to persist, and the length of the corresponding bar increases (grey bar in the bottom of Figure 3.5). The death of this homological feature is reached when the Euclidean space contained in the phenyl ring is completely covered by the spheres.

Finally, a second hole appears in the PD with short persistence. The birth of the second hole corresponds to the H-C_p-C_p-C-C_e-H chain, where C_p are the carbon atoms that belong to the phenyl group of 2-benzyloxarine, and C_e to the carbon that is part of the epoxide. The small persistence value results from a brief hole appearing before all spheres merge to form one connected component. This naturally ends the algorithm.

We should mention that the three-atom epoxide does not produce a hole since when the three spheres overlap, they have completely covered the Euclidean space within the three-member ring and thus, birth and death coincide. Since these two functional groups have completely different coordinates in the PD, we can consider these points in the PD as their "fingerprint". Indeed, for any six-atom planar group, the birth coordinate of the hole is around 1.4 Å, while the persistence value varies based on the composition of these groups, since subtle structural changes will affect the coordinates of the homological features. For example, a phenyl group (C_6H_5) has larger persistence than a pyridyl group (C_5NH_4), since substitution of CH by a nitrogen atom shrinks the hexagonal structure of the phenyl which, consequently, affects the (*birth*, *persistence*) or (*birth*, *death*) coordinates.

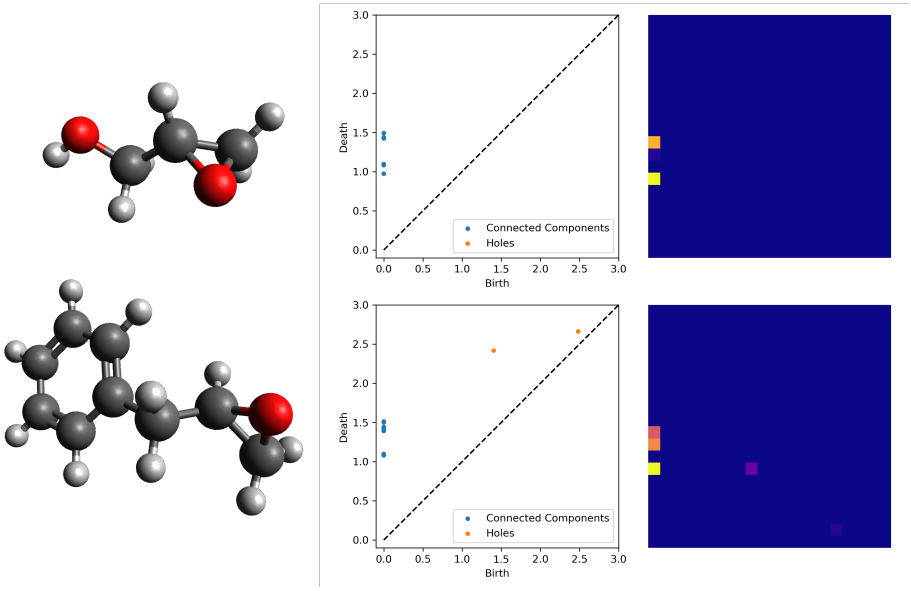
Once the PD is generated, it can be vectorized into a PI. For example, the connected components and holes of the PD of Fig. 3.6a are converted into non-zero values in the PI, as shown in Fig. 3.6b and 3.6c, respectively. Based on the above, it becomes evident that a conventional PI cannot differentiate between bonds of similar length, since the radii r of all atoms increase at the same rate. For example, the persistence images of the diatomic molecules HBr and F_2 (Fig. 3.6d and 3.6e, respectively) are identical since the interatomic distance is the same (1.4 Å). To differentiate between bonds of similar length but varying composition, a metric that depends on the atomic types is introduced. For example, the electronegativity difference between an atom pair can be added into the variance of an additive Gaussian placed on each connected component of a PI[53]. This addition allows for differentiation between different homonuclear and heteronuclear bonds (e.g., Fig. 3.6h and 3.6i for HBr and F_2, respectively).

Figure 3.6: The persistence diagram of a molecule (a) is converted into a persistence image, which includes the connected components (b) and holes (c). The molecules (d) HBr and (e) F_2 have the same bond length, resulting in equivalent features on the persistence diagram. Without incorporating the chemically driven PI, the representations in (f) and (g) are equivalent. By incorporating the pairwise electronegativity difference into the variance of the Gaussian kernel, the input vectors (h) and (i) for the two molecules are distinguished by the variance of the vector for HBr.



In our code example, we will generate the PDs and PIs of our sample molecules, as shown in Figure 3.7. These two molecules offer interesting test cases since glycidol does not have any holes and 2-benzyloxirane does.

Figure 3.7: An example persistence diagram (PD) and persistence image (PI) of glycidol and 2-benzyloxirane. Note the absence of holes in the PD and PI of glycidol.



Python Example 7: Chemically-Driven Persistence Images

```
# Chemically driven persistence homology example
# Required files to be in same directory:
#    -Element_PI.py
#    -elements.py
# From Element_PI.py import PersDiagram to generate
#    a persistence diagram (PD)
from Element_PI import PersDiagram
# From Element_PI.py import VariancePersist to generate
#    a persistence image (PI)
from Element_PI import VariancePersist as VP
import matplotlib.pyplot as plt

test_file1='glycidol.xyz'
test_file2='2-benzyloxirane.xyz'

# We will create a four panel figure with a PD and PI both
#    sample molecules
plt.figure(figsize=(8, 8))

# Create subfigures for glycidol
plt.subplot(221)
PersDiagram(test_file1,lifetime=False)
plt.legend(loc=4)
plt.subplot(222)
VP(test_file1,pixelx=20, pixely=20, myspread=0.1,
   myspecs={"maxBD": 3, "minBD":0},showplot=False)

# Create subfigures for 2-benzyloxirane
plt.subplot(223)
PersDiagram(test_file2,lifetime=False)
plt.legend(loc=4)
plt.subplot(224)
VP(test_file2,pixelx=20, pixely=20, myspread=0.1,
   myspecs={"maxBD": 3, "minBD":0},showplot=False)
plt.tight_layout()
plt.show()
```

3.7 That's a Wrap

- Persistent homology can be used to capture the structure (connected components, holes, voids, etc.) of molecules.
- Persistence barcodes and persistence diagrams record the persistence information, the foundation of persistence-based molecular representations

for machine learning.

- The Bottleneck distance and Wasserstein distance are two ways to compare persistence diagrams and hence, compare molecular representations.
- Persistence images are one way to utilize persistence-based representations of molecules for Machine Learning applications.
- Additional resources on persistent homology:
  - Computational topology: an introduction, Herbert Edelsbrunner and John L Harer. **2008**, American Mathematical Society.
  - L. Wasserman. Topological Data Analysis. *Annu. Rev. Stat. Appl.*, **2018**, *5*, 501.
  - Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Probl.*, **2011**, *27*, 124007.

Chapter 4

Physics-Based Representations

4.1 Introduction

The physics-based representation is the last family of molecular representations discussed in this eBook. These methodologies have emerged as a result of efforts to bridge high-dimensional physical properties such as the molecular wave function to low-dimensional and more compact vectorizations. In physics-informed machine learning, the physical laws of the problem under consideration are included as part of the model, as it happens for example in kernel-based regression. For molecular systems, enforcing the physical laws of symmetry or electronic structure as the input molecular representation offers additional advantages on the discovery of relationships between molecular structures at the atomic level and their properties. Thus, the machine learning algorithm is able to learn more effectively since it connects patterns from the underlying structural information parsed into the model with molecular energies or properties. Such physics-based geometric and electronic structure information span from the pairwise electrostatic interactions between two atoms in a molecule, and can go as deep as properties obtained from the electronic wave function (e.g. electron correlation). These types of representations have helped the field of computational chemistry and chemoinformatics to build *transferable* and *interpretable* machine learning models for a variety of applications, including the exploration of the chemical space for the discovery of molecules with enhanced properties and the development of accurate force fields for large-scale molecular simulations.

In this chapter, the most widely-applied physics-based representations are presented. We chose not to list them in chronological order, but in order of relevance and increasing complexity. We begin with the conceptually simplest method, the Coulomb Matrix (CM), followed by its derivatives that offer extensions and solutions to the inherit problems of CMs. Then, we move to

atom-centered symmetry functions which introduce a systematic expansion of convergent terms of increasing complexity and other spectral representations, and we close this Chapter with *ab initio* or "first principles" representations that are based on known quantum mechanical and quantum chemical relations, functions, and expansions.

4.2 Coulomb Matrices and Derivatives

In this section, we discuss Coulomb matrices (CMs) and methods that are either derived from CMs or are related to CMs. The main motivation behind our choice to begin this Chapter with CMs is that these molecular representations are generally easy to compute and offer interpretable input to ML models. First, we will cover CMs and bag of bonds (BoBs), which a student with a basic understanding of introductory physics can easily compute by hand, and move towards more advanced topics such as the many-body tensor representation (MBTR). Code examples for every method in this section are provided at the end of the subchapter.

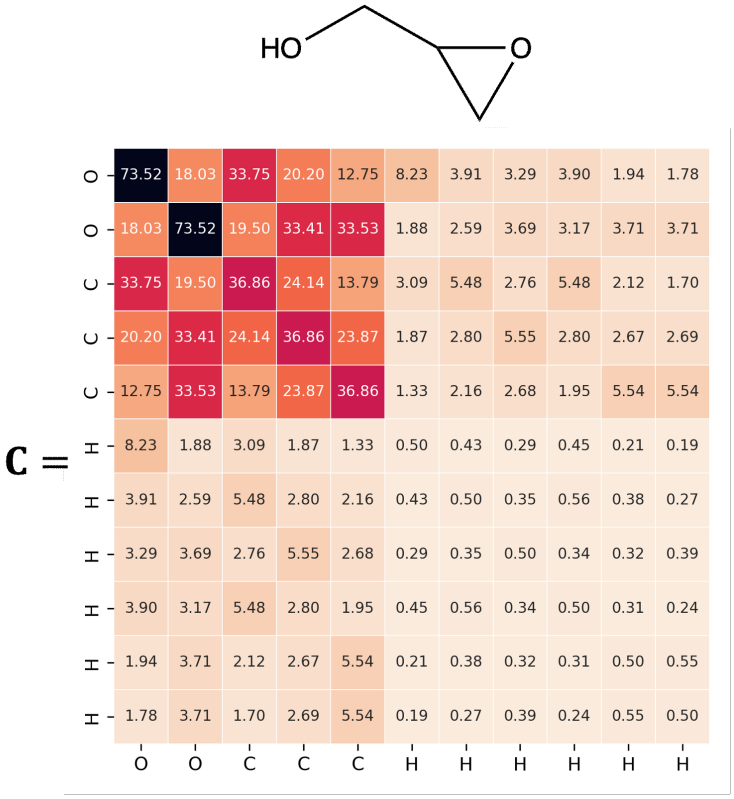
4.2.1 Coulomb Matrix

One of the most straightforward physics-based molecular representations are the Coulomb matrices (CMs), first introduced by Rupp *et al.*[70] to predict atomization energies of organic molecules. CMs are square matrices with dimensions $N_{\text{atoms}} \times N_{\text{atoms}}$, where N_{atoms} is the number of atoms of a given molecule, which hold information related to the atom type (diagonal elements) and the pairwise atomic interaction (off-diagonal elements). The construction of a CM requires a set of nuclear charges $\{Z_i\}$ and the Cartesian coordinates of the atomic positions, $\{\mathbf{R}_i\}$. The Coulomb matrix, $\mathbf{x}^{\text{Coulomb}}$, is then defined as

$$x_{ij}^{\text{Coulomb}} = \begin{cases} 0.5 Z_i^{2.4}, & \forall i = j \\ \frac{Z_i Z_j}{\|\mathbf{R}_i - \mathbf{R}_j\|_2}, & \forall i \neq j \end{cases} \tag{4.1}$$

where the off-diagonal elements correspond to the Coulomb repulsion between atoms i and j and the diagonal elements correspond to a polynomial fit of the nuclear charges to the total atomic energies. In the equation above, $\|\mathbf{R}_i - \mathbf{R}_j\|_2$ denotes the Euclidean distance, or l_2-norm, between the atomic coordinates of atoms i and j. Figure 4.1 shows an example of a CM for glycidol. The two dominant values of 73.52 correspond to oxygen's self-interaction ($0.5 \cdot 8^{2.4} = 73.52$). The second largest value considers the self-interaction of carbon with a value of 36.86. The largest off-diagonal element, $((6 \cdot 8)/1.42 = 33.75)$, corresponds to a C-O bond between the hydroxyl group and the carbon atom that bridges the hydroxyl and epoxy group.

Figure 4.1: A sorted Coulomb Matrix for glycidol.



While CMs are invariant to rotations, translation, and symmetry operations with respect to the total energy, the CM representation has two important drawbacks. The first is related to their dimensionality since CMs of molecules of varying size will have matrices with varying dimensions. The CM of a molecule composed by 10 atoms will have 10×10 dimension, while for a 100-atom molecule, the size will be 100×100. To remedy this issue and to offer same-size representations, CMs are padded with "dummy atoms" and matrix elements with zero as a value to represent an atom with zero nuclear charge and no interaction. This is typically done in order to match the dimensions of the largest molecules.

The second drawback is related to the permutational invariance of the CM. For a given molecule, there are $N_{atoms}!$ ways to permute the columns and rows of the CM without effecting the energy of the molecule. While this ambiguity has no obvious solution, three suggestions that circumvent this issue are the eigenspectrum representation, the sorted CMs, and the randomly sorted CMs.[71] The *extended Coulomb matrix* representation is an extension of CMs to periodic systems, where the electrostatic interaction between an atom in a unit cell with the atoms of the neighboring units cells is considered. Another extension of the Coulomb matrix is the *Sine matrix* representation, which only depends

on the positions of atoms in a single unit cell. The sums of the electrostatic interaction are replaced with an arbitrary two-point potential, $\tilde{\Phi}(\mathbf{R}_i, \mathbf{R}_j)$,

$$x_{ij}^{Sine} = \begin{cases} 0.5 Z_i^{2.4} \text{ if } i = j \\ Z_i Z_j \tilde{\Phi}(\mathbf{R}_i, \mathbf{R}_j) \text{ if } i \neq j, \end{cases} \tag{4.2}$$

where $\mathbf{R}_l$ is the l^{th} atomic coordinate in the unit cell.

Python Example 8: Coulomb Matrices

```
# An example of how to generate a Coulomb matrices (CM)
# for glycidol using DScribe and Atomic Simulation Environment (ASE)
#
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
from dscribe.descriptors import CoulombMatrix
from ase.build import molecule
from ase.io import read
# This code block shows the default parameters for generating CMs
# in DScribe. By default, a sorted CM is generate with L2-norm
# sorted rows and columns. n_atoms_max is a parameter that helps
# with padding, for one molecule it is the number of atoms in
# the molecule.
print('Sorted CM')
cm = CoulombMatrix(n_atoms_max=11)

# Generate the ethanol molecule using ASE

mol = read('../glycidol.xyz')
print(type(mol))

# Create CM output for the system
cm_mol = cm.create(mol)

# Print the sorted CM and it's corresponding shape
print("Flattened shape of the sorted CM", cm_mol.shape)
print(cm_mol)

# Set the parameters, sigma and seed, of the unsorted CM
print('\nUnsorted CM')
cm_unsrtd=CoulombMatrix(n_atoms_max=11,permutation='none')

# Generate the unsorted CM for mol
cm_unsrtd_mol = cm_unsrtd.create(mol)

# Print the unsorted CM and it's corresponding shape
print("Flattened shape of the unsorted CM", cm_unsrtd_mol.shape)
print(cm_unsrtd_mol)

# Set the parameters of the eigenspectrum representation
print('\nEigenspectrum')
cm_eigen=CoulombMatrix(n_atoms_max=11,permutation='eigenspectrum')

# Generate the eigenspectrum representation of the CM
cm_eigen_mol = cm_eigen.create(mol)

# Print the eigenspectrum and it's corresponding shape
print("Flattened shape of the eigenspectrum", cm_eigen_mol.shape)
print(cm_eigen_mol)

# Set the parameters, sigma and seed, of the randomly sorted CM
# Examine how sigma effects the sorting of the randomly sorted CM
print('\nRandomly sorted CM')
cm_random=CoulombMatrix(n_atoms_max=11,permutation='random',
                        sigma=1e-3,seed=42)
```

```
# Generate the randomly sorted CM for mol
cm_random_mol = cm_random.create(mol)

# Print the randomly sorted CM and it's corresponding shape
print("Flattened shape of the randomly sorted CM",
      cm_random_mol.shape)
print(cm_random_mol)
```

4.2.2 Bag of Bonds (BoBs)

The bag of bonds (BoBs) descriptor builds on common ML descriptors such as CMs (Section 4.2.1) and bag-of-words.[72] The bag-of-words descriptor is used in natural language processing and classification applications to encode the frequency of words in a given text. The BoBs descriptor uses the same principle to group ("bag") bonds of a particular type (i.e. C-C, C-N, N-H, etc.), where each entry in every bag is computed using the off-diagonal Coulomb matrix formula, Eq. 4.1, as shown in Figure 4.2 (b) and (c). A vectorizable, permutationally invariant input is then formed by concatenating all bags of bonds in a specific order (Figure 4.2 (d)) with padding between each bag.

We will discuss an example of the BoBs representation related to the development of a ML model for predicting molecular energies. For a given molecule (Figure 4.2 (a)) represented by the BoBs vector $\mathbf{M}$ (Figure 4.2 (d)), the energy, $\hat{E}_{\text{BoBs}}$, takes the form of the predicted value from kernel ridge regression, where the sum of regression coefficients, α_I, and a kernel, $k(\mathbf{M}, \mathbf{M}_I)$, centered on a training molecule, I, are defined as

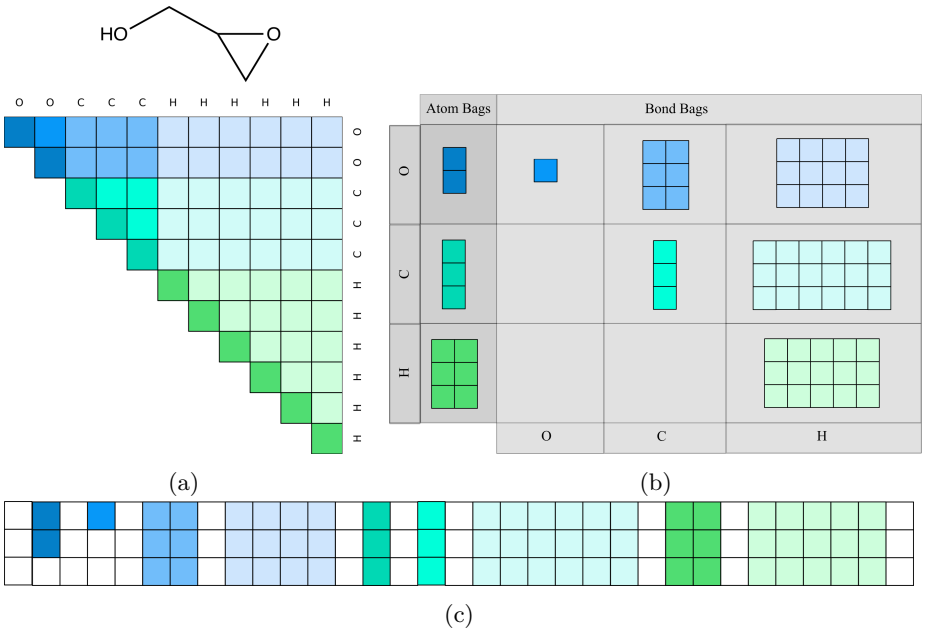
$$\hat{E}_{\text{BoBs}}(\mathbf{M}) = \sum_{I=1}^{N} \alpha_I k(\mathbf{M}, \mathbf{M}_I). \tag{4.3}$$

The BoB kernel is then defined as,

$$k(\mathbf{M}, \mathbf{M}_I) = \exp\left(-\frac{d(\mathbf{M}, \mathbf{M}_I)}{\sigma}\right), \tag{4.4}$$

where the kernel metric, $d(\mathbf{M}, \mathbf{M}_I) = \sum_j \| M^j - M_I^j \|_p$, is the l_p-norm (or distance) between BoB vectors $\mathbf{M}$ and $\mathbf{M}_I$ and σ is the kernel width. In theory, any kernel used in kernel ridge regression can be used in the BoBs representation but the original BoB formulation uses a Laplacian kernel ($p = 1$) due to improved performance over a Gaussian kernel ($p = 2$) on the GDB-7 database. To obtain bags of equal size for all molecules in a given data set, bags are padded with zeros, as it was discussed for CMs as well.

Figure 4.2: The Bag of Bonds (BoBss) molecular representation for glycidol. (a) The upper-diagonal elements of the Coulomb matrix for glycidol. (b) Bags grouped by atom type (O, C, H) and by atom pairs (OO, OC, OH, CC, CH, and HH). (c) The BoBs representation shown as an array, instead of a vector, for visualization choices. In the BoBs representation, zeroes are often introduced for padding to make the representation the same size between molecules of variable size. In (c), zeroes are shown as white squares.



An extension of the BoBs representation, the *BA-representation*, expands the use of bags to include atoms, bonds, angles, and torsions.[73] The hierarchy of bags used are (1) dressed atoms ($\mathbf{M}^{\mathrm{D}}$), (2) atoms and bonds ($\mathbf{M}^{\mathrm{B}}$), (3) atoms, bonds, and angles ($\mathbf{M}^{\mathrm{A}}$), and (4) atoms, bonds, angles, and torsions ($\mathbf{M}^{\mathrm{T}}$). Along with the BA-representation, they offer a reformulated CM, using scaled, pairwise London dispersion interactions, called the *London Matrix* (LMs). LMs were developed, based on improvements found when changing from Coulomb $1/R$ to van der Waals $1/R^6$ power laws, to model molecular atomization energies.

Python Example 9: Bag of Bonds

```
# Bag of Bonds (BoBs) example using QML
from qml import representations
import os
from collections import Counter
# Create the QML compound using the glycidol xyz file
mol = qml.Compound(xyz="glycidol.xyz")

# Generate the BoB vector for glycidol using Counter to create
# a dictionary with the number of each atom type
num_atoms = Counter(mol.atomtypes)
print(num_atoms)
mol.generate_BoBs(asize=num_atoms)
print(mol.representation)
```

4.2.3 Many-Body Tensor Representation (MBTR)

The many-body tensor representation (MBTR) is based on a distribution of atomic terms organized by elements and is invariant to translations, rotations, and permutations.[74] The MBTR provides a unique, differentiable descriptor for both molecules and crystals, and builds on established methods covered in the previous sections, such as CMs and BoBs, while incorporating features derived from many-body expansions. When molecular representations are used in force-field based ML methods, the differentiability plays an important role since the derivative of energy, with respect to the atomic coordinates, is the force.

In the MBTR representation, for a molecule $\mathcal{M}$, the BoBs tensor is rewritten as,

$$f_{\text{BoBs}}(x, z_1, z_2) = \sum_{i,j=1}^{N_{\text{atoms}}} \delta(x - d_{i,j}^{-1})\delta(z_1, Z_i)\delta(z_2, Z_j), \tag{4.5}$$

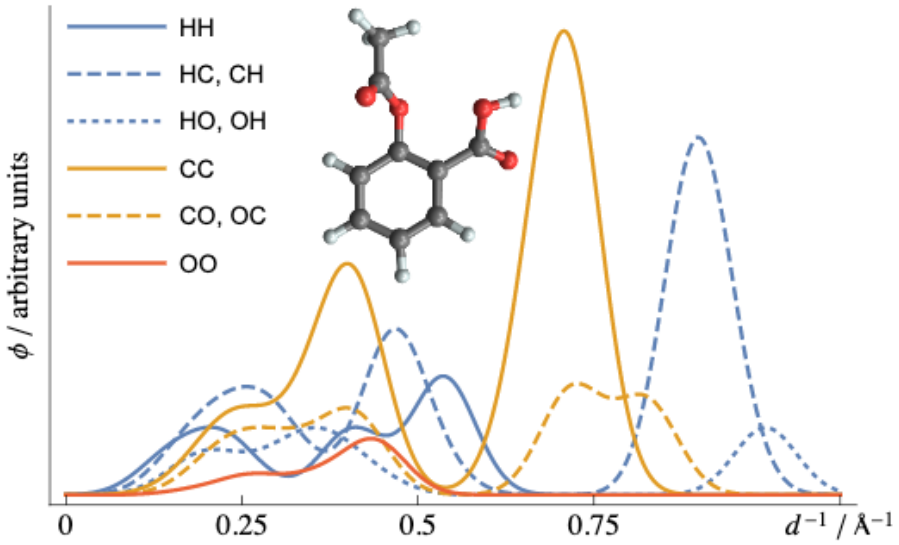
where the dimensions of $f_{\text{BoBs}}(x, z_1, z_2)$ is equal to the $(N_{\text{atoms}} \times N_{\text{atoms}} \times p)$, where p denotes the amount of padding needed in the tensor. The Dirac delta and Kronecker delta are denoted by $\delta(\cdot)$ and $\delta(\cdot, \cdot)$, respectively. The sorting step of the BoBs representation is removed by mapping the Euclidean distance between atoms i and i, $d_{i,j} = \|\mathbf{R}_i - \mathbf{R}_j\|_2$, to the real axis x, using the Dirac delta. In MBTR, the arrangement of the bags by elements is still used.

As an intermediate step, we will show the two-body tensor, which is a a special case of the BoBs tensor, where the Dirac delta is replaced with a smoother probability distribution and a weighting function w_2 and a correlation matrix between atoms ($C \in \boldsymbol{R}^{N_{\text{atoms}} \times N_{\text{atoms}}}$) is introduced. The two-body tensor is,

$$f_2(x, z_1, z_2) = \sum_{i,j=1}^{N_{\text{atoms}}} w_2(i, j)\mathcal{D}(x, g_2(i, j))C_{z_1,Z_i}C_{z_2,Z_j}, \tag{4.6}$$

where the Kronecker deltas in Eq. 4.5 are replaced by the element correlation matrix. g_2 describes the relationship between atoms using the inverse distances between two atoms i and j. An example of the two-body tensor for aspirin is shown in Figure 4.3.

Figure 4.3: One key feature of the many-body tensor representation (MBTR) is that elements of belonging to the representation can be visualized. The left panel shows the distributions of inverse distances ($k = 2$, quadratic weighting) for aspirin ($C_9O_4H_8$). *Reproduced from H. Huo and M. Rupp. Unified representation of molecules and crystals for machine learning. arXiv, 4 2017.*



Using the notation introduced for the simple case of the two-body tensor, the g_2 term is reformulated to a more general term g_k that introduces features commonly found in many-body expansions such as atom counts ($k = 1$), (inverse) distances ($k = 2$), angles ($k = 3$), and dihedral angles ($k = 4$). The general many-body tensor is then defined as,

$$f_k(x, z) = \sum_{i=1}^{N_{\text{atoms}}} w_k(i) \mathcal{D}(x, g_k(i)) \prod_{j=1}^{k} C_{z_j, Z_{i_j}}. \tag{4.7}$$

In the following example, we use DScribe to obtain the $k = 1, 2,$ and 3 terms for glycidol.

Python Example 10: Many-Body Tensor Representation

```
# Example of generating a many-body tensor representation
# (MBTRs) representation for glycidol using DScribe and
# the Atomic Simulation Environment (ASE)
#
from dscribe.descriptors import MBTR
from ase.io import read

# Set up the MBTR descriptor with parameters:
# species, rcut, nmax, and lmax
mbtr = MBTR(species=["H", "O","C"],
    k1={
        "geometry": {"function": "atomic_number"},
        "grid": {"min": 0, "max": 16, "n": 100, "sigma": 0.1},
    },
    k2={
        "geometry": {"function": "inverse_distance"},
        "grid": {"min": 0, "max": 1, "n": 100, "sigma": 0.1},
        "weighting": {"function": "exp", "scale": 0.5,
                      "threshold": 1e-3},
    },
    k3 = {
        "geometry": {"function": "angle"},
        "grid": {"min": 0, "max": 180, "sigma": 5, "n": 50},
        "weighting" : {"function": "exp", "r_cut": 10,
                       "threshold": 1e-3}
    },
    periodic=False,
    normalization="l2_each",)

# Import the glycidol xyz
mol = read('../glycidol.xyz')
print(type(mol))

# Create MBTR representation of glycidol
mbtr_mol = mbtr.create(mol)

# Print the MBTR representation for glycidol and the shape of the
# feature vector
print(mbtr_mol)
print('Shape of the MBTR representation of glycidol',
      mbtr_mol.shape)
```

Available Software Packages

Packages with relevant representations covered in Section 4.2:

- ChemML
  - Coulomb Matrix
  - Bag of Bonds
- DScribe
  - Coulomb Matrix
  - Ewald Sum Matrix
  - Sine Matrix
  - Many-Body Tensor Representation
- QML
  - Coulomb Matrix
  - Bag of Bonds

4.3 Atom-centered Symmetry Functions

An alternative approach for the construction of physics-informed molecular representations is the introduction of a series of functions placed on atomic positions. These functions, called Atom-centered Symmetry Functions (ACSFs), are combined into multiple two- and three-body terms and aim to capture the short-range atomic interactions of a molecule. Some of the attractive advantages of ACFSs are that they are based on known mathematical expansions that describe the geometric and electronic structure of the local environment near an atom, they are systematically improvable, and they can be modified to represent molecular properties and other physical effects for specific chemical applications. Some of the most popular ACSFs are discussed in this section. Our discussion begins with the Behler-Parrinello functions and then explores other representations that have been extensively applied in previous years for ML-based studies of molecules and materials.

4.3.1 Behler-Parrinello Atom-centered Symmetry Functions (ACSFs)

Behler and Parrinello proposed the first general ACSFs, which built on the previous work of Blank *et al.*[75] and Lorenz *et al.*[76], which include a novel set of atom-centered radial and angular symmetry functions. These ACSFs incorporate permutational invariance to the representation that previous ACSFs

lacked. The symmetry functions of each atom must also be rotationally and translationally invariant and reflect the local environment that determines the atoms energy. To maintain generality, since coordination numbers of atoms can change during the course of a molecular mechanics simulation, the symmetry functions must be independent of the coordination of the atom.[77]

The first step of this scheme is to transform the Cartesian coordinates α of atom i, $\{R_i^\alpha\}$, into symmetry functions, $\{G_i^\mu\}$. The local environment can be enforced by the cutoff function,

$$f_c(R_{ij}) = \begin{cases} 0.5 \times \left[\cos\left(\frac{\pi R_{ij}}{R_c}\right) + 1\right] & \text{for } R_{ij} \leq R_c \\ 0 & \text{for } R_{ij} > R_c, \end{cases} \tag{4.8}$$

where R_{ij} is defined as the interatomic distance and R_c is the cutoff value. The function $f_c(R_{ij})$ has a slope and value of zero at $R_{ij} > R_c$. The cutoff functions are used to ensure that the total symmetry functions decay to zero in value and slope at the cutoff radius and to eliminate atoms beyond the cutoff radius from contributing to the energy of the atom centered at i. This cutoff function is incorporated into all of the following ACSFs.[78]

First, we will discuss the radial symmetry functions, which are constructed as sums of two-body terms and describe the coordination of a central atom at varying distances. The radial symmetry functions can also be considered an effective coordination number. The first radial function,

$$G_i^1 = \sum_j f_c(R_{ij}), \tag{4.9}$$

is the sum of cutoff functions with respect to all neighboring atoms j. The second radial function,

$$G_i^2 = \sum_j \exp\left(-\eta(R_{ij} - R_s)^2\right) f_c(R_{ij}), \tag{4.10}$$

is a sum of Gaussians multiplied by the cutoff function with parameters η, related to the width of the Gaussian and R_s, a radial distance that can shift the center of the Gaussian. The third radial function represents a damped cosine function,

$$G_i^3 = \sum_j \cos\left(\kappa R_{ij}\right) \cdot f_c(R_{ij}), \tag{4.11}$$

with period length κ.

Next, we will discuss the angular symmetry functions, which describe the angular environments of atom i using the two functions, G_i^4 and G_i^5. Both functions are formulated using the angles $\theta_{ijk} = \arccos\left(\mathbf{R}_{ij} \cdot \mathbf{R}_{ik} / R_{ij} \cdot R_{ik}\right)$, where the first angular symmetry function is defined as

$$\begin{aligned} G_i^4 = 2^{1-\zeta} \sum_{j,k \neq i} &\left(1 + \lambda \cos\left(\theta_{ijk}\right)\right)^\zeta \cdot \\ &\exp\left(-\eta(R_{ij}^2 + R_{ik}^2 + R_{jk}^2)\right) \cdot f_c(R_{ij}) \cdot f_c(R_{ik}) \cdot f_c(R_{jk}) \end{aligned} \tag{4.12}$$

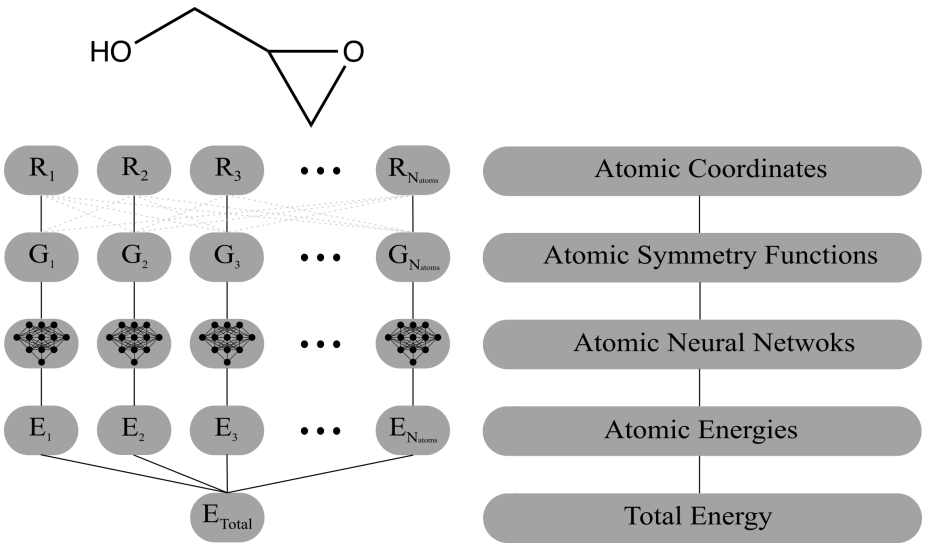
and the second as,

$$G_i^5 = 2^{1-\zeta} \sum_{j,k \neq i} (1 + \lambda \cos(\theta_{ijk}))^{\zeta} \cdot \exp(-\eta(R_{ij}^2 + R_{ik}^2)) \cdot f_c(R_{ij}) \cdot f_c(R_{ik}). \quad (4.13)$$

The parameters for Eq. 4.12 and 4.13 include λ, which shifts the maxima of the cosine function to $\theta_{ijk} = 0°$ ($\lambda = +1$) or $\theta_{ijk} = 180°$ ($\lambda = -1$), and ζ, which provides angular resolution.

Figure 4.4 shows an example of the neural network incorporated in the Behler-Parrinello ACSFs scheme.

Figure 4.4: An example of the Behler-Parrinello neural network. The first layer takes the atomic coordinates, $\mathbf{R} = \{R_1, \ldots, R_{N_{atoms}}\}$, and transforms them into symmetry functions, $\mathbf{G} = \{G_1, \ldots, G_{N_{atoms}}\}$. Each atom has a corresponding neural network to learn the atomic energy, which is then summed to recover the total energy.



In the following example, we generate ACSFs using DScribe and the Atomic Simulation Environment (ASE).

Python Example 11: Atom-Centered Symmetry Functions (ACSFs)

```
# Example of the atom-centered symmetry functions (ACSFs)
# representation of glycidol using DScribe and the Atomic
# Simulation Environment (ASE)
#
from dscribe.descriptors import ACSF
from ase.io import read

# Import the glycidol xyz
mol = read('../glycidol.xyz')
print(type(mol))

# Set up the ACSF descriptor
acsf = ACSF(species=["H", "O","C"],rcut=6.0)

# Create the glycidol ACSFs
acsf_mol = acsf.create(mol)

# Print the ACSFs for glycidol and the corresponding shape
print(acsf_mol)
print('Shape of the ACSF feature matrix',acsf_mol.shape)
```

4.3.2 Atomic Environment Vectors (AEV)

One of the short-comings of Behler-Parrinello ACSFs is the tranferability between systems, to address this Smith, Isayev, and Roitberg proposed the Accurate NeurAl networK engINe for Molecular Energies (ANAKIN-ME, or ANI) family of potentials.[79] The modified angular ACSFs introduced in ANI have been developed for the transferable exploration of configurational and conformational space and are incorporated into a new molecular representation called *atomic environment vectors* (AEVs). The AEVs, $\vec{G}_i^X$, are input for the neural network potentials (NNPs) used in the ANI method. For each atom, i, with atomic number, X, the AEVs take the form,

$$\vec{G}_i^X = \{G_1, G_2, G_3, \ldots, G_M\}, \tag{4.14}$$

where the elements, G_M, describe the radial and angular environment of a specific atom.

The components of the AEV, the radial and angular symmetry functions, follow a similar derivation as that of Behler and Parrinello. ANI utilizes the piece-wise cutoff function defined in Eq. 4.8 and the Behler-Parrinello radial functions defined in Eq. 4.10. To make the notation consistent for the AEV formulation, the radial functions, G_i^2, are redefined as the G_m^R elements of $\vec{G}_i^x$,

$$G_m^R = \sum_{i \neq j} \exp\left(-\eta(R_{ij} - R_s)^2\right) \cdot f_c(R_{ij}), \tag{4.15}$$

where the index m is a set of of parameters, η and R_s. For G_M^R, the set of parameters can be defined as, $M = \{m_1, m_2, m_3, \ldots\} = \{(\eta_1, R_{s_1}), (\eta_2, R_{s_2}), (\eta_3, R_{s_3}), \ldots\}$. It should be noted that the parameters, η and R_s, are the same as those defined by Behler and Parrinello but the ANI potential is built using a singular η and multiple R_s.

The angular symmetry functions in Eq. 4.13 are then modified to introduce additional information related to the local angular environments. The proposed modified angular symmetry function, $G_m^{\mathrm{A_{mod}}}$, is an element of $\vec{G}_X^i$, defined as,

$$\begin{aligned} G_m^{\mathrm{A_{mod}}} = 2^{1-\zeta} \sum_{j,k\neq i} (1 + \cos(\theta_{ijk} - \theta_s))^{\zeta} \cdot \\ \exp\left(-\eta\left(\frac{R_{ij} + R_{ik}}{2} - R_s\right)^2\right) \cdot f_c(R_{ij}) \cdot f_c(R_{ik}), \end{aligned} \tag{4.16}$$

for atoms i, j, and k, the associated angle, θ_{ijk}, centered on atom i, and distances R_{ij} and R_{ik}. The index m runs over four parameters: ζ, θ_s, η, R_s. The modified exponential introduces R_s, which uses the average of the distance of neighboring atoms to consider the angular environment within a given radial shell. The width of peaks in the angular environment are controlled by the ζ parameter. To introduce better angular resolution, the parameter θ_s introduces an arbitrary number of shifts in the angular environment.

4.3.3 Smooth Overlap of Atomic Positions (SOAPs)

The Smooth Overlap of Atomic Positions (SOAPs) is an alternative representation of Behler-Parrinello ACSFs that explicitly defines the similarity between two neighboring atomic environments.[80] Bartók *et al.* introduced SOAPs to bypass the need for descriptors by using a similarity measure, i.e. a kernel $K(\mathbf{q}, \mathbf{q}')$, between atomic neighborhoods. A good similarity measure is characterized as one that has (1) a well-defined limit when comparing either two very similar or different environments, (2) changes smoothly with respect to deviations of the Cartesian coordinate system, and (3) is invariant to symmetry operations of the atoms in each environment. SOAPs define atomic environments by atomic neighbor density functions defined as the sum of Gaussians, expanded in terms of spherical harmonic functions, centered on each neighbor,

$$\rho(\mathbf{r}) = \sum_i \exp(-\alpha|\mathbf{r} - \mathbf{r}_i|^2) = \sum_i \sum_{lm} c_{lm}^i(r) Y_{lm}(\hat{\mathbf{r}}), \tag{4.17}$$

where i runs over the neighboring atoms within a cutoff region. The coefficients $c_{lm}^i(r)$ are defined as:

$$c_{lm}^i(r) \equiv 4\pi \exp\left[\alpha(r^2 + r_i^2)\right] i_l(2\alpha r r_i) Y_{lm}^*(\hat{\mathbf{r}}_i), \tag{4.18}$$

where i_l are the modified spherical Bessel functions of the first kind. The similarity measure (or overlap), which satisfy the permuational invariance criterion,

is then defined as the inner product of two atomic neighbor density functions, $\rho(\mathbf{r})$ and $\rho'(\mathbf{r})$:

$$S(\rho,\rho') = \int \rho(\mathbf{r})\rho'(\mathbf{r})d(\mathbf{r}). \tag{4.19}$$

The overlap between an atomic environment (ρ) and a rotated environment ($\hat{R}\rho'$) is

$$\begin{aligned} S(\hat{R}) \equiv S(\rho,\hat{R}\rho') &= \int d\mathbf{r}\rho(\mathbf{r})\rho'(\hat{R}\mathbf{r}) \\ &= \sum_{i,i'}\sum_{\substack{l,m \\ l',m',m''}} D^{l'}_{m'm''}(\hat{R})\int dr c^{i*}_{lm}(r)c^{i'}_{l'm'}(r)\int d\hat{\mathbf{r}}Y^*_{lm}(\hat{\mathbf{r}})Y_{l'm''}(\hat{\mathbf{r}}) \\ &= \sum_{i,i'}\sum_{l,m,m'}\tilde{I}^l_{mm'}(\alpha,r_i,r_{i'})D^l_{mm'}(\hat{R}) = \sum_{l,m,m'} I^l_{mm'}D^l_{mm'}(\hat{R}), \end{aligned} \tag{4.20}$$

where the integral of the coefficients is

$$\tilde{I}^l_{mm'}(\alpha,r_i,r_{i'}) = 4\pi\exp[-\alpha(r_i^2+r_{i'}^2)/2]i_l(\alpha r_i r_{i'})Y_{lm}(\hat{\mathbf{r}}_i)Y^*_{lm}(\hat{\mathbf{r}}_{i'}) \tag{4.21}$$

and

$$I^l_{mm'} \equiv \sum_{i,i'}\tilde{I}^l_{mm'}(\alpha,r_i,r_{i'}). \tag{4.22}$$

The rotationally invariant similarity kernel is formed via the integration of Eq. 4.19 over all possible rotations of one of the environments:

$$k(\rho,\rho') = \int |S(\rho,\hat{R}\rho')|^n d\hat{R} = \int d\hat{R}\left|\int \rho(\mathbf{r})\rho'(\hat{R}\mathbf{r}))d\mathbf{r}\right|^n \tag{4.23}$$

With $n = 2$, the rotationally invariant kernel becomes

$$\begin{aligned} k(\rho,\rho') &= \int d\hat{R}S^*(\hat{R})S(\hat{R}) \\ &= \sum_{l,m,m'\lambda,\mu,\mu'}(I^l_{mm'})^* I^\lambda_{\mu\mu'}\int d\hat{R}D^*(\hat{R})^l_{mm'}D(\hat{R})^\lambda_{\mu\mu'} \\ &= \sum_{l,m,m'}(I^l_{mm'})^* I^l_{mm'}. \end{aligned} \tag{4.24}$$

Raising k to some power $\zeta \geq 2$ accentuates the sensitivity of the kernel to changing the atomic positions. The SOAP kernel is normalized by dividing by $\sqrt{k(\rho,\rho)k(\rho',\rho')}$ and defined as

$$K(\rho,\rho') = \left(\frac{k(\rho,\rho')}{\sqrt{k(\rho,\rho)k(\rho',\rho')}}\right)^\zeta, \tag{4.25}$$

where ζ is any positive integer.

Python Example 12: SOAPs

```
# Example of generating a smooth overlap of atomic positions
# (SOAPs) representation for glycidol using DScribe and
# the Atomic Simulation Environment (ASE) environment
#
from dscribe.descriptors import SOAP
from ase.io import read

# Import the glycidol xyz
mol = read('../glycidol.xyz')
print(type(mol))


# Set up the SOAP descriptor with parameters:
# species, rcut, nmax, and lmax
soap = SOAP(species=["H", "O","C"],rcut=6.0,nmax=8,lmax=6)


# Create SOAP representation of glycidol
soap_mol = soap.create(mol)

# Print the SOAP representation for glycidol and the shape of the
# feature vector
print(soap_mol)
print('Shape of the SOAP representation of glycidol',
      soap_mol.shape)
```

4.3.4 Faber–Christensen–Huang–Lilienfeld (FCHL)

The Faber-Christensen-Huang-Lilienfeld (FCHL) representation, referred to as FCHL-18 in later articles [81], represents atoms as a sum of multi-dimensional Gaussians representing interatomic many-body terms.[82] Unlike BoBs (Section 4.2.2) representations, FCHL avoids binning by encoding information about atomic species directly into the distributions. Structural and elemental degrees of freedom are explicitly treated and the Gaussian distribution functions are scaled by power laws. The FCHL representation is also rotationally, translationally, and permuationally invariant.

For an atom I in molecule $\mathcal{M}$, the set of interatomic M-body expansions is,

$$\mathcal{A}_M(I) = \{A_1(I), A_2(I), A_3(I), \ldots, A_M(I)\}, \tag{4.26}$$

where the elements, $A_m(I)$, are weighted sums over all m-body interactions. Each element, consisting of Gaussian basis functions, in the sums is weighted by a scaling factor, ξ_m, which determines the importance of each Gaussian. Geometrical information is encoded via structural values and elemental values are encoded using parameters from the periodic table such as P, the period,

and G, the group. For element I, the first order expansion, $A_1(I)$, encodes information related to the chemical composition of the model,

$$A_1(I) = \mathcal{N}(\mathbf{x}_I^{(1)}) = \exp\left(-\frac{(P_I - \chi_1)^2}{2\sigma_P^2} - \frac{(G_I - \chi_2)^2}{2\sigma_G^2}\right), \tag{4.27}$$

with the set of parameters $\mathbf{x}_I^{(1)} = \{P_I, \sigma_p; G_I, \sigma_G\}$. The Gaussian is centered at the period P_I and group G_I on the periodic table, σ_p and σ_G are elemental smearing parameters, χ_1 and χ_2 are dummy variables that are integrated out, and the scaling factor (excluded from Eq. 4.27) is equal to 1. The two-body term, $A_2(I)$, is a product of Eq. 4.27 and a sum over all neighboring atoms I,

$$A_2(I) = \mathcal{N}(\mathbf{x}_I^{(1)}) \sum_{i \neq I} \mathcal{N}(\mathbf{x}_{iI}^{(2)}) \xi_2(d_{iI}). \tag{4.28}$$

The parameters of the Gaussian are defined as $\mathbf{x}_{iI}^{(2)} = \{d_{iI}, \sigma_d; P_i, \sigma_P; G_i, \sigma_G\}$ where the Gaussian is centered at the interatomic distance d_{iI} with width σ_d. The scaling factor, ξ_2,

$$\xi_2(d_{iI}) = \frac{1}{d_{iI}^{n2}}, \tag{4.29}$$

with $n2 = 4$ in the original formulation.

The three-body term, $A_3(I)$, runs over all neighboring atoms j,

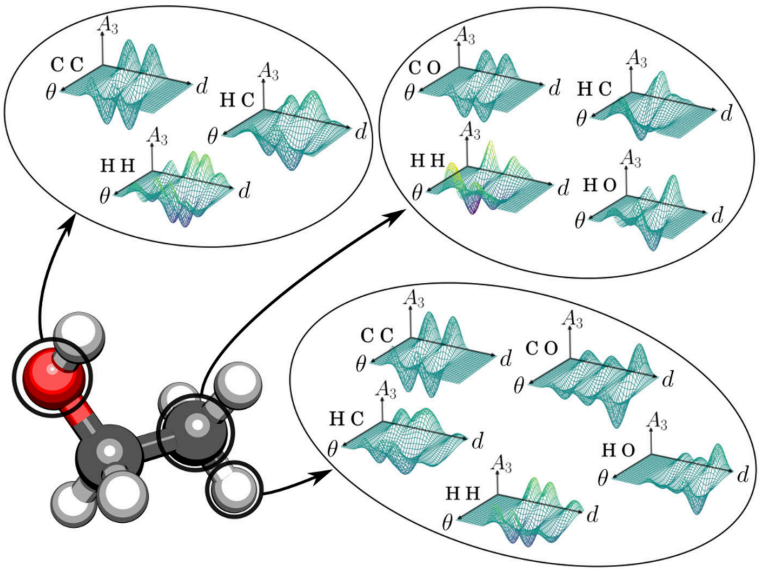
$$A_3(I) = \mathcal{N}(\mathbf{x}_I^{(1)}) \sum_{i \neq I} \mathcal{N}(\mathbf{x}_{iI}^{(2)}) \sum_{j \neq i, I} \mathcal{N}(\mathbf{x}_{ijI}^{(3)}) \xi_3(d_{iI}, d_{jI}, \theta_{ij}^I), \tag{4.30}$$

with the set of associated parameters, $\mathbf{x}_{ijI}^{(3)} = \{\theta_{ij}^I, \sigma_\theta; P_j, \sigma_P; G_j, \sigma_G\}$, where σ_θ is the width of the Gaussian centered at θ_{ij}^I. The three-body scaling function, $\xi_3(d_{iI}, d_{jI}, \theta_{ij}^I)$, is a function of the interatomic distances, d_{iI} and d_{iI}, and the angle, θ_{ij}^I, between the distance vectors $\vec{r}_{Ij}$ and $\vec{r}_{Ij}$,

$$\xi_3(d_{iI}, d_{jI}, \theta_{ij}^I) = \frac{1 - 3\cos(\theta_{ij}^I)\cos(\theta_{Ij}^i)\cos(\theta_{iI}^j)}{(d_{iI} d_{jI} d_{ij})^{n3}}, \tag{4.31}$$

with $n3 = 2$ in the original formulation. An example of the A_3 terms of ethanol is shown in Figure 4.5, where the inlay circles show A_3 as a function of the angle (θ) and radius (d) for H, C, and O.

Figure 4.5: An example of the three-body (A_3) terms, as a function of the angular (θ) and radial (d) degrees of freedom, for ethanol. The inlaid circles show the atomic environments of O, C, and H. *Reproduced from F. A. Faber, A. S. Christensen, B. Huang, and O. A. von Lilienfeld. Alchemical and structural distribution based representation for universal quantum machine learning. Journal of Chemical Physics, 148, 6 2018.*



It should be mentioned, that A_2 and A_3 are equivalent to radial distribution functions (RDF) and angular distribution functions (ADF). When σ_P and σ_d approach zero, A_2 is equivalent to using a ADF for each element pair and as σ_d approaches infinity, A_3 is equivalent to using an ADF.

To incorporate this method into kernel ridge regression, a novel distance metric must be derived between two atomic environments I and J,

$$\Delta(\mathcal{A}_M(I), \mathcal{A}_M(J))^2 = \sum_{m=0}^{M} \beta_m \Delta(A_m(I), A_m(J))^2 \tag{4.32}$$

where β_m is a weighting hyperparameter. A more in depth derivation of the individual distances used in the FCHL scheme can be found in Ref. [82]. The associated Gaussian FCHL kernel is,

$$K(\mathcal{M}, \mathcal{M}') = \sum_{I \epsilon \mathcal{M}} \sum_{I \epsilon \mathcal{M}'} k(\Delta(\mathcal{A}_M(I), \mathcal{A}_M(J))), \tag{4.33}$$

for the molecule of interest, $\mathcal{M}$, and a query compound, $\mathcal{M}'$.

Available Software Packages

Some useful packages that include the molecular representations discussed in the previous sections:

- RuNNer
  - Atom-centered Symmetry Functions (ACSF)
- Accurate NeurAl networK engINe for Molecular Energies (ANI)
  - ASE-ANI
  - TorchANI
- ChemML
  - Coulomb Matrix
  - Bag of Bonds
- DScribe
  - Atom-centered Symmetry Functions (ACSF)
  - Smooth Overlap of Atomic Positions (SOAP)
- QML
  - Faber–Christensen–Huang–Lilienfeld (FCHL)

4.3.5 Permutationally Invariant Potentials (PIPs)

Permutationally invariant potentials (PIPs) are a class of molecular representations originally developed for the study of potential energy surfaces (PESs).[83, 84, 85, 86, 87, 88] PIPs are based on invariant polynomials, which are a class of polynomials that are invariant under a group acting on a vector space. A brief overview of the theory of invariant polynomials from a mathematical perspective, with respect to the application in PIPs, can be found in Ref. [83]. The motivation behind PIPs is that permutational symmetry should be incorporated into the PES representation through the use of a permutationally invariant fitting basis. The PIP is expressed as

$$V(\mathbf{y}) = \sum_{\alpha=1}^{M} h_\alpha(\mathbf{p}(\mathbf{y}))q_\alpha(\mathbf{y}) \tag{4.34}$$

where the vector $\mathbf{y}$ is the collection of Morse variables y_{ij} with $N_{atoms}(N_{atoms}-1)/2$ components. The vector of primary invariant polynomials $\mathbf{p}(\mathbf{y})$ has $N_{atoms}(N_{atoms}-1)/2$ components and for $1 \leq \alpha \leq M$ the secondary invariant polynomials are denoted as $q_\alpha(\mathbf{y})$. At each order, the number of terms in Eq. 4.34 is determined

by the Molien series,

$$M(t) = (\sum_{\alpha=1}^{M} t^{e_\alpha})\Pi_{i=1}^{n}(1-t^{d_i})^{-1}. \tag{4.35}$$

A set of invariant polynomials is the set of polynomials on a n-dimensional vector space over a field for which the polynomials are invariant under action of a group on the vector space. The set of invariant polynomials forms a vector space, ring, and algebra.

4.3.6 SchNet

As mentioned in Chapter 2, molecular graphs have been extensively applied in the fields of chemoinformatics and chemical machine learning. Graphs, along with graph neural networks (GNNs), can be extended to study quantum mechanical properties of molecules. SchNet is a hybrid quantum chemical/machine learning model that extends the ideas of ACSFs and utilizes a molecular graph-based architecture.[89, 90]

Like other graph-based molecular representations, the graph of a molecule, $\mathcal{M}$, is defined by the nuclear charges, $Z = (Z_1, \ldots, Z_{N_{\text{atoms}}})$, and nuclear positions, $R = (\mathbf{r}_1, \ldots, \mathbf{r}_{N_{\text{atoms}}})$. As input for a machine learning model, the tuple of nuclear charges does not provide a unique representation since molecules often contain multiple atoms of the same type. For example, in glycidol there are six hydrogens–all of which would be indistinguishable to a machine learning model. SchNet incorporates atom-dependent embeddings, which are a common starting point in molecular graph-based neural networks. An atom i is initially represented by the atom-dependent embedding,

$$\mathbf{x}_i^0 = \mathbf{a}_{Z_i}, \tag{4.36}$$

where each atom is assigned a unique random number, which initially, does not carry any chemical information. These embeddings are updated to include chemical information as they are passed through the network layers defined below and are optimized during the training iterations. In this sense, atom embeddings are *learned* by the network during the training process. Going forward this embedding is denoted by the tuple of features $X^l = (\mathbf{x}_1^l \ldots, \mathbf{x}_n^l)$ for $\mathbf{x}_l^i \in \mathbb{R}^F$, where $\mathbb{R}^F$ is the number of associated feature maps and l denote the current layer.

Once the nuclear charges are transformed into unique atom embeddings, the embeddings are passed to the interaction block, which incorporates pairwise interactions between neighboring atoms. Within the interaction block (Figure 4.6a right), the atom embeddings are passed through an atom-wise layer for each atom i, which is defined as,

$$\mathbf{x}_i^{l+1} = W^l \mathbf{x}_i^l + \mathbf{b}^l, \tag{4.37}$$

where the weights W^l and b^l are shared across each atoms to maintain a scalable network architecture.

The next layer within the interaction block incorporates a continuous-filter convolutional (cfconv) layer. While image and audio data can be treated as discrete objects, the position of atoms in a molecule cannot be treated with a traditional filter-tensor. The cfconv layer is formulated as,

$$\mathbf{x}_i^{l+1} = (X^l * W^l)_i = \sum_{j=0}^{N_{atoms}} \mathbf{x}_j^l \circ W^l(\mathbf{r}_j - \mathbf{r}_i), \tag{4.38}$$

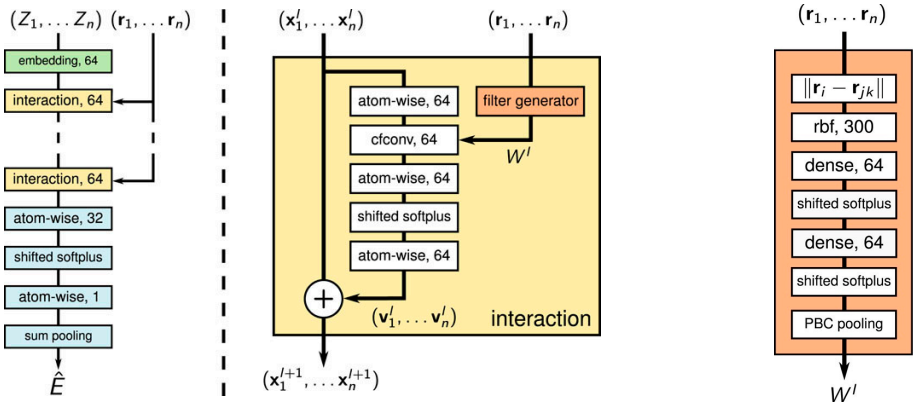
where W^l denotes a filter and $\circ$ is defined as element-wise multiplication. W^l is created by the filter generator block (Figure 4.6b) to include the interactions between atoms i and a neighboring atom j.

Within the filter generator block, the interaction between atoms is *learned* by the filter value, $W(\mathbf{r}_j - \mathbf{r}_i)$, where $\mathbf{r}_j - \mathbf{r}_i$ is the vector from atoms i to j. Rotational invariance is incorporated into the model by a Gaussian basis,

$$e_k(\mathbf{r}_j - \mathbf{r}_i) = \exp\left(-\gamma(\|\mathbf{r}_j - \mathbf{r}_i\|_2 - \mu_k)^2\right), \tag{4.39}$$

where the centers μ_k are on a uniform grid from zero to the cutoff distance. This introduces decorrelation among filter values to improve optimization conditioning. The resolution of the filter is determined by the number of Gaussians and the hyperparameter γ. The remaining layers in the overall architecture (Figure 4.6a) are additional interaction, atom-wise, or dense layers.

Figure 4.6: (a) The full SchNet architecture is shown on the left, while an example of the interaction block is shown in depth on the right. In (b) the filter generator block is shown. *Reproduced from K. T. Schüutt, H. E. Sauceda, P. J. Kindermans, A. Tkatchenko, and K. R. Müller. Schnet - a deep learning architecture for molecules and materials. Journal of Chemical Physics, 148, 6 2018.*



4.4 *Ab Initio* Representations

We now turn our attention to molecular representations that utilize data from molecular wave functions and, in particular, from quantum chemical methods. These methods are typically parameter-free methods and they are solely based on laws of quantum mechanics. For that reason, they often called as *ab initio* or *first principles* methods. Recently, there has been a high interest from the theoretical and computational chemistry community on the development of methodologies that utilize the strength of modern data sciences for the acceleration or completely elimination of the solution of complex equations, without of course the loss of the high accuracy of the quantum chemical methods. The motivation behind such data-driven approaches is that those complex equations, such as the post-Hartree-Fock (post-HF) methods,[91] require a significant amount of computational time and resources.

The field of quantum chemistry investigates the application of quantum mechanics on atomistic systems for the study of their physicochemical properties. The molecular wave function is the cornerstone function of quantum mechanics that holds all the pertinent information with respect to the electron and nuclear positions. Therefore, it becomes evident that information that is directly or indirectly extracted from these wave functions can be effectively used as input for machine learning for the training of models that can predict molecular energies or properties. Input features can range from molecular orbital energies, electron densities, atom-based properties (e.g. partial atomic charges), till intrinsic wave function properties derived from the quantum chemical operators, such as Fock matrix elements, one- and two-electron integrals, or electron correlation terms. In this Section, we will discuss representative methodologies that utilize such *molecular electronic structure representations*.

4.4.1 Molecular orbital based-ML (MOB-ML)

The scope of *Molecular orbital based-ML* (MOB-ML) models are the fast estimation of accurate electronic energies and nuclear gradients at low cost.[92, 93, 94] MOB-ML utilizes information pertinent to molecular orbitals as input, an approach that secures transferability across different molecular species of variable size and composition. Here, we will focus on the quantum chemical information that is introduced in MOB-ML. In our discussion, we have selected to follow the original notation used in Ref. [92].

MOB-ML is a straightforward method for learning the electron correlation of post-HF methods, such as Møller-Plesset second-order perturbation theory (MP2) and coupled-cluster singles and doubles (CCSD). Using Nesbet's theorem, the correlation energy of a general post-HF method can be written in the form,

$$E_{\text{corr}} = \sum_{ij} \varepsilon_{ij}, \tag{4.40}$$

where ε_{ij} are the pair-energies, i.e. the electron correlation energy terms that

arise from promoting (exciting) two electrons from occupied orbitals to unoccupied (virtual) orbitals.

The main idea behind MOB-ML is that each pair energy ε_{ij} depends on information related to molecular orbitals i and j. Thus, a ML model can be developed for estimating the expensive ε_{ij} of coupled-cluster theory by using as input a representation that is based on quantum chemical properties related to the two molecular orbitals i and j. MOB-ML correlates the pair-energies ε_{ij} with a feature vector $\mathbf{f}$, while it considers separately two-electron promotions from the same molecular orbital i (diagonal or d) and promotions from different orbitals i and j (off-diagonal or o):

$$\varepsilon_{ii} = \varepsilon_d(\mathbf{f}_i) \text{ and } \varepsilon_{ij} = \varepsilon_o(\mathbf{f}_{ij}). \tag{4.41}$$

For the training and application of the MOB-ML models, we require data that are obtained from a quantum chemical method that is computationally less demanding than the expensive coupled-cluster scheme (e.g. CCSD). Hartree-Fock theory serves that purpose, since it is the method that provides the set of optimized molecular orbitals that CCSD requires. Specifically, a MOB-ML feature set is derived from the Fock matrix ($\mathbf{F}$), Coulomb matrix ($\mathbf{J}$), and exchange matrix ($\mathbf{K}$). The full feature set for a given pair of occupied MOs i, j is

$$\mathbf{f}_{ij} = \left(\mathbf{f}_{ij}^{(\mathbf{F})}, \mathbf{f}_{ij}^{(\mathbf{J})}, \mathbf{f}_{ij}^{(\mathbf{K})}\right), \tag{4.42}$$

where $\mathbf{f}_{ij}^{(\mathbf{F})}$, $\mathbf{f}_{ij}^{(\mathbf{J})}$, and $\mathbf{f}_{ij}^{(\mathbf{K})}$ denote the feature vectors associated with the aforementioned matrices, defined as

$$\mathbf{f}_{ij}^{(\mathbf{F})} = \left(F_{ii}, F_{ij}, F_{jj}, \mathbf{F}_{ij}^{vv}\right) \tag{4.43}$$

$$\mathbf{f}_{ij}^{(\mathbf{J})} = \left(J_{ii}, J_{ij}, J_{jj}, \mathbf{J}_i^v, \mathbf{J}_j^v, \mathbf{J}_{ij}^{vv}\right) \tag{4.44}$$

$$\mathbf{f}_{ij}^{(\mathbf{K})} = \left(K_{ij}, \mathbf{K}_i^v, \mathbf{K}_j^v, \mathbf{K}_{ij}^{vv}\right). \tag{4.45}$$

The terms with the superscript v denote the inclusion of virtual orbitals, i.e.

$$\mathbf{J}_i^v = (J_{ia}, J_{ib}, J_{ic}, \ldots) \tag{4.46}$$

$$\mathbf{J}_j^v = (J_{ja}, J_{jb}, J_{jc}, \ldots) \tag{4.47}$$

$$\mathbf{K}_i^v = (K_{ia}, K_{ib}, K_{ic}, \ldots) \tag{4.48}$$

$$\mathbf{K}_j^v = (K_{ja}, K_{jb}, K_{jc}, \ldots), \tag{4.49}$$

where $a, b, c, \ldots$ denote virtual orbitals. $\mathbf{J}_i^v$, $\mathbf{J}_j^v$, $\mathbf{K}_i^v$, and $\mathbf{K}_j^v$ are sorted based on the contribution of the virtual orbital to the term, i.e. $J_{ia} > J_{ib} > J_{ic} > \ldots$. $\mathbf{K}_{ij}^{vv}$,$\mathbf{J}_{ij}^{vv}$, and $\mathbf{K}_{ij}^{vv}$ denote virtual-virtual matrix elements selected and sorted such that $J_{ia} + J_{ja} > J_{ib} + J_{jb}$, etc.

4.4.2 Data-Driven Quantum Chemistry (DDQC)

The last section of this eBook covers an alternative approach to represent molecular information based on *abinitio* data for machine learning algorithms.

This representation encodes wave function information from low-level quantum chemical methods for the development of models that predict wave functions of high-level, accurate wave quantum chemical methods. As we will discuss here, it has been used successfully applied in a family of methods called data-driven quantum chemistry (DDQC) methods which include the data-driven coupled cluster (DDCC) and data-driven complete active space second-order perturbation theory (DDCASPT2) methods.[95, 96, 52] Here, we will briefly cover the fundamentals of the DDCC methodology. Further information on the implementation of DDCC and DDCASPT2 can be found in our recent book chapter (Ref. [52]) and the associated website with case studies https://chemracer.github.io/DDQC_Demo/.

DDCC comes in several flavors: an non-alchemical approach to learn the exact $\hat{T}_2$ amplitudes in coupled-cluster singles and doubles (CCSD) by iterating over a predicted set of amplitudes, an alchemical approach where the predicted $\hat{T}_2$ amplitudes are used explicitly for the calculation of the CCSD energy, or where the CCSD pair-energies (ε_{ij}) are learned from the Møller-Plesset second-order perturbation theory (MP2) pair-energies. All three flavors are formulated using features derived at the Hartree-Fock (HF) and MP2 level.

To capture the missing dynamical correlation from HF, coupled-cluster theory introduces electron promotions (excitations) from occupied spin orbitals i, j, to unoccupied (virtual) orbitals a, b. This is formulated via the coupled-cluster wave function,

$$|\Psi_{\mathrm{CC}}\rangle = e^{(\hat{T})}|\Psi_0\rangle = e^{(\hat{T}_1+\hat{T}_2+\hat{T}_3+\ldots)}|\Psi_0\rangle, \tag{4.50}$$

where $\hat{T}$ is the cluster excitation operator. In CCSD, the cluster operator is truncated to only use single ($\hat{T}_1$) and double ($\hat{T}_2$) electron promotions from the occupied to the unoccupied orbitals. An individual coupled-cluster amplitude (*weight*) corresponds to each single (t_i^a) and double (t_{ij}^{ab}) electron promotion. To obtain the coupled-cluster energy, the set of projected coupled-cluster equations are iteratively solved to find the optimal amplitudes.

The computation of the one-electron t_i^a amplitudes is fast, while the two-electron t_{ij}^{ab} amplitudes requires significantly more computational effort. A common approach to initialize the t_{ij}^{ab} amplitudes is to use the two-electron amplitudes from a computational less demanding method such as MP2. The MP2 amplitudes are defined as

$$t_{ij(MP2)}^{ab} = \frac{\langle ij||ab\rangle}{\varepsilon_i + \varepsilon_j - \varepsilon_a - \varepsilon_b} \tag{4.51}$$

where the term $\langle ij||ab\rangle$ on the numerator are two-electron integrals and the ε_p terms of the denominator are the orbital energies of the occupied i, j and unoccupied a, b orbitals. Since the final set of two-electron amplitudes in CCSD rely on the the MP2 amplitudes, we incorporate $t_{ij(MP2)}^{ab}$, along with the subsequent components, i.e. the two-electron integrals ($\langle ab||ij\rangle$) and the difference in orbital energy ($\varepsilon_i+\varepsilon_j-\varepsilon_a-\varepsilon_b$), are incorporated into our feature set. Our previous study has reaffirmed this and have shown that the most important features in

the DDCC feature set is the MP2 two-electron amplitudes. Features related to the energy of each orbital involved in a two-electron excitation are incorporated which include the orbital energies ($\varepsilon_i, \varepsilon_j, \varepsilon_a, \varepsilon_b$), Coulomb ($J_i^a$, J_j^b) and exchange terms (K_i^a, K_j^b), a Boolean feature to denote whether two-electrons are promoted to the same virtual orbital and one to denote whether the t_2 amplitude is greater than zero, and the log_{10} value of the MP2 amplitudes and of the Coulomb and exchange integrals. For a single t_2 amplitude, the feature set consists of 30 electronic properties.

As mentioned previously, the set of predicted t_2 amplitudes can be either iteratively updated to solve for the exact CC energy using Eq. **??** or use to predict the final CC energy without iteration. Since the first case provides an exact CC energy, the DDCC method *learns* the wavefunction defined in Eq. 4.50. Another highlight of the DDCC method is that the feature set is permutationally, rotationally, and translationally invariant and offers a non-alchemical, transferable data-driven quantum chemistry method.

4.4.3 Density Functional Based Molecular Representations

In this section, we will briefly mention some density functional theory (DFT) based ML representations. We will avoid an in-depth look into these methods since two book chapters in the book *Quantum Chemistry in the Age of Machine Learning* provide adequate overviews of the modern methods.[97, 98] DFT is one of the most important and broadly applied methods in computational chemistry due to its efficiency and accuracy.[99] Many DFT base ML approaches are formulated to learn the exchange-correlation (XC) functionals[100, 101, 102, 103, 104, 105, 106, 107, 108], kinetic energy functionals[109, 110, 111, 112, 113, 114, 115, 116], local range-separated hybrid functionals[117], and electron densities.[118, 119, 120]

4.5 That's a Wrap

End-of-Chapter

A comment on physics-based representation, their strengths, their weaknesses, and where do we use them. These statements will be added once we have a first complete version of the ACS in Focus book.

Read This Next

- Machine Learning Meets Quantum Physics, ed. K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, K.-R. Müller, **2020**, Springer.
- Felix Musil, Andrea Grisafi, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Physics-Inspired Structural Representations for Molecules and Materials. *Chem. Rev.*, **2021**, *121*, 9759–9815.
- Quantum Chemistry in the Age of Machine Learning. Editor: Pavlo Dral, **2022**, Elsevier.

Bibliography

[1] William J. Wiswesser. 107 years of line-formula notations (1861-1968). *Journal of Chemical Documentation*, 8:146–150, 8 1968. ISSN 0021-9576. doi: 10.1021/c160030a007. URL https://pubs.acs.org/doi/abs/10.1021/c160030a007.

[2] William J. Wiswesser. The wiswesser line formula notation. *Chemical & Engineering News Archive*, 30(34):3523–3526, 1952. ISSN 0009-2347. doi: 10.1021/cen-v030n034.p3523. URL https://doi.org/10.1021/cen-v030n034.p3523. doi: 10.1021/cen-v030n034.p3523.

[3] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965. ISSN 0021-9576. doi: 10.1021/c160017a018. URL https://doi.org/10.1021/c160017a018. doi: 10.1021/c160017a018.

[4] William J. Wiswesser. How the wln began in 1949 and how it might be in 1999. *Journal of Chemical Information and Computer Sciences*, 22(2):88–93, 11982. ISSN 0095-2338. doi: 10.1021/ci00034a005.

[5] William J. Wiswesser. Historic development of chemical notations. *Journal of Chemical Information and Computer Sciences*, 25(3):258–263, 1985. doi: 10.1021/ci00047a023. URL https://doi.org/10.1021/ci00047a023.

[6] Thomas Engel and Johann Gasteiger. *Chemoinformatics: basic concepts and methods*. John Wiley & Sons, 2018.

[7] P G Dittmar, N A Farmer, W Fisanick, R C Haines, and J Mockus. The cas online search system. 1. general system design and selection, generation, and use of search screens. *J. Chem. Inf. Comput. Sci*, 23:93–102, 1983. URL https://pubs.acs.org/sharingguidelines.

[8] Yanli Wang, Jewen Xiao, Tugba O Suzek, Jian Zhang, Jiyao Wang, and Stephen H Bryant. Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research*, 37(suppl_2):W623–W633, 2009.

[9] Greg Landrum, Paolo Tosco, Brian Kelley, Ric, sriniker, gedeck, Riccardo Vianello, NadineSchneider, Eisuke Kawashima, David Cosgrove, Andrew Dalke, Dan N, Gareth Jones, Brian Cole, Matt Swain, Samo Turk, AlexanderSavelyev, Alain Vaucher, Maciej Wójcikowski, Ichiru Take, Daniel Probst, Kazuya Ujihara, Vincent F. Scalfani, guillaume godin, Axel Pahl, Francois Berenger, JLVarjo, strets123, JP, and DoliathGavid. rdkit/rdkit: 2022_03_5 (q1 2022) release, August 2022. URL https://doi.org/10.5281/zenodo.6961488.

[10] Noel M. O'Boyle, Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. Open babel: An open chemical toolbox. *Journal of Cheminformatics*, 3, 10 2011. ISSN 17582946. doi: 10.1186/1758-2946-3-33.

[11] Mojtaba Haghighatlari, Gaurav Vishwakarma, Doaa Altarawy, Ramachandran Subramanian, Bhargava Urala Kota, Aditya Sonpal, Srirangaraj Setlur, and Johannes Hachmann. Chemml: A machine learning and informatics program package for the analysis, mining, and modeling of chemical and materials data. *ChemRxiv*, page 8323271, 2019. doi: 10.26434/chemrxiv.8323271.v1.

[12] Lauri Himanen, Marc O.J. Jäger, Eiaki V. Morooka, Filippo Federici Canova, Yashasvi S. Ranawat, David Z. Gao, Patrick Rinke, and Adam S. Foster. Dscribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications*, 247:106949, 2 2020. ISSN 00104655. doi: 10.1016/j.cpc.2019.106949. URL https://linkinghub.elsevier.com/retrieve/pii/S0010465519303042.

[13] Sergey N. Pozdnyakov, Michael J. Willatt, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Incompleteness of atomic structure representations. *Physical Review Letters*, 125, 10 2020. ISSN 10797114. doi: 10.1103/PhysRevLett.125.166001.

[14] Henry Wiener. Structural determination of paraffin boiling points. *J Am Chem Soc*, 69(1):17–20, Jan 1947. ISSN 0002-7863 (Print); 0002-7863 (Linking). doi: 10.1021/ja01193a005.

[15] Milan Randić. Novel molecular descriptor for structure—property studies. *Chemical Physics Letters*, 211(4):478–483, 1993.

[16] V. Consonni, R. Todeschini, R. Mannhold, H. Kubinyi, and G. Folkers. *Molecular Descriptors for Chemoinformatics: Volume I: Alphabetical Listing / Volume II: Appendices, References*. Methods & Principles in Medicinal Chemistry. Wiley, 2009. ISBN 9783527628773. URL https://books.google.com/books?id=03iAsdAcXHcC.

[17] Milan Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23):6609–6615, 11 1975. doi: 10.1021/ja00856a001. URL https://doi.org/10.1021/ja00856a001.

[18] I. Gutman and N. Trinajstić. Graph theory and molecular orbitals. total ϕ-electron energy of alternant hydrocarbons. *Chemical Physics Letters*, 17(4):535–538, 1972.

[19] Kinkar Ch. Das, Kexiang Xu, and Junki Nam. Zagreb indices of graphs. *Frontiers of Mathematics in China*, 10(3):567–582, 2015.

[20] Boris Furtula, Ante Graovac, and Damir Vukičević. Augmented zagreb index. *Journal of Mathematical Chemistry*, 48(2):370–380, 2010.

[21] Ernesto Estrada, Luis Torres, Lissette Rodriguez, and Ivan Gutman. An atom-bond connectivity index: Modeling the enthalpy of fomation of alkanes. *Indian Journal of Chemistry*, 37:849–855, 10 1998.

[22] Haruo Hosoya. Topological index. a newly proposed quantity characterizing the topological nature of structural isomers of saturated hydrocarbons. *Bulletin of the Chemical Society of Japan*, 44(9):2332–2339, 1971.

[23] Ernesto Estrada. Characterization of 3d molecular structure. *Chemical Physics Letters*, 319(5):713–718, 2000.

[24] S. Klavžar, A. Rajapakse, and I Gutman. The szeged and the wiener index of graphs. *Applied Mathematics Letters*, 9(5):45–49, 1996.

[25] Padmakar V. Khadikar, Sneha Karmarkar, and Vijay K. Agrawal. A novel pi index and its applications to qspr/qsar studies. *Journal of Chemical Information and Computer Sciences*, 41(4):934–949, 07 2001.

[26] Ivan Gutman. Selected properties of the schultz molecular topological index. *Journal of Chemical Information and Computer Sciences*, 34(5): 1087–1089, 1994.

[27] Lingping Zhong. The harmonic index for graphs. *Applied Mathematics Letters*, 25(3):561–566, 2012.

[28] Yan Yuan, Bo Zhou, and Nenad Trinajstić. On geometric-arithmetic index. *Journal of Mathematical Chemistry*, 47:833–841, 02 2010. doi: 10.1007/s10910-009-9603-8.

[29] Jon Paul Janet and Heather J Kulik. Resolving transition metal chemical space: Feature selection for machine learning and structure–property relationships. *The Journal of Physical Chemistry A*, 121(46):8939–8954, 2017.

[30] Efthymios I. Ioannidis, Terry Z. H. Gani, and Heather J. Kulik. molsimplify: A toolkit for automating discovery in inorganic chemistry. *Journal of Computational Chemistry*, 37(22):2106–2117, 2016. ISSN 1096-987X. doi: 10.1002/jcc.24437. URL http://dx.doi.org/10.1002/jcc.24437.

[31] Aditya Nandy, Chenru Duan, Jon Paul Janet, Stefan Gugler, and Heather J. Kulik. Strategies and software for machine learning accelerated discovery in transition metal chemistry. *Industrial & Engineering Chemistry Research*, 57(42):13973–13986, 2018. ISSN 0888-5885. doi: 10.1021/acs.iecr.8b04015. URL https://doi.org/10.1021/acs.iecr.8b04015.

[32] A. Cereto-Massagué, M. J. Ojeda, C. Valls, M. Mulero, S. Garcia-Vallvé, and G. Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63, 2015. ISSN 1095-9130 (Electronic) 1046-2023 (Linking). doi: 10.1016/j.ymeth.2014.08.005. URL https://www.ncbi.nlm.nih.gov/pubmed/25132639.

[33] Daniel S. Wigh, Jonathan M. Goodman, and Alexei A. Lapkin. A review of molecular representation in the age of machine learning. *WIREs Computational Molecular Science*, 2 2022. ISSN 1759-0876. doi: 10.1002/wcms.1603. URL https://onlinelibrary.wiley.com/doi/10.1002/wcms.1603.

[34] Robert C Glen, Andreas Bender, Catrin H Arnby, Lars Carlsson, Scott Boyer, and James Smith. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to adme. *IDrugs*, 9(3):199, 2006.

[35] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50:742–754, 5 2010. ISSN 1549960X. doi: 10.1021/ci100050t.

[36] Joseph L. Durant, Burton A. Leland, Douglas R. Henry, and James G. Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of Chemical Information and Computer Sciences*, 42:1273–1280, 11 2002. ISSN 0095-2338. doi: 10.1021/ci010132r. URL https://pubs.acs.org/doi/10.1021/ci010132r.

[37] Accelerys. The keys to understanding mdl keyset technology. Report, 2011. URL https://www.yumpu.com/en/document/read/4611723/the-keys-to-understanding-mdl-keyset-technology-accelrys.

[38] Openeye toolkits 2022.1.1. openeye scientific software and santa fe, nm. URL http://www.eyesopen.com.

[39] Keith Taylor. Description of public maccs keys, 2007. URL https://list.indiana.edu/sympa/arc/chminf-l/2007-11/msg00058.html.

[40] Manish Sud. Maccskeys, 2022. URL http://www.mayachemtools.org/docs/modules/html/MACCSKeys.html.

[41] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28:31–36, 2 1988. ISSN 1549-9596. doi:

10.1021/ci00057a005. URL https://pubs.acs.org/doi/abs/10.1021/ci00057a005.

[42] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4:268–276, 2 2018. ISSN 23747951. doi: 10.1021/acscentsci.7b00572.

[43] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of Cheminformatics*, 9:48, 12 2017. ISSN 1758-2946. doi: 10.1186/s13321-017-0235-x. URL https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0235-x.

[44] Noel M. O'Boyle. Towards a universal smiles representation - a standard method to generate canonical smiles based on the inchi. *Journal of Cheminformatics*, 4, 9 2012. ISSN 17582946. doi: 10.1186/1758-2946-4-22.

[45] Nadine Schneider, Roger A. Sayle, and Gregory A. Landrum. Get your atoms in order-an open-source implementation of a novel and robust molecular canonicalization algorithm. *Journal of Chemical Information and Modeling*, 55:2111–2120, 10 2015. ISSN 1549960X. doi: 10.1021/acs.jcim.5b00543.

[46] David Weininger, Arthur Weininger, and Joseph L. Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *Journal of Chemical Information and Computer Sciences*, 29:97–101, 5 1989. ISSN 0095-2338. doi: 10.1021/ci00062a008. URL https://pubs.acs.org/doi/abs/10.1021/ci00062a008.

[47] Daylight chemical information systems inc. 4. smarts—a language for describing molecular patterns. URL https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html.

[48] Noel M O'boyle and Andrew Dalke. Deepsmiles: An adaptation of smiles for use in machine-learning of chemical structures. *chemrxiv*. doi: https://doi.org/10.26434/chemrxiv.7097960.v1. URL https://chemrxiv.org/engage/chemrxiv/article-details/60c73ed6567dfe7e5fec388d.

[49] Mario Krenn, Florian Häse, Akshat Kumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1, 12 2020. ISSN 26322153. doi: 10.1088/2632-2153/aba947.

[50] Stephen Heller, Alan McNaught, Stephen Stein, Dmitrii Tchekhovskoi, and Igor Pletnev. Inchi - the worldwide chemical structure identifier standard. *Journal of Cheminformatics*, 5, 1 2013. ISSN 17582946. doi: 10.1186/1758-2946-5-7.

[51] Bryn Keller, Michael Lesnick, Suny Albany, and Ted Willke. Persistent homology for virtual screening. *chemrXiv*, 2018. doi: 10.26434/chemrxiv.6969260.v3.

[52] Grier M Jones, PD Varuna S Pathirage, and Konstantinos D Vogiatzis. Data-driven acceleration of coupled-cluster and perturbation theory methods. In *Quantum Chemistry in the Age of Machine Learning*, pages 509–529. Elsevier, 2023.

[53] Jacob Townsend, Cassie Putman Micucci, John H. Hymel, Vasileios Maroulas, and Konstantinos D. Vogiatzis. Representation of molecular structures with persistent homology for machine learning applications in chemistry. *Nature Communications*, 11:3230, 12 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17035-5. URL http://www.nature.com/articles/s41467-020-17035-5.

[54] Yair Schiff, Vijil Chenthamarakshan, Samuel Hoffman, Karthikeyan Natesan Ramamurthy, and Payel Das. Augmenting molecular deep generative models with topological data analysis representations. *arXiv*, 6 2021. URL http://arxiv.org/abs/2106.04464.

[55] Clément Maria, Jean-Daniel Boissonnat, Marc Glisse, and Mariette Yvinec. The Gudhi library: Simplicial complexes and persistent homology. In *International Congress on Mathematical Software*, pages 167–174. Springer, 2014.

[56] The GUDHI Project. *GUDHI User and Reference Manual*. GUDHI Editorial Board, 2015. URL http://gudhi.gforge.inria.fr/doc/latest/.

[57] Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J. Guibas, and Steve Y. Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry*, SCG '09, page 237–246, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585017. doi: 10.1145/1542362.1542407. URL https://doi.org/10.1145/1542362.1542407.

[58] Andrew Marchese and Vasileios Maroulas. Signal classification with a point process distance on the space of persistence diagrams. *Advances in Data Analysis and Classification*, 12(3):657–682, 2018.

[59] Vasileios Maroulas, Cassie Putman Micucci, and Adam Spannaus. A stable cardinality distance for topological classification. *Advances in Data Analysis and Classification*, 14(3):611–628, 2020.

[60] Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2022.

[61] Peter Bubenik. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.*, 16(1):77–102, jan 2015. ISSN 1532-4435.

[62] Vasileios Maroulas, Joshua L Mike, and Christopher Oballe. Nonparametric estimation of probability density functions of random persistence diagrams. *Journal of Machine Learning Research*, 20(151):1–49, 2019. URL http://jmlr.org/papers/v20/18-618.html.

[63] Theodore Papamarkou, Farzana Nasrin, Austin Lawson, Na Gong, Orlando Rios, and Vasileios Maroulas. A random persistence diagram generator. *Statistics and Computing*, 32(5):88, 2022. doi: 10.1007/s11222-022-10141-y. URL https://doi.org/10.1007/s11222-022-10141-y.

[64] Vasileios Maroulas, Farzana Nasrin, and Christopher Oballe. A bayesian framework for persistent homology. *SIAM Journal on Mathematics of Data Science*, 2(1):48–74, 2020.

[65] Andrew Marchese, Vasileios Maroulas, and Josh Mike. K-means clustering on the space of persistence diagrams. In Yue M. Lu, Dimitri Van De Ville, and Manos Papadakis, editors, *Wavelets and Sparsity XVII*, volume 10394, page 103940W. International Society for Optics and Photonics, SPIE, 2017.

[66] Christopher Oballe, David Boothe, Piotr J. Franaszczuk, and Vasileios Maroulas. Tofu: Topology functional units for deep learning. *Foundations of Data Science*, 4(4), 2022. doi: 10.3934/fods.2021021.

[67] Henry Adams, Tegan Emerson, Michael Kirby, Rachel Neville, Chris Peterson, Patrick Shipman, Sofya Chepushtanova, Eric Hanson, Francis Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *Journal of Machine Learning Research*, 18(8):1–35, 2017. URL http://jmlr.org/papers/v18/16-337.html.

[68] Pietro Donatini, Patrizio Frosini, and Alberto Lovato. Size functions for signature recognition. In Robert A. Melter, Angela Y. Wu, and Longin Jan Latecki, editors, *Vision Geometry VII*, volume 3454, pages 178 – 183. International Society for Optics and Photonics, SPIE, 1998.

[69] Massimo Ferri, Patrizio Frosini, Alberto Lovato, and Chiara Zambelli. Point selection: A new comparison scheme for size functions (with an application to monogram recognition). In Roland Chin and Ting-Chuen Pong, editors, *Computer Vision — ACCV'98*, pages 329–337, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[70] Matthias Rupp, Alexandre Tkatchenko, Klaus Robert Müller, and O. Anatole von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108, 1 2012. ISSN 00319007. doi: 10.1103/PhysRevLett.108.058301.

[71] Katja Hansen, Grégoire Montavon, Franziska Biegler, Siamac Fazli, Matthias Rupp, Matthias Scheffler, O. Anatole von Lilienfeld, Alexandre Tkatchenko, and Klaus Robert Müller. Assessment and validation of

machine learning methods for predicting molecular atomization energies. *Journal of Chemical Theory and Computation*, 9:3404–3419, 8 2013. ISSN 15499618. doi: 10.1021/ct400195d.

[72] Katja Hansen, Franziska Biegler, Raghunathan Ramakrishnan, Wiktor Pronobis, O. Anatole von Lilienfeld, Klaus Robert Müller, and Alexandre Tkatchenko. Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space. *Journal of Physical Chemistry Letters*, 6:2326–2331, 6 2015. ISSN 19487185. doi: 10.1021/acs.jpclett.5b00831.

[73] Bing Huang and O. Anatole von Lilienfeld. Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity. *Journal of Chemical Physics*, 145, 10 2016. ISSN 00219606. doi: 10.1063/1.4964627.

[74] Haoyan Huo and Matthias Rupp. Unified representation of molecules and crystals for machine learning. *arXiv*, 4 2017. URL http://arxiv.org/abs/1704.06439.

[75] Thomas B. Blank, Steven D. Brown, August W. Calhoun, and Douglas J. Doren. Neural network models of potential energy surfaces. *The Journal of Chemical Physics*, 103:4129–4137, 9 1995. ISSN 0021-9606. doi: 10.1063/1.469597. URL http://aip.scitation.org/doi/10.1063/1.469597.

[76] Sönke Lorenz, Axel Groß, and Matthias Scheffler. Representing high-dimensional potential-energy surfaces for reactions at surfaces by neural networks. *Chemical Physics Letters*, 395:210–215, 9 2004. ISSN 00092614. doi: 10.1016/j.cplett.2004.07.076. URL https://linkinghub.elsevier.com/retrieve/pii/S000926140401125X.

[77] Jörg Behler and Michele Parrinello. Generalized neural-network representation of high-dimensional potential-energy surfaces. *Physical Review Letters*, 98, 4 2007. ISSN 00319007. doi: 10.1103/PhysRevLett.98.146401.

[78] Jörg Behler. Atom-centered symmetry functions for constructing high-dimensional neural network potentials. *Journal of Chemical Physics*, 134, 2 2011. ISSN 00219606. doi: 10.1063/1.3553717.

[79] J. S. Smith, O. Isayev, and A. E. Roitberg. Ani-1: an extensible neural network potential with dft accuracy at force field computational cost. *Chemical Science*, 8:3192–3203, 2017. ISSN 20416539. doi: 10.1039/C6SC05720A.

[80] Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87, 5 2013. ISSN 10980121. doi: 10.1103/PhysRevB.87.184115.

[81] Anders S. Christensen, Lars A. Bratholm, Felix A. Faber, and O. Anatole von Lilienfeld. Fchl revisited: Faster and more accurate quantum machine learning. *Journal of Chemical Physics*, 152, 1 2020. ISSN 00219606. doi: 10.1063/1.5126701.

[82] Felix A Faber, Anders S Christensen, Bing Huang, and O Anatole von Lilienfeld. Alchemical and structural distribution based representation for universal quantum machine learning. *The Journal of chemical physics*, 148:241717, 6 2018. ISSN 1089-7690. doi: 10.1063/1.5020710. URL http://www.ncbi.nlm.nih.gov/pubmed/29960351.

[83] Bastiaan J Braams and Joel M Bowman. Permutationally invariant potential energy surfaces in high dimensionality. *International Reviews in Physical Chemistry*, 28(4):577–606, 2009.

[84] Alex Brown, Anne B McCoy, Bastiaan J Braams, Zhong Jin, and Joel M Bowman. Quantum and classical studies of vibrational motion of ch 5+ on a global potential energy surface obtained from a novel ab initio direct dynamics approach. *The Journal of chemical physics*, 121(9):4105–4116, 2004.

[85] Joel M Bowman, BJ Braams, S Carter, C Chen, G Czakó, B Fu, X Huang, E Kamarchik, AR Sharma, BC Shepler, et al. Ab-initio-based potential energy surfaces for complex molecules and molecular complexes. *The Journal of Physical Chemistry Letters*, 1(12):1866–1874, 2010.

[86] Zhen Xie and Joel M Bowman. Permutationally invariant polynomial basis for molecular energy surface fitting via monomial symmetrization. *Journal of Chemical Theory and Computation*, 6(1):26–34, 2010.

[87] Bin Jiang and Hua Guo. Permutation invariant polynomial neural network approach to fitting potential energy surfaces. *The Journal of chemical physics*, 139(5):054112, 2013.

[88] Jun Li, Bin Jiang, and Hua Guo. Permutation invariant polynomial neural network approach to fitting potential energy surfaces. ii. four-atom systems. *The Journal of chemical physics*, 139(20):204103, 2013.

[89] K. T. Schütt, H. E. Sauceda, P. J. Kindermans, A. Tkatchenko, and K. R. Müller. Schnet - a deep learning architecture for molecules and materials. *Journal of Chemical Physics*, 148, 6 2018. ISSN 00219606. doi: 10.1063/1.5019779.

[90] K T Schütt, P.-J Kindermans, H E Sauceda, S Chmiela, A Tkatchenko, and K.-R Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems*, pages 922–1002, 2017. URL www.quantum-machine.org.

[91] Jacob Townsend, Justin K. Kirkland, and Konstantinos D. Vogiatzis. Chapter 3 - post-hartree-fock methods: configuration interaction, many-body perturbation theory, coupled-cluster theory. In S.M. Blinder and J.E. House, editors, *Mathematical Physics in Theoretical Chemistry*, Developments in Physical 'I&' Theoretical Chemistry, pages 63–117. Elsevier, 2019. ISBN 978-0-12-813651-5. doi: https://doi.org/10.1016/B978-0-12-813651-5.00003-6. URL https://www.sciencedirect.com/science/article/pii/B9780128136515000036.

[92] Matthew Welborn, Lixue Cheng, and Thomas F. Miller. Transferability in machine learning for electronic structure via the molecular orbital basis. *Journal of Chemical Theory and Computation*, 14:4772–4779, 9 2018. ISSN 15499626. doi: 10.1021/acs.jctc.8b00636.

[93] Lixue Cheng, Matthew Welborn, Anders S. Christensen, and Thomas F. Miller. A universal density matrix functional from molecular orbital-based machine learning: Transferability across organic molecules. *Journal of Chemical Physics*, 150, 4 2019. ISSN 00219606. doi: 10.1063/1.5088393.

[94] Tamara Husch, Jiace Sun, Lixue Cheng, Sebastian J.R. Lee, and Thomas F. Miller. Improved accuracy and transferability of molecular-orbital-based machine learning: Organics, transition-metal complexes, non-covalent interactions, and transition states. *The Journal of Chemical Physics*, 154:064108, 2 2021. ISSN 0021-9606. doi: 10.1063/5.0032362. URL https://aip.scitation.org/doi/abs/10.1063/5.0032362.

[95] Jacob Townsend and Konstantinos D. Vogiatzis. Data-driven acceleration of the coupled-cluster singles and doubles iterative solver. *Journal of Physical Chemistry Letters*, 10:4129–4135, 6 2019. ISSN 19487185. doi: 10.1021/acs.jpclett.9b01442.

[96] Jacob Townsend and Konstantinos D. Vogiatzis. Transferable mp2-based machine learning for accurate coupled-cluster energies. *Journal of Chemical Theory and Computation*, 16(12):7453–7461, 2020. doi: 10.1021/acs.jctc.0c00927. URL https://doi.org/10.1021/acs.jctc.0c00927. PMID: 33138363.

[97] Bruno Cuevas-Zuviría. Learning electron densities. In *Quantum Chemistry in the Age of Machine Learning*, pages 431–451. Elsevier, 2023.

[98] Jiang Wu, Guanhua Chen, Jingchun Wang, and Xiao Zheng. Redesigning density functional theory with machine learning. In *Quantum Chemistry in the Age of Machine Learning*, pages 531–558. Elsevier, 2023.

[99] Wolfram Koch and Max C Holthausen. *A chemist's guide to density functional theory*. John Wiley & Sons, 2015.

[100] Kanun Pokharel, James William Furness, Yi Yao, Volker Blum, Tom James Patrick Irons, Andrew Michael Teale, and Jianwei Sun. Exact constraints and appropriate norms in machine learned exchange-correlation functionals. *The Journal of Chemical Physics*, 2022.

[101] David J Tozer, Victoria E Ingamells, and Nicholas C Handy. Exchange-correlation potentials. *The Journal of chemical physics*, 105(20):9200–9213, 1996.

[102] Xiao Zheng, Li Hong Hu, Xiu Jun Wang, and Guan Hua Chen. A generalized exchange-correlation functional: The neural-networks approach. *Chemical Physics Letters*, 390:186–192, 5 2004. ISSN 00092614. doi: 10.1016/j.cplett.2004.04.020.

[103] Yi Zhou, Jiang Wu, Shuguang Chen, and GuanHua Chen. Toward the exact exchange–correlation potential: A three-dimensional convolutional neural network construct. *The journal of physical chemistry letters*, 10 (22):7264–7269, 2019.

[104] Ryo Nagai, Ryosuke Akashi, and Osamu Sugino. Completing density functional theory by machine learning hidden messages from molecules. *npj Computational Materials*, 6(1):1–8, 2020.

[105] Li Li, Stephan Hoyer, Ryan Pederson, Ruoxi Sun, Ekin D Cubuk, Patrick Riley, Kieron Burke, et al. Kohn-sham equations as regularizer: Building prior knowledge into machine-learned physics. *Physical review letters*, 126 (3):036401, 2021.

[106] Sebastian Dick and Marivi Fernandez-Serra. Machine learning accurate exchange and correlation functionals of the electronic density. *Nature communications*, 11(1):1–10, 2020.

[107] Ryo Nagai, Ryosuke Akashi, Shu Sasaki, and Shinji Tsuneyuki. Neural-network kohn-sham exchange-correlation potential and its out-of-training transferability. *The Journal of chemical physics*, 148(24):241737, 2018.

[108] Etienne Cuierrier, Pierre-Olivier Roy, Rodrigo Wang, and Matthias Ernzerhof. The fourth-order expansion of the exchange hole and neural networks to construct exchange–correlation functionals. *The Journal of Chemical Physics*, 157(17):171103, 2022.

[109] Li Li, John C Snyder, Isabelle M Pelaschier, Jessica Huang, Uma-Naresh Niranjan, Paul Duncan, Matthias Rupp, Klaus-Robert Müller, and Kieron Burke. Understanding machine-learned density functionals. *International Journal of Quantum Chemistry*, 116(11):819–833, 2016.

[110] Kun Yao and John Parkhill. Kinetic energy of hydrocarbons as a function of electron density and convolutional neural networks. *Journal of chemical theory and computation*, 12(3):1139–1147, 2016.

[111] Junji Seino, Ryo Kageyama, Mikito Fujinami, Yasuhiro Ikabata, and Hiromi Nakai. Semi-local machine-learned kinetic energy density functional with third-order gradients of electron density. *The Journal of chemical physics*, 148(24):241705, 2018.

[112] Finding density functionals with machine learning. *Physical Review Letters*, 108, 6 2012. ISSN 00319007. doi: 10.1103/PhysRevLett.108.253002.

[113] Pavlo Golub and Sergei Manzhos. Kinetic energy densities based on the fourth order gradient expansion: performance in different classes of materials and improvement via machine learning. *Physical Chemistry Chemical Physics*, 21(1):378–395, 2019.

[114] Junji Seino, Ryo Kageyama, Mikito Fujinami, Yasuhiro Ikabata, and Hiromi Nakai. Semi-local machine-learned kinetic energy density functional demonstrating smooth potential energy curves. *Chemical Physics Letters*, 734:136732, 2019.

[115] Ralf Meyer, Manuel Weichselbaum, and Andreas W Hauser. Machine learning approaches toward orbital-free density functional theory: Simultaneous training on the kinetic energy density functional and its functional derivative. *Journal of chemical theory and computation*, 16(9):5685–5694, 2020.

[116] Felix Brockherde, Leslie Vogt, Li Li, Mark E. Tuckerman, Kieron Burke, and Klaus Robert Müller. Bypassing the kohn-sham equations with machine learning. *Nature Communications*, 8, 12 2017. ISSN 20411723. doi: 10.1038/s41467-017-00839-3.

[117] James Kirkpatrick, Brendan McMorrow, David HP Turban, Alexander L Gaunt, James S Spencer, Alexander GDG Matthews, Annette Obika, Louis Thiry, Meire Fortunato, David Pfau, et al. Pushing the frontiers of density functionals by solving the fractional electron problem. *Science*, 374(6573):1385–1389, 2021.

[118] Bruno Cuevas-Zuviría and Luis F Pacios. Analytical model of electron density and its machine learning inference. *Journal of Chemical Information and Modeling*, 60(8):3831–3842, 2020.

[119] Bruno Cuevas-Zuviría and Luis F Pacios. Machine learning of analytical electron density in large molecules through message-passing. *Journal of Chemical Information and Modeling*, 61(6):2658–2666, 2021.

[120] Peter Bjørn Jørgensen and Arghya Bhowmik. Deepdft: Neural message passing network for accurate charge density prediction. *arXiv preprint arXiv:2011.03346*, 2020.

Glossary

k-simplex A generalization of a triangle or tetrahedron in k-dimensional space; the smallest polytope in k-dimensions. For example, a 0-simplex is a point, a 1-simplex a line segment, a 2-simplex a triangle, etc.. 52

adjacency matrix A matrix that captures the pairwise adjacency between all nodes of a graph. For a formal definition, please see Subsection 2.3.1.. 19, 20

adjacent Given a graph G with vertex set V and edge set E, two vertices, v_i and v_j, are adjacent if the edge v_iv_j is in E.. 17, 20

Atom-centered Symmetry Functions Atom-centered Symmetry Functions. 77

bag of bonds . 68, 72

bijection A map f between two sets, X and Y such that $f : X \to Y$ is one-to-one (injective) and onto (surjective).. 56, 57

Bottleneck distance A distance defined on persistence diagrams that yields the minimal matching between two persistence diagrams.. 56, 66

chemical graph A graph where the nodes and edges represent chemical properties. For example, a molecular graph (atoms as nodes and bonds as edges) is a chemical graph.. 17

correlation matrix . 74

Coulomb matrices Representation. 68

Coulomb matrix QC. 90

degree For some vertex v, the degree of v, $D(v)$, is the number of edges incident to v.. 18, 23–25

Dirac delta . 26, 74

distance matrix A matrix that captures the distance (such as the geometric or the topological) between two nodes.. 19

edge A link between two nodes, v_i and v_j, often written as $v_iv_j = e_{ij}$.. 17–20, 26, 48

electronegativity ability of an atom in a molecule to attract electrons to itself. 27

Euclidean distance The Euclidean distance, or l_2 norm, is a common distance metric. For a vector, $\mathbf{x} = (x_1, \ldots, x_n)$, the l_2 norm is defined as the length of a vector: $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$. For two points in $\mathbb{R}^n$ the distance between two points $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$ in as $\|x - y\|_2 = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$. . 68, 74

Euclidean space Let $\mathbb{R}$ be the set of real numbers. Euclidean space of dimension n is $\mathbb{R}^n$. In this work, we will be mainly concerned with $\mathbb{R}^2$; the real plane and $\mathbb{R}^3$; real space. . 20

exchange matrix . 90

Faber-Christensen-Huang-Lilienfeld Faber-Christensen-Huang-Lilienfeld. 83

filtration parameter Given a simplicial complex, a filtration parameter, $r \geq 0$, is the radius of the ball placed around each node in the simplicial complex and is allowed to vary over values greater than or equal to 0.. 53, 54, 61

Fock matrix . 90

graph An ordered pair, $G = (V, E)$, of a vertex set, $V = \{1, \ldots, n\}$, and an edge set of unordered pairs, $E = \{e_1, \ldots, e_m\}$ where $e_i = v_jv_k$.. 17–20, 22

graph neural network Neural networks that take graphs as input.. 87

graph theory A branch of mathematics that studies graphs, both directed and undirected.. 17, 20

homology A branch of mathematics that associates algebraic objects with other mathematical objects. In this work, we use homology as a way to understand the underlying structure of a space, such as the number of connected components, holes, and voids.. 52, 53, 61

hyperparameter External parameters that control the learning process and determine the model parameters values of the optimized learning algorithm.. 13, 85

incident A vertex v is incident with an edge e if v is one of the vertices that makes up the edge.. 17, 18

kernel ridge regression (Grier: Place a definition of ridge regression and a kernel in here... mention regularization). 72

Kronecker delta . 74

length The number of edges in a walk or path.. 19, 20

many-body tensor representation . 68, 74

matrix An m by n array of numbers arranged in rows and columns. Often denoted $A_{m,n}$ where an entry $a_{i,j}$ represents the element in the ith row and jth column.. 19, 20

molecular graph A graph where the nodes represent atoms and the edges represent the bonds between atoms.. 17–19, 22, 24–27, 30, 32, 39, 41–43, 45, 48, 51, 87

node A point which represents an object. Also referred to as a vertex. Usually connected to other nodes via edges.. 17, 42

path A walk in which no vertices or edges are repeated.. 18–20, 22

Permutationally invariant potentials . 86

persistence Given some homological feature with birth b and death d, the persistence of that feature is $d - b$.. 54, 55, 58, 59, 61, 62

persistence barcode A representation of persistence information where each homological feature is represented by a bar. The x-axis denotes the time the feature is born or dies and the y-axis counts the number of bars. The color of each bar denotes the dimension of the homological feature.. 54, 55, 61, 62

persistence diagram A representation of persistence information where each homological feature is represented by a point. The x-axis denotes the time the feature is born and the y-axis represents the time at which the feature dies. The color of each point denotes the dimension of the homological feature.. 54–58, 60, 65, 66

persistent homology A mathematical method used to capture the underlying structure of a space at different filtration parameters.. 51, 53, 56

probability distribution A function that gives the probability of different events or outcomes occurring. . 58, 59, 74

SchNet . 87

simplicial complex A set consisting of points, lines, faces, and their k-dimensional equivalents. Specifically, for some simplicial complex K, each face of a simplex in K is also in K and the non-empty intersection of any two simplices of K is a face (lower dimensional simplex) of both simplices.. 51, 52, 59

Smooth Overlap of Atomic Positions . 81

topological distance The smallest number of edges one must traverse to get from one node to another.. 20, 22, 23

traversal A walk that visits each vertex exactly once.. 18, 19

walk An alternating sequence of vertices and edges that begins and ends with a vertex such that each edge is incident with both of its defining vertices.. 18, 19

Wasserstein distance A distance defined on persistence diagrams we can visualize as how much effort it takes to move a mound of Earth from one place to another.. 56, 57, 66