COMPARISON OF NON-LEARNED AND LEARNED MOLECULE REPRESENTATIONS FOR CATALYST DISCOVERY

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Qianqian Yao

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2022

Fargo, North Dakota

North Dakota State University
Graduate School

Title

Comparison of Non-learned and Learned
Molecule Representations for Catalyst Discovery

By

Qianqian Yao

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Saeed Salem (co-advisor)

Dr. Mingao Yuan

Approved:

4/11/2022
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

Catalyst discovery is one very important task in storing renewable energy to address climate change and energy scarcity globally. Catalyst candidates can be represented by molecular descriptors and also can be modeled by graphs. Properties of catalyst candidates can be calculated by Density Functional Theory (DFT). Machine learning algorithms are applied to predict properties of catalyst candidates because DFT is computationally expensive. However, machine learning algorithms cannot operate over some standard molecular formats. Therefore, to represent molecules in the format that is required by machine learning algorithms is the primary task to tackle. Thus, this paper compared non-learned and learned representation methods. Accuracy of each representation method using RMSE are provided and discussed. The results show that the learned representations perform in a more stable manner than non-learned representations regardless of the linear models used for the downstream task.

ACKNOWLEDGMENTS

To my family who have always loved me unconditionally, to professors who have advised and tolerated me, to people I have got to know about, to whoever have came into my life and enriched my life experience, I appreciate all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1. Overview

Recently, there is more and more research about storing renewable energy to address climate change and energy scarcity around the world. Due to expected increases of industrialization and modernization, the expansion of renewable energy has been demanded more. However, with this expansion, a challenge arises: unlike traditional power sources such as coal, power provided by solar and wind is intermittent because the wind does not always blow and the sun also sets. Energy storage by Hydrogen Energy Storage (HES) [1]can be a good solution for this challenge.

HES offers a unique combination of scalability, long-term storage and portability. Specifically, HES can store vast quantities of energy cheaply in the form of hydrogen using the existing gas storage infrastructure. However, high costs of HES in its relatively low efficiency and the capital costs associated with systems for power conversion (electricity to hydrogen and back to electricity) have limited the broad application and adoption of HES. The power conversion systems for HES involve splitting water into its constituent parts (hydrogen and oxygen) using an electrolyzer[2] and converting hydrogen back to electricity using fuel cells[3] or other methods such as gas-fired turbines and engines. Both of these systems rely on materials called electrocatalysts[4] to increase the rate and efficiency of their respective electrochemical reactions. Unfortunately, common state-of-the-art electrocatalysts use expensive noble metals [5] such as iridium and

[1] https://www.sciencedirect.com/topics/engineering/hydrogen-energy-storage
[2] electrolyzer is a system that uses electricity to break water into hydrogen and oxygen in a process called electrolysis. https://en.wikipedia.org/wiki/Electrolysis
[3] https://en.wikipedia.org/wiki/Fuel_cell
[4] https://en.wikipedia.org/wiki/Electrocatalyst
[5] noble metals are metallic elements that show outstanding resistance to chemical attack even at high temperatures.

platinum. A major challenge is finding low-cost, efficient, and durable electrocatalysts for use for these reactions. If such catalysts can be discovered, earlier and wider adoption of HES systems could be achieved due to significant cost reductions.

Catalysts discovery by Volcano Plots have been conducted by many researchers in chemistry. In [1], the theory of Volcano plots[6] for catalysts discovery has been introduced. Volcano plots are efficient tools for optimizing catalytic reactions and correspondingly have found widespread use within many areas of catalysis, in which a descriptor variable (e.g., the binding energy of hydrogen) plotted along the x-axis and a measurement of catalytic activity (e.g., the experimental current density) plotted as y-axis. The appealing simplicity of the volcano concept led to numerous applications in the fields of electrocatalysis[7] and heterogeneous catalysis[8], largely because of Density Functional Theory (DFT)[9] computations, which provide direct access to the key energetic quantities needed to construct the plots, specifically the energy of the descriptor (x-axis), and a measure of activity (y-axis). Chapter 4 introduces the Volcano Plot and explains the theory of it.

Unfortunately, the DFT computation is extremely expensive and scales $O(n^3)$ with the number of electrons in the system. Thus, if we want to use DFT to calculate the energy of the descriptor, which is x-axis in the volcano plot so that the activity of the molecule can be predicted efficiently for large-scale exploration of new catalysts, efficient computational approximations of DFT are needed. One particularly promising approach is to use machine learning. Given a training dataset of DFT calculations, machine learning techniques have shown the ability to learn good

[6] https://en.wikipedia.org/wiki/Volcano_plot_(statistics)
[7] https://www.sciencedirect.com/topics/chemistry/electrocatalysis
[8] https://en.wikipedia.org/wiki/Heterogeneous_catalysis
[9] https://en.wikipedia.org/wiki/Density_functional_theory

approximations for certain problems. The training dataset for DFT calculations can be a list of molecules of cartesian coordinates and their energy by DFT calculations.

With the understanding of the background of catalyst discovery, we know that machine learning can be applied to a list of molecules of catalyst candidates to predict their energy calculated by DFT. Hence, we have converted the problem of catalyst discovery in chemistry into a machine learning prediction task. However, how to apply machine learning algorithms on molecules?

Because most machine learning algorithms cannot operate over standard molecular formats such as SMILES[10] or Connection Tables[11]. We first need to describe molecules in a form understandable by the machine learning algorithms. We call this new form its "representation"[12] and the process by which this form is obtained as "featurization".

Over the last few years, increasingly improved representations that progressively encode increasing amounts of physical information of molecules have been proposed. [2] focused on some conventional methods in Quantum Machine Learning (QML)[13] such as sorted Coulomb Matrix (CM), the Bag of Bonds (BoB), and the Spectrum of London and Axilrod-Teller-Muto potential (SLATM), all of which are based on molecule descriptors. [3] also explored many methods for molecular featurization: graph representation, simplified molecular input line entry system representation, molecular fingerprinting, coulomb matrix, bag of bonds, bonds angles representation, bonds in molecules, bonds angles nonbonds dihedrals, behler and parrinello

[10] SMILES, simplified molecular-input line-entry system, is a specification in the form of a line notation for describing the structure of chemical species using short ASCII strings. More descriptions is in Chapter 2.
[11] https://chem.libretexts.org/Courses/Intercollegiate_Courses/Cheminformatics_OLCC_(2019)/2._Representing_Small_Molecules_on_Computers/2.2%3A_Connection_Tables
[12] representation, featurization and embedding are used interchangeably in this paper
[13] https://en.wikipedia.org/wiki/Quantum_machine_learning

symmetry functions, ANI, weighted atom-centered symmetric functions, neighborhood density functions, smooth overlap of atomic positions, gaussian potential, machine learned features.

As the development of graph machine learning[14] (graph representation learning[15]), there also has been a rich body of work on node-level representations that turn each node in a graph into a vector preserving node-node proximity (similarity/distance) and edge-level representations turning each edge in a graph into a vector, and some work on graph-level representations that turn each graph into a vector. Since molecules can be modeled as graphs by regarding atoms as nodes and bonds as edges, all graph machine learning algorithms can be applied on molecule representations.

Although there is a lot of research on the representation methods, we don't know some basic universal principles on how to choose one representation method for a specific problem since the approach that selected to represent a molecule to construct a meaningful relationship between the representation and the molecule has a crucial impact on the learning curve for the downstream tasks.

In the evolution of research on featurizations, there are mainly two categories: non-learned methods such as hand engineered which are the conventional methods and learned methods which are automatically learned by the algorithms. Such automation by learned methods for featurization is the popular trend right now because of the development of deep learning. Just like convolutional neural network to learn latent features for images in computer vision, there are some research on how to learn latent features for graphs. However, there is very few research comparing the performances between non-learned representations and learned representations for molecules.

---

[14] graph machine learning is to apply machine learning algorithms on graph-structured dataset. This paper use graph machine learning and graph representation learning interchangeably.

[15] https://www.cs.mcgill.ca/~wlh/grl_book/files/GRL_Book.pdf

Thus, in order to find out how the methods in these two categories perform and whether there are obvious advantages of learned representations by graph machine learning algorithms over non-learned representations by molecule descriptors, the comparison among methods in these two categories is our main purpose for this paper by applying the representations to the downstream tasks[16] which is to predict the properties of molecules.

1.2. Outline

We have first illustrated the big picture of this paper in **Figure 1** and visualized a general pipeline to apply machine learning on molecules of catalyst candidates in **Figure 2**.

As shown in the rectangle boxes (in orange color) at the top of **Figure 1**, the motivation is to discover new catalysts for renewable energy storage by Hydrogen. There is a theory that the Volcano Plot can be used to infer whether a molecule is an ideal catalyst candidate or not by its property that is calculated by DFT. However, there are some limitations of DFT to calculate the properties for all catalyst candidates, so we propose to use machine learning algorithms to predict properties of catalyst candidates. And to represent molecules by different methods and to compare these methods are the core task in this paper. The rectangle boxes at the right side of **Figure 1** indicates the related work in Chapter 2 and the rectangle boxes at the bottom shows the applied methods in Chapter 3.

Since conventional molecule formats can not be used directly by machine learning algorithms, we first need to represent each molecule into a vector by multiple methods, as shown in **Figure 2**. After each molecule in dataset is represented as a vector, the feature matrix would be obtained. If there are L representation methods applied, there will be L feature matrices obtained.

---

[16] downstream task means the task you actually want to solve, and generally are the tasks of classification, regression and clustering for the conventional machine learnings
https://ai.stackexchange.com/questions/28410/which-tasks-are-called-as-downstream-tasks

The conventional machine learning algorithms which in this paper are linear models including the ordinary linear regression and Gaussian Naïve Bayes can be applied on these obtained feature matrices, the accuracies by RMSE are calculated to evaluate the performances of the representation methods based on downstream tasks.

Thus, the comparison of the performances of non-learned and learned molecule representations based on the accuracies of linear models for the downstream tasks is the focus of our paper. We have compared non-learned representation methods and learned representation methods by some experiments. We also have explored how embedded dimensions of feature matrix could affect the performance of downstream tasks. The libraries and packages are publicly available. The implementations in this paper have been introduced in detail in Chapter 3, Chapter 4, and Chapter 5. The results in our paper dependents on the libraries and packages used for implementations in some degree, because there is little space left to customize. However, such comparison still can shed light on the performances of representation methodologies and show effects of the embedded dimensions.

The paper is outlined as follows. Chapter 2 is about related work; Chapter 3 introduces applied models in detail; Chapter 4 introduces the computational details for the implementation in our work; Chapter 5 describes experiments and result analysis; Chapter 6 contains the conclusions and future work.

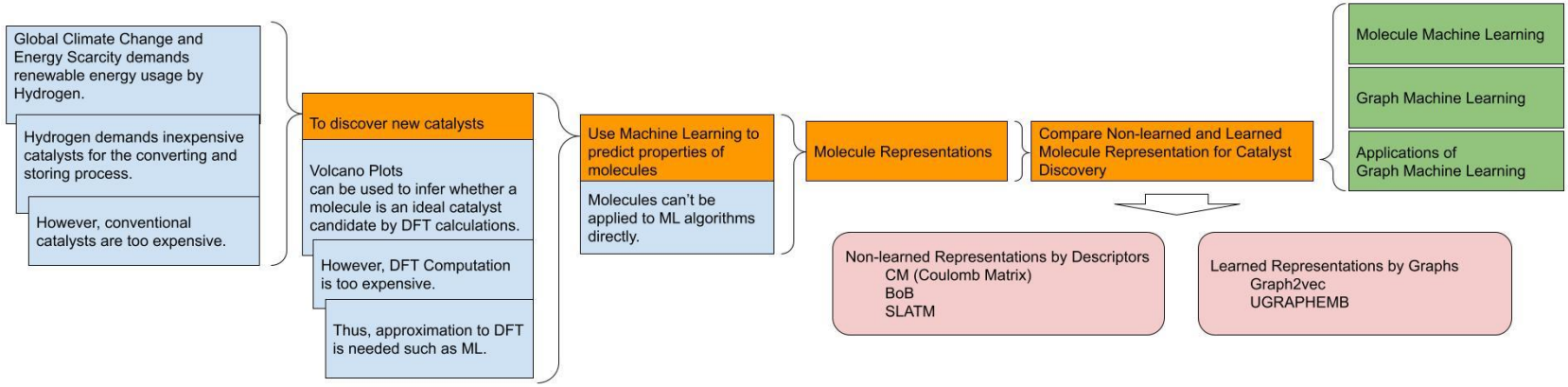


Figure 1: Big Picture

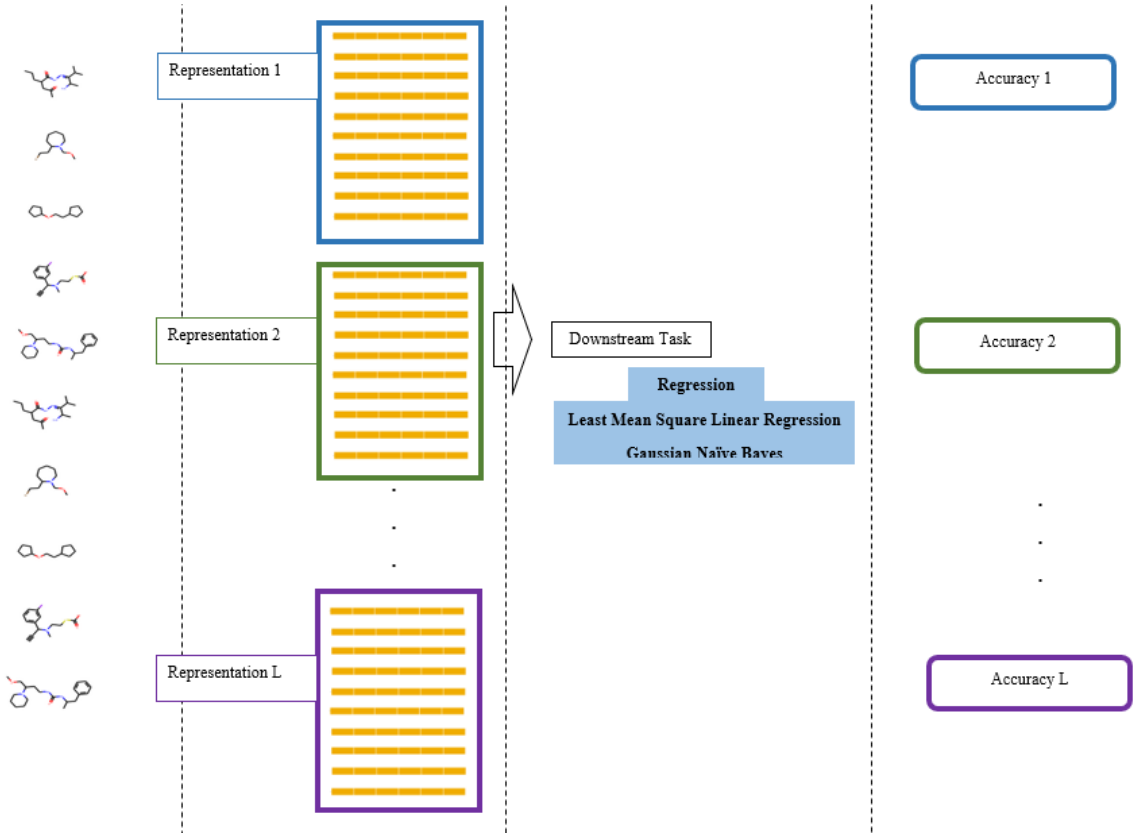


Figure 2: Pipeline

2. RELATED WORK

There is a lot of research that has been done related to this topic, which fall in the intersection of cheminformatics and graph machine learning. Thus, in this section we introduce related work from the aspects of molecular machine learning in Section 2.1, graph representation learning in Section 2.2, and applications of graph machine learning in Section 2.3.

2.1. Molecular Machine Learning

The intersection of machine learning with chemistry has been going on for decades. Molecular machine learning has seen tremendous growth in recent years and has also seen an increase in predictions about molecular properties. All machine learning approaches need data to learn. But before this data is fed to the machine learning algorithms, it is required to find a way to represent these molecule properties in the form of data, so that the machine can understand and learn. When it comes to molecules, it is represented as a bunch of atoms connected through chemical bonds, and machines are not capable of understanding this type of data. For better understanding, molecular data is being represented in the form of line notation and this system of representing the molecular data in a line notation is called SMILES – Simplified Molecular-Input Line-Entry System. SMILES [4] was first developed in the 1980s to represent molecules and it has since been modified and extended by others, most notably by Daylight Chemical Information Systems. In 2007, an open standard called "OpenSMILES" was developed by the Blue Obelisk open-source chemistry community. In July 2006, the IUPAC introduced the InChl [5] as a standard for formula representation. SMILES is generally considered to have the advantage of being more human-readable than InChl and it has also a wide base of software support with extensive theoretical backing such as graph theory.

[6] proposes the Extended-connectivity fingerprints (ECFP), a novel class of topological fingerprints for molecular characterization. [7] introduced Coulomb Matrix (CM) as feature of molecules and apply a machine learning model to predict atomization energies of a diverse set of organic molecules. The CM representation consists of a square atom by atom matrix, where the diagonal elements model the potential energies of free atoms while the off-diagonal elements correspond to the Coulomb nuclear repulsion between atom pairs. [8] proposed the Bag of Bonds (BoB) approach. In the BoB model, each molecule is represented as a vector composed of bags, where each bag represents a particular pair of elements (i.e. C-C, C-N, and so on), irrespective of electronic hybridization state. This approach builds on the Coulomb Matrix. [9] proposed Spectrum of London and Axilrod-Teller-Muto potential (SLATM). SLATM is based on the dissociative limits of intermolecular dispersion contributions between unpolarized moieties. They account for interatomic two-body terms through London's dispersion curve (rather than Coulomb), and for the three-body terms according to Axilrod-Teller-Muto.

[10] proposed graph convolutional network that operates directly on the graph, which could learn better representations than conventional method like ECFP. [11] combined graph convolutional network with residual LSTM embedding for one-shot learning on drug discovery. [12] applied neural network to predicate the molecular energy and reduced the predication error to 1 kcal/mol. [13] proposed atomic neural network to predicate the binding free energy of a subset of protein-ligand complexes found in the PDBBind dataset. [14] applied Massively multitask neural architectures to synthesize information from many distinct biological sources. [15] introduced MoleculeNet, a large-scale benchmark for molecular machine learning, which contains multiple public datasets, and establish metrics for evaluation and high quality open-source implementations. [1] introduced the genesis of molecular volcano plots and how it can be used for

catalyst discovery in cheminformatics. [2] applied volcano plots for computational discovery of cross-coupling catalysts by using molecule representations of CM, Bob and SLATM.

2.2. Graph Representation Learning

Graph-structured data is ubiquitous throughout the natural and social sciences, from telecommunication networks to quantum chemistry. In recent years, there has been an increasing interest in applying machine learning to graph-structured data. The primary objective is to automatically learn suitable representations to make predictions, discover new patterns, and understand complex dynamics in a better manner with respect to "traditional" machine learning approaches. For example, we might be interested in determining the role of a protein in a biological interaction graph, predicting the evolution of a collaboration network, recommending new products to a user in a social network, and many more. Due to their nature, graphs can be analyzed at different levels of granularity: at the node, edge, and graph level (the whole graph), as depicted in **Figure 3** [43]. Node-level embedding is to embed each node into a vector, Edge-level embedding is to embed each edge into a vector, and correspondingly graph level is to embed the whole graph into a vector. For each of those levels, different problems could be faced and thus specific algorithms should be applied.

There is a lot of research work about graph representation learning and some related work are as follows. [16] introduced representation on graphs comprehensively. Methods of direct encodings generalized encoder-decoder architectures, and subgraph embeddings by graph neural networks are talked about. [17] categorized graph representation learning methods into four types: shallow embeddings, auto-encoders, graph regularization, graph neural networks. [18] surveyed graph machine learning algorithms by categorizing methods into four categories which are graph signal processing based methods (sampling and recovery, learning topology structure), matrix

factorization based methods (graph Laplacian, matrix factorization, vertex proximity), random walks based methods (structure based random walk, random walk in heterogeneous networks, random walk in time-varying networks), deep learning based methods (graph convolution networks, graph attention networks, graph generative networks, graph spatial-temporal networks, graph auto-encoder).

In [19], deep learning on graphs have divided the existing methods into three main categories: semi-supervised methods including graph neural networks and graph convolutional networks, unsupervised methods including graph autoencoders, recent advancements including graph recurrent neural networks and graph reinforcement learning. A comprehensive survey on graph neural networks by [20] was based on four groups: recurrent graph neural networks, convolutional graph neural networks (spectral methods and spatial methods), graph autoencoders (network embedding and graph generation), spatial-temporal graph neural networks. This paper also offered insights on the brief history of graph neural networks.

[21] stated some limitations of some popular algorithms in graph representation learnings and the incentives to development graph neural networks. In the field of graph analysis, traditional machine learning approaches usually rely on hand engineered features and are limited by its inflexibility and high cost. Following the idea of representation learning and the success of word embedding [22], DeepWalk [23] is regarded as the first graph embedding method based on representation learning, applying skip-gram model on the generated random walks. Similar approaches such as node2vec [24], LINE [25], and TADW [26] also achieved breakthroughs. However, these methods suffer from two severe drawbacks. First, no parameter is shared between nodes in the encoder, which leads to computational inefficiency, since it means the number of parameters grows linearly with the number of nodes. Second, the direct embedding methods lack

the ability of generalization, which means they cannot deal with dynamic graphs or generalize to new graphs.

On the other hand, in the nineties, Recursive Neural Networks are first utilized on directed acyclic graphs [27] [28]. Afterwards, Recurrent Neural Networks and Feedforward Neural Networks are introduced respectively in [29] to tackle cycles. Although being successful, the universal idea behind these methods is building state transition systems on graphs and iterate until convergence, which constrains the extendibility and representation ability. Recent advancements of deep neural networks, especially convolutional neural networks (CNNs) [30], result in the rediscovery of GNNs. Extending deep neural models to non-Euclidean domains, which is generally referred to as geometric deep learning, has been an emerging research area [31]. Under this umbrella term, deep learning on graphs receives enormous attention.

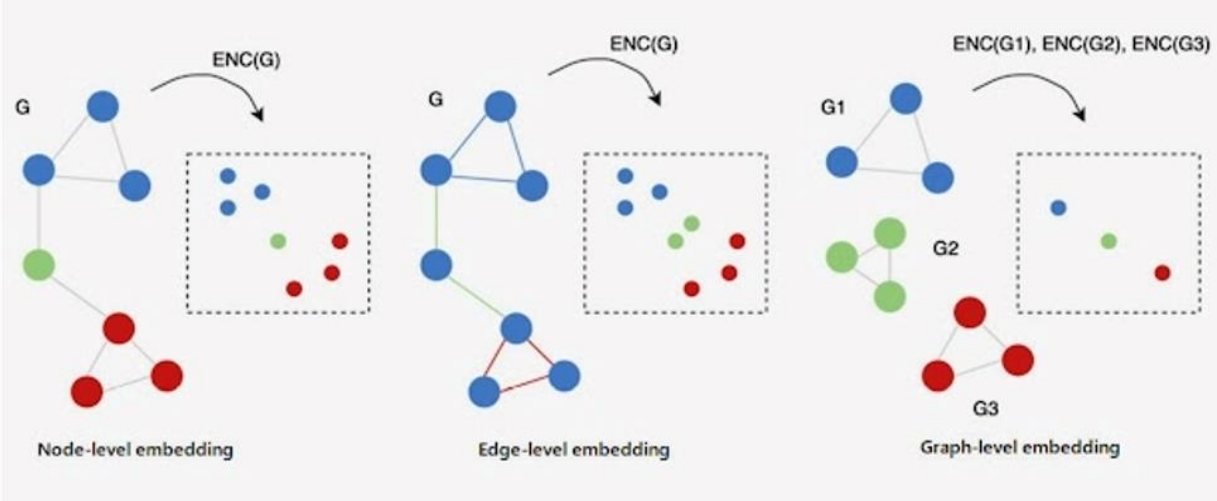


Figure 3: Visual Representation of the three different levels (source from [43])

2.3. Applications of Graph Machine Learnings

Graph machine learnings have been applied in many areas: biology and medicine, bioinformatics, cheminformatics, molecular property prediction, catalyst discovery, recommender systems, google mappings, trading networks, community detection and so on.

[32] explored applications of representation learning for networks in biology and medicine. They synthesize a spectrum of algorithmic approaches that leverage topological features to embed networks into compact vector spaces and provided a taxonomy of biomedical areas that are likely to benefit most from algorithmic innovation. [33] summarized the advances of graph representation learning and its representative applications in bioinformatics. [34] reviewed the discussions on how active research in molecular deep learning can address limitations of current descriptors and fingerprints while creating new opportunities in cheminformatics and virtual screening. The authors also provided a concise overview of the role of representations in cheminformatics, key concepts in deep learning, and argued that learning representation provides a way forward to improve the predictive modeling of small molecule bioactivities and properties. [35] structured the highly dynamic field of AI-research by collecting and classifying 80 GNNs that have been used to predict more than 20 molecular properties using 48 different datasets as graph neural networks are becoming more and more powerful and useful in the field of drug discovery. [36] provided a systematic review of graph learning recommender systems by discussing how they extract important knowledge from graph-based representations to improve the accuracy, reliability and explainability of the recommendations. [37] presented a graph neural network estimator for estimated time of arrival which is deployed in google maps. The main architecture consists of standard GNN building blocks and the usage of training schedule methods such as Meta-Gradients in order to make the model robust and production-ready. [38] have applied graph neural networks for modeling causality in international trade by introducing a method that measures causal scenarios during outlier events using neural networks. [39] have explored unsupervised training of graph neural network pooling for graph clustering.

This paper also is an application of graph machine learnings by modeling molecules by graphs and learning the features of molecules by graph machine learning algorithms that applied in this paper are Graph2vec and UGRAPHEMB. In order to verify any advantages of such learned features by graph models, we compare the methods that are non-learned with the methods that are learned in this paper. Chapter 3 introduces all the applied methods. Chapter 4 and Chapter 5 are about the computations of these methods and their applications on the downstream tasks.

3. APPROACHES

We have described the background for catalyst discovery and why it matters and the related research work in the previous sections. We also know that this problem can be solved by machine learning algorithms to predict the properties of molecules. Thus, the core task in this paper is to represent molecules into the format that can be used for machine learning algorithms by the methods that are non-learned and learned, with expectation to find out how different representation methods perform.

There are some differences between learned and non-learned representations as follows: non-learned representations are derived from molecules using heuristics or physics with the outputs arranged into feature vectors, while learned representations of molecules are intermediate, latent vectors produced by transformations learned directly from data by an ML model. Over the years, numerous molecular representations have been developed by several research groups working on Quantum Machine Learning (QML). There are also a lot of developments in the domain of graph representation learnings. However, it is not our focus to experiment on all of them, but only use the popular ones.

In this section, three methods of non-learned representations and two methods of Learned representations are described and applied, as shown in **Table 1**. First, the non-learned representation methods by molecular descriptors including Coulomb Matrix (CM) [7], Bag of Bonds (BoB) [8], Spectrum of London and Axilrod-Teller-Muto potential (SLATM) [9] are introduced in Section 3.1. Second, the learned representations by graphs including Graph2vec proposed by [40] and unsupervised inductive graph-level representation based on graph convolutional network (UGRAPHEMB) proposed by [41] are introduced in Section 3.2.

Table 1: Methods for Molecular Representations

Non-learned Representations by Descriptors			Learned Representations by Graphs	
CM	BoB	SLATM	Graph2vec	UGRAPHEMB

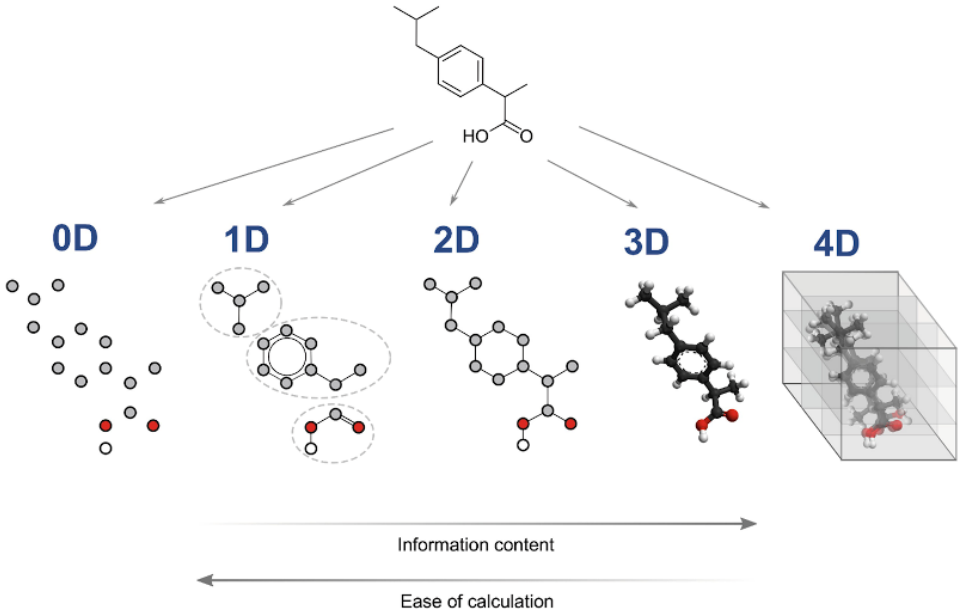


Figure 4: Theoretical Descriptors (source from [44])

3.1. Non-learned Representations by Descriptors

Molecular descriptors can be defined as mathematical representations of molecules' properties that are generated by algorithms. The numerical values of molecular descriptors are used to quantitatively describe the physical and chemical information of molecules. When selecting a descriptor to encode molecules, the descriptor should be correlated with the properties of the molecules that are to be predicted, and it can generate distinct values for structurally different molecules and can be adapted to the size of the molecules that are used in the machine learning algorithms.

Molecular descriptors can be classified into two categories: the experimental descriptors and the theoretical descriptors. Experimental descriptors are any physicochemical properties obtained by experimental measurements or by numerical simulations. Theoretical descriptors are derived from the symbolic representations of the molecules, such as the structural formula or the empirical formula. The five classes of theoretical descriptors, as shown in **Figure 4** [44], and the relationship between their dimensionality and the information they provide are explained below.

0D descriptors regroup all molecular descriptors that do not provide any information about the molecular structure or connectivity of atoms. For example, atom counts, bound counts, or molecular weights are 0D descriptors. one advantage of these descriptors is that they are easily obtained. However, these descriptors do not contain much information about the structural features of the molecules and, therefore, they are often combined with other descriptors. **1D descriptors** regroup the molecular descriptors that can be calculated from a set of substructures such as functional groups. The most common 1D descriptors are the fingerprints. For example, a fingerprint can be a binary vector where 1 indicates the presence of a structural feature and 0 its absence. 1D descriptors are also easily obtained. **2D descriptors** regroup all descriptors that provide information on molecular topology based on the graph representation of the molecules. Typical 2D descriptors are the adjacency matrix, the coulomb matrix or the distance matrix. In the case of the adjacency matrix, the descriptor indicates which atoms are bonded in a molecule. Because the 2D descriptors are sensitive to structural features of the molecule (size, shape and symmetry), they are a common choice as molecular descriptors. **3D descriptors** regroup all geometrical descriptors that provide information about the spatial coordinates of atoms of a molecule. The most well-known 3D descriptors are the molecular matrix and the 3D-MoRSE descriptors. In the case of the molecular matrix, the descriptor represents the cartesian coordinates

(x, y, z) of each atom. These descriptors provide a lot of information about molecules and have the advantage of differentiating isomeric molecules, which is not the case for all descriptors. However, because of their complexity, the geometrical descriptors can be time-consuming to calculate. **4D descriptors** are also called "grid-based descriptors". These descriptors, in addition to the molecular geometry, introduce a fourth dimension. This new dimension usually characterizes the interactions between the molecules and the active sites of a receptor or the multiple conformational states of the molecules. Common 4D descriptors are CoMFA and GRID. An advantage of the 4D descriptors is that they provide more information than the other descriptors and are always able to generate dissimilar values for structurally different molecules. However, like the 3D descriptors, the 4D descriptors are not easy to obtain because of their higher complexity.

The non-learned representation methods of CM, BoB, and SLATM that are applied in this paper are using 3D theoretical descriptors representing the cartesian coordinates (x, y, z) of each atom. Descriptions of these three methods are as follows.

3.1.1. Coulomb Matrix (CM)

The Coulomb Matrix is a simple global descriptor, which mimics the electrostatic interaction between nuclei. Coulomb matrix is calculated by **Equation (1)**, which is using the molecular information of the set of Cartesian coordinates, $\{R_i\}$, and nuclear charges, $\{Z_i\}$. The diagonal elements can be seen as the interaction of an atom with itself and are essentially a polynomial fit of the atomic energies to the nuclear charge. The off-diagonal elements represent the Coulomb repulsion between nuclei *i* and *j*.

$$M_{IJ} = \begin{cases} 0.5Z_I^{2.4} & for\ I = J \\ \dfrac{Z_I Z_J}{\left|\boldsymbol{R}_I - \boldsymbol{R}_J\right|} & for\ I \neq J \end{cases} \quad (1)$$

The Matrix in **Figure 5** [42] is the CM for methanol[17], as an example to visualize. The first row corresponds to carbon (C) in methanol interacting with all the other atoms (columns 2-5) and itself (column 1). Likewise, the first column displays the same numbers, since the matrix is symmetric. Furthermore, the second row (column) corresponds to oxygen, and the remaining rows (columns) correspond to hydrogen (H).

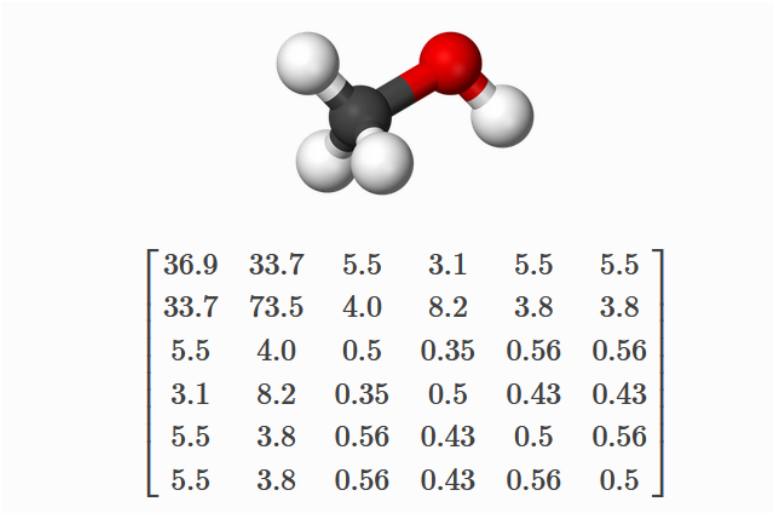
3.1.2. Bag of Bonds (BoB)

The "bag-of-bonds" (BoB) featurization is a variant of the Coulomb Matrix. It is inspired by the so-called "bag of words" descriptor used in natural language processing where a bag encodes the frequency of a particular word appearing in the text.

BoB follows a similar approach by making bags of different types of bonds (C-O, C-N, and so on), and order of the bond (single, double, triple). Each bag is basically a vector. Each element of such vector is computed as **Equation (2)**, where Z_i and Z_j are the nuclear charges, R_i and R_j are the positions of the atoms participating in a given bond. All bags are concatenated in a specific order and also are padded with zeros to make bags of equal sizes of all the molecules across the data set. This representation is invariant under rotation, translation and permutation. **Figure 6** [3] shows a visualization of BoB with an example of thiazole molecule (C3H3NS). In this figure, (A) is the 3D models, (B) is the Coulomb Matrix representation, and (C) is the atoms bags, bonds bags, and the concatenated bags.

$$Z_i Z_j \,/\, \left|R_i - R_j\right| \tag{2}$$

[17] Methanol, also known as methyl alcohol, amongst other names, is a chemical and the simplest alcohol, with the formula CH3OH



$$\begin{bmatrix} 36.9 & 33.7 & 5.5 & 3.1 & 5.5 & 5.5 \\ 33.7 & 73.5 & 4.0 & 8.2 & 3.8 & 3.8 \\ 5.5 & 4.0 & 0.5 & 0.35 & 0.56 & 0.56 \\ 3.1 & 8.2 & 0.35 & 0.5 & 0.43 & 0.43 \\ 5.5 & 3.8 & 0.56 & 0.43 & 0.5 & 0.56 \\ 5.5 & 3.8 & 0.56 & 0.43 & 0.56 & 0.5 \end{bmatrix}$$

Figure 5: Visualization of CM (source from [42])[18]

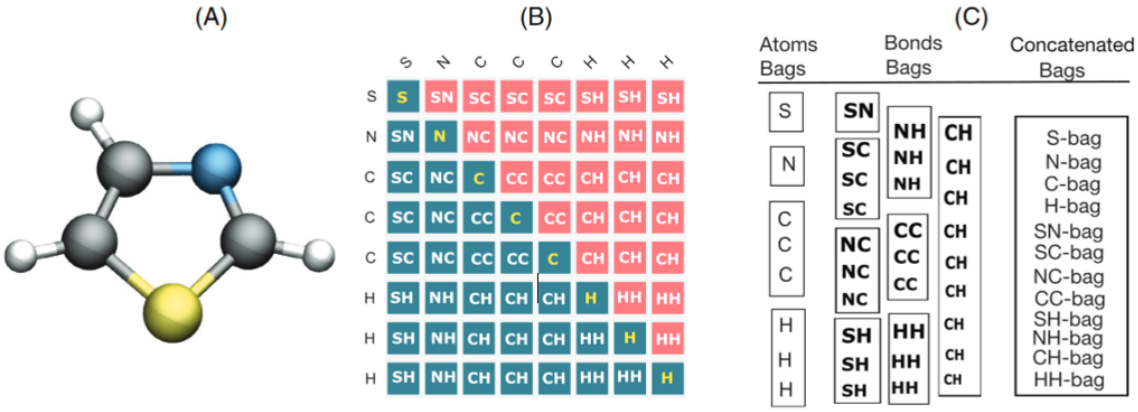


Figure 6: Visualization of BoB (source from [3])[19]

3.1.3. Spectrum of London and Axilrod-Teller-Muto (SLATM)

SLATM is based on the dissociative limits of intermolecular dispersion contributions between unpolarized moieties. They account for interatomic two-body term through London's dispersion curve (rather than Coulomb), and for the three-body terms according to Axilrod-Teller-

[18] Methanol: CH3OH

[19] (A) ball and stick representation of the thiazole molecule (C3H3NS), (B) Coulomb matrix elements, (C) different Coulomb matrix entries (off-diagonal) are sorted into different bags. Here, Concatenated bags are not padded with zeros for clarity

Muto. It has two variants: a local and a global one. The basic idea of SLATM is to represent an atom indexed I in a molecule by accounting for all possible interactions between atom I and its neighboring atoms through many-body potential terms multiplied by a normalized Gaussian distribution centered on the relevant variable (distance or angle). 1-body, 2-body and 3-body can be considered for different versions of representations. The 1-body term is simply represented by the nuclear charge, while the 2-body part is expressed as **Equation (3)**. The 3-body distribution reads as **Equation (4)**.

$$\frac{1}{2} Z_I \sum_{J \neq I} Z_J \delta\left(\mathbf{r} - \mathbf{R}_{IJ}\right) g(\mathbf{r}) \tag{3}$$

where $\delta(\cdot)$ is set to normalized Gaussian function as $\delta(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2}$; $g(r)$ is a distance dependent scaling function, capturing the locality of chemical bond and chosen to correspond to the leading order term in the dissociative tail of the London potential $g(R) = \frac{1}{R^6}$.

$$\frac{1}{3} Z_I \sum_{J \neq K \neq I} Z_J Z_K \delta(\theta - \theta_{IJK}) h(\theta, R_{IJ}, R_{IK}) \tag{4}$$

where θ is the angle spanned by vector R_{IJ} and R_{IK} (i.e., θ_{IJK}), and treated as a variable. $h(\theta, R_{IJ}, R_{IK})$ is the 3-body contribution depending on both internuclear distance and angle and is chosen in form to model the Axilrod-Teller-Muto Axilrod and Teller (1943), as **Equation (5)**.

$$h\left(\theta, \mathbf{R}_{IJ}, \mathbf{R}_{IK}\right) = \frac{1 + \cos\theta \cos\theta_{JKI} \cos\theta_{KIJ}}{\left(R_{IJ} R_{IK} R_{KJ}\right)^3} \tag{5}$$

3.2. Learned Representations by Graphs

In graph-based models, molecules can be represented as undirected graphs with atoms as graph nodes and bonds as graph edges. The basic chemical information encoded by molecular graphs, and then graph machine learning algorithm is used to train a model and to learn the representation of molecules.

Most existing methods for graph-level representations assume a supervised model and few are unsupervised. There are some limitations on the usability of supervised graph-level embeddings because the representations that are learned are specific to one particular machine learning task and cannot be used or transferred to other tasks or problems. In this paper, our work is to embed each graph into a feature vector with assumption of that molecular representations are independent with downstream tasks, so unsupervised graph-level representation learnings are applied. The applied unsupervised graph-level representation learning methods of Graph2vec, UNGRAPHEMB in this paper are described by the following.

3.2.1. Graph2vec

Graph2vec is inspired by the neural document embedding models. These document embedding models exploit the way how words/word sequences compose documents to learn the embeddings, similarly, graph2vec is to view an entire graph as a document and the rooted subgraphs around every node in the graph as words that compose the document and extend document embedding neural networks to learn representations of entire graphs. By the following we introduce the models of word2vec, doc2vec and graph2vec in detail.

3.2.1.1. Word2vec

Word2vec uses a simple and efficient feed forward neural network architecture called “skip-gram” to learn distributed representations of words. Word2vec works based on the rationale that the words appearing in similar contexts tend to have similar meanings and hence should have similar vector representations. To learn a target word’s representation, this model exploits the notion of context, where a context is defined as a fixed number of words surrounding the target word.

To this end, given a sequence of words $\{w_1, w_2, \ldots, w_t, \ldots, w_T\}$, the target word w_t whose representation has to be learnt, and the length of the context window *c*, the objective of skip-gram model, is to maximize the likelihood of **Equation (6)**.

$$\sum_{t=1}^{T} log\ Pr(w_{t-c}, \ldots, w_{t+c} \mid w_t) \tag{6}$$

where $t = 1, 2, \ldots, T$, and $w_{t-c}, \ldots, w_{t+c}$ are the context of the target word w_t. The probability $Pr(w_{t-c}, \ldots, w_{t+c})$ is computed as

$$\prod_{-c \leqq j \leqq c,\ j \neq 0} Pr\left(w_{t+j} \middle| w_t\right) \tag{7}$$

Furthermore, $Pr\left(w_{t+j} \middle| w_t\right)$ is defined as:

$$\frac{exp\left(\vec{w}_t \cdot \vec{w}'_{t+j}\right)}{\sum_{w \in \mathcal{V}} exp(\vec{w}_t \cdot \vec{w})} \tag{8}$$

where $\vec{w}$ and $\vec{w}'$ are the input and output vectors of word *w* and $\mathcal{V}$ is the vocabulary of all the words.

3.2.1.2. Doc2vec

Doc2vec is a straightforward extension to word2vec from learning embeddings of words to those of word sequences. Given a set of documents $D = d_1, d_2, \ldots, d_N$ and a sequence of words $c(d_i) = w_1, w_2, \ldots, w_{l_i}$ sampled from document $d_i \in D$, doc2vec skipgram learns a δ dimensional embeddings of the document $d_i \in D$ and each word w_j sampled from $c(d_i)$ i.e., $\vec{d}_i \in R^{\delta}$ and $\vec{w}_i \in R^{\delta}$, respectively.

The model works by considering a word $w_j \in c(d_i)$ to be occurring in the context of document d_i and tries to maximize the log likelihood in **Equation (9)**.

$$\sum_{j=1}^{l_i} log\ Pr\left(w_j \middle| d_i\right) \tag{9}$$

where the probability $Pr(w_j|d)$ is defined as:

$$\frac{exp(\vec{d} \cdot \vec{w}_j)}{\sum_{w \in \mathcal{V}} exp(\vec{d} \cdot \vec{w})} \tag{10}$$

where $\mathcal{V}$ is the vocabulary of all the words across all documents in *D*.

3.2.1.3. Graph2vec

Graph2vec is to view an entire graph as a document and each of its subgraphs, generated as an ego graph of each node, as words that comprise the document. In other words, a graph is composed of subgraphs as a document is composed of sentences. According to this description, the algorithm can be summarized into the following steps:

1). Subgraph generation: a set of rooted subgraphs is generated around every node.

2). Doc2vec training: the doc2vec skip-gram is trained using the subgraphs generated by the previous step.

3). Embedding generation: the information contained in the hidden layers of the trained doc2vec model is used in order to extract the embedding of each node.

Pseudocodes for graph2vec are shown in **Figure 7** and **Figure 8** [40]. As shown in **Figure 7**, we can see that the only required input is a corpus of graphs for Graph2vec to learn their representations. Given a dataset of graphs, Graph2vec considers the set of all rooted subgraphs (i.e., neighborhoods) around every node (up to a certain degree) as its vocabulary. Subsequently, representations of each graph is learned by following the doc2vec skipgram training process. The WL relabeling strategy is used to extract rooted subgraphs and assign a unique label for all the rooted subgraphs in the vocabulary during the training process of the skipgram model, as shown in **Figure 8**.

Algorithm 1: GRAPH2VEC $(\mathbb{G}, D, \delta, \mathfrak{e}, \alpha)$

input : $\mathbb{G} = \{G_1, G_2, ..., G_n\}$: Set of graphs such that each graph $G_i = (N_i, E_i, \lambda_i)$ for which embeddings have to be learnt
D: Maximum degree of rooted subgraphs to be considered for learning embeddings. This will produce a vocabulary of subgraphs, $SG_{vocab} = \{sg_1, sg_2, ...\}$ from all the graphs in $\mathbb{G}$
δ: number of dimensions (embedding size)
$\mathfrak{e}$: number of epochs
α: Learning rate

output: Matrix of vector representations of graphs $\Phi \in \mathbb{R}^{|\mathbb{G}| \times \delta}$

```
begin
    Initialization: Sample Φ from R^{|G|×δ}
    for e = 1 to 𝔢 do
        𝔊 = SHUFFLE (𝔾)
        for each G_i ∈ 𝔊 do
            for each n ∈ N_i do
                for d = 0 to D do
                    sg_n^(d) := GETWLSUBGRAPH(n, G_i, d)
                    J(Φ) = −log Pr (sg_n^(d) | Φ(G))
                    Φ = Φ − α ∂J/∂Φ
    return Φ
```

Figure 7: Graph2vec (source from [40])

Algorithm 2: GETWLSUBGRAPH (n, G, d)

input : n: Node which acts as the root of the subgraph
$G = (N, E, \lambda)$: Graph from which subgraph has to be extracted
d: Degree of neighbours to be considered for extracting subgraph

output: $sg_n^{(d)}$: Rooted subgraph of degree d around node n

```
begin
    sg_n^(d) = {}
    if d = 0 then
        sg_n^(d) := λ(n)
    else
        N_n := {n' | (n, n') ∈ E}
        M_n^(d) := {GETWLSUBGRAPH(n', G, d − 1) | n' ∈ N_n}
        sg_n^(d) := sg_n^(d) ∪ GETWLSUBGRAPH
          (n, G, d − 1) ⊕ sort(M_n^(d))
    return sg_n^(d)
```

Figure 8: GetWLSubgraph (source from [40])

3.2.2. Unsupervised Graph-level Embedding (UGRAPHEMB)

Graph2vec's embeddings are learnt in an unsupervised manner and are task agnostic, but it is transductive (non-inductive), i.e. it does not naturally generalize to unseen graphs outside the training set. Unsupervised Graph-Level Embedding (UGRAPHEMB) has improved such limitation of Graph2vec and it is a general framework that provides a novel means to perform graph-level embedding in a completely unsupervised and inductive manner. The learned neural network can be considered as a function that receives any graph as input, either seen or unseen in the training set and transform it into an embedding.

The philosophy of UGRAPHEMB is to generate one embedding per graph from node embeddings using a novel mechanism called Multi-Scale Node Attention (MSNA) and computes the proximity of using the two graph-level embeddings. UGRAPHEMB employs multi-scale aggregations of node-level embeddings, guided by the graph-graph proximity defined by well-accepted and domain-agnostic graph proximity metrics such as Graph Edit Distance and Maximum Common Subgraph. The goal of this algorithm is to learn high-quality graph-level representations in a completely unsupervised and inductive fashion: During training, it learns a function that maps a graph into a universal embedding space best preserving graph-graph proximity, so that after training, any new graph can be mapped to this embedding space by applying the learning function.

3.2.2.1. GCN based neighbor aggregation for node embedding

Multi-Scale Node Attention (MSNA) mechanism denote the input node embeddings of graph *G* as $U_G \in R^{N \times D}$, where the n-th row, $u_n = R^D$ is the embedding of node *n*. The graph level embedding is obtained as shown in **Equation (11)**.

$$h_G = MLP_w\left(\|_{k=1}^{K}\ \mathrm{ATT}_{\Theta_{(k)}}(U_G)\right) \tag{11}$$

where ‖ denotes concatenation, *K* denotes the number of neighbor aggregation layers, *ATT* denotes the following multi-head attention mechanism that transforms node embeddings into a graph-level embedding, and MLP_w denotes multi-layer perceptrons with learnable weights *W* applied on the concatenated attention results.

The intuition behind Equation (11) is that instead of only using the node embeddings generated by the last neighbor aggregation layer, the node embeddings generated by each of the *K* neighbor aggregation layers were used. *ATT* is defined as **Equation (12)**.

$$ATT_\Theta(U_G) = \sum_{n=1}^{N} \sigma\left(u_n^T ReLU\left(\Theta\left(\frac{1}{n}\sum_{m=1}^{N} u_m\right)\right)\right) u_n \tag{12}$$

where *N* is the number of nodes, σ is the sigmoid function $\sigma(x) = 1/1 + exp(-x)'$, and $\Theta^{(k)} \in R^{D\times D}$ is the weight parameters for the *k-th* node embedding layer.

The intuition behind Equation (12) is that during generation of graph-level embeddings, the attention weight assigned to each node should be adaptive to the graph proximity metric. To achieve that, the weight is determined by both the node embedding u_n, and a learnable graph representation. The learnable graph representation is adaptive to a particular graph proximity via the learnable weight matrix $\Theta^{(k)}$.

3.2.2.2. Unsupervised Loss by inter-graph proximity preservation

UGRAPHEMB uses graph-graph proximity, so it is important to select an appropriate graph proximity (similarity/distance) metric. Once the proximity metric is defined, and the graph-level embeddings for G_i and G_j are obtained, denoted as h_{G_i} and h_{G_j}, the similarity/ distance between the two graphs can be computed then.

Multidimensional scaling is to embed data points in a low dimensional space so that their pairwise distances are preserved via minimizing the loss function as **Equation (13)**.

$$L(h_i, h_j, d_{ij}) = \left(\| h_i - h_j \|_2^2 - d_{ij}\right)^2 \tag{13}$$

where h_i and h_j are the embeddings of points i and j, and d_{ij} is their distance.

Based on Equation (10) the difference between the predicted distance and the ground-truth distance can be minimized as **Equation (14) (15)**. If the metric is about similarity, the loss function can be used as **Equation (16) (17)**.

Distance:

$$\mathrm{L} = E_{(i,j)\sim D}\left(\hat{d}_{ij} - d_{ij}\right)^2 \tag{14}$$

$$\mathrm{L} = E_{(i,j)\sim D}\left(\| h_{G_i} - h_{G_j} \|_2^2 - d_{ij}\right)^2 \tag{15}$$

Similarity:

$$\mathrm{L} = E_{(i,j)\sim D}\left(\hat{s}_{ij} - s_{ij}\right)^2 \tag{16}$$

$$\mathrm{L} = E_{(i,j)\sim D}\left(h_{G_i}^T h_{G_j} - s_{ij}\right)^2 \tag{17}$$

where (i, j) is a graph pair sampled from the training set, and d_{ij} is the distance and s_{ij} is the similarity between them.

4. COMPUTATIONAL DETAILS

This chapter describes the data set and the linear models including least mean square linear regression and Gaussian Naïve Bayes for downstream task which is to predict the property of molecule. The method of measuring accuracy of the linear models is introduced. Dimension reduction method on representations, which is to make all representations with same dimensions before they are fed into the linear models, is also explained. The theories of volcano plot are also explained in detail. At the end, the implementation tools for experiments in Chapter 5 are listed.

4.1. Dataset Description

QML, a python toolkit for quantum machine learning, provided implementations of Coulomb Matrix, Bag of bonds, and SLATM. The dataset for our project is from QML github repository, which is the Cartesian Coordinates for each molecule. The downloaded folder contains Cartesian Coordinate files, as shown in **Figure 9.** The left side in Figure 9 shows samples and the right side shows y-labels of the samples. Only top ten files are listed for the purpose of demonstration. Each sample is a txt file that contains the Cartesian Coordinates of each atom, shown in **Figure 10**.

Each sample of txt file corresponds to numerical number of y-labels, which indicates the energy of the molecule calculated by Density Functional Theory (DFT). The value of the DFT energy of each molecule can be further used to indicate whether this molecule is an ideal catalyst candidate by looking up in a Volcano Plot which should be provided by chemistry scientists. The theory of indicating whether a molecule is an ideal catalyst candidate or not by Volcano Plot is introduced further in Section 4.3.

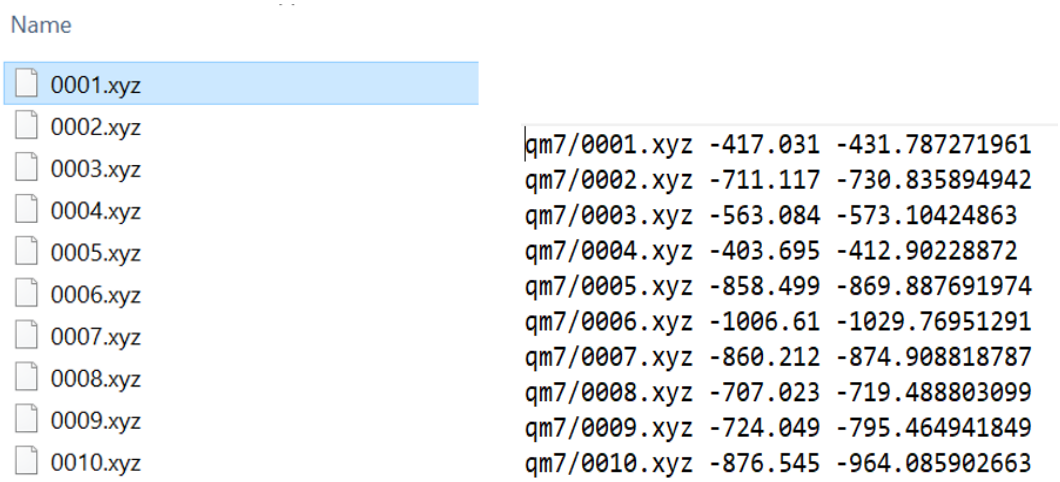


Figure 9: Data Set

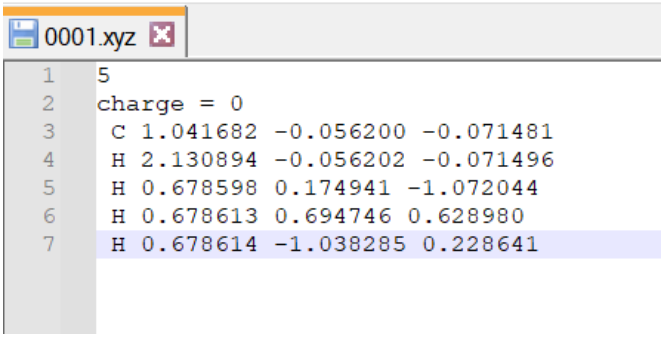


Figure 10: Cartesian Coordinate File

4.2. Linear Models[20] for Downstream Task

Downstream task is the task we actually want to solve, which in this paper is to predict the properties of catalyst candidates. Once representations of catalyst candidates are obtained, we can move on to predicting task by some machine learning algorithms.

Because the y-labels of the molecules are the values of DFT energy of the Cartesian Coordinate files, which are continuous, this is a linear regression problem. Linear models expressed as **Equation (18)** are applied to such linear regression problem which in this paper is to

[20] https://scikit-learn.org/stable/modules/linear_model.html#

predict the molecular property. Library of Scikit Learn contains most of the commonly used linear algorithms. Besides linear models in machine learning, there are many other algorithms can be applied such as [45] compared two fusion methods of ensemble method and Choquet fuzzy integral method. However, which machine learning methods to use and how different algorithms perform on this downstream task is not the purpose of this paper. Given linear models are selected, how different the linear models perform is also not the focus of this paper. Thus, some popular algorithms of linear model are applied in this paper by convenience.

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{p} X_j \hat{\beta}_j \tag{18}$$

where $X^T = (X_1, X_2, \dots, X_p)$. If 1 is included in X, the above equation can be presented as **Equation (19)**, where $X^T = (1, X_1, X_2, \dots, X_p)$.

$$\hat{Y} = X^{\top} \hat{\beta} \tag{19}$$

We know that the difference between the linear models is that which decision theory[21] is applied, such as least square, K nearest neighbors, probabilities and so on. In this paper, we only choose Least Mean Square Linear Regression and Gaussian Naïve Bayes, which are the popular choices for linear regression tasks. The two models are related by the way that the probabilistic interpretation can further verify that least squares is a reasonable choice. The two methods are introduced as below mathematically.

The standard linear regression model is shown in **Equation (20)**. The Least square Loss function is given in **Equation (21)**. We want to choose $\beta_0, \beta_1, \dots, \beta_p$ so as to minimize L, which can be done by the gradient descent algorithm, which starts with some initial $\hat{\beta}$, and repeatedly performs the updates as **Equation (22)** until convergence. α is called the learning rate.

[21] https://en.wikipedia.org/wiki/Decision_theory

$$Y = \beta_0 + \beta_1 X_1 + \ ... + \beta_p X_p + \varepsilon \tag{20}$$

$$L = \frac{1}{2}\sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \tag{21}$$

$$\hat{\beta} := \hat{\beta} - \alpha \cdot \frac{\partial L}{\partial \hat{\beta}} \tag{22}$$

Bayes' Theorem is shown as **Equation (23)**, given a sample $\vec{x}$, the probability this sample belongs to a group can be calculated by the division of Joint probability of $(\vec{x}, \mathrm{y}_i)$ and the prior probability of $\vec{x}$. According to the probability chain rule, the joint probability can be calculated by **Equation (24)**. According to total law of probability, the denominator in Equation (23) can be calculated as in **Equation (25)**.

$$P(\mathrm{y}_i|\vec{x}) = P(y_i, \vec{x})/P(\vec{x}) \tag{23}$$

$$P(y_i, \vec{x}) = P(y_i)P(\vec{x}|y_i) \tag{24}$$

$$P(\vec{x}) = \sum_{j=1}^{c} P(y_j)P(\vec{x}|y_j) \tag{25}$$

The denominator is the same irrespective of the class to which $\vec{x}$ may belong, so it can be ignored. Hence, the final simplified version of the Bayes' Classifier can be written as in **Equation (26)**.

$$P(y_i|\vec{x}) \sim P(y_i) * P(\vec{x}|y_i) \tag{26}$$

Naïve Bayes assumes that features are conditionally independent. Thus, the Naïve Bayes' Classifier can be written as in **Equation (27)**, assuming that $P(x_j|\mathrm{y}_i)$ follow a Gaussian distribution[22], which is shown as **Equation (28)**.

$$P(y_i|\vec{x}) = P_i \times P(x_1|y_i) \times P(x_2|y_i) \times \ldots \times P(x_n|y_i) \tag{27}$$

[22] https://en.wikipedia.org/wiki/Normal_distribution

$$f(y_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i-\mu}{\sigma}\right)^2} \quad (28)$$

The y_i which has maximum likelihood value of $P(y_i|\vec{x})$ will be taken as the prediction of the sample $\vec{x}$.

4.3. Measuring the Quality of Fit

In order to evaluate the performance of a machine learning model on a given dataset, we need some way to measure how well its predictions actually match the observed data. That is, we need to quantify the extent to which the predicted response value for a given observation is close to the true response value for that observation.

In the regression setting, the most commonly-used measure is the mean squared error (MSE), given by **Equation (29)**.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (29)$$

where $\hat{y}_i$is the prediction for the i-th observation. The MSE will be small if the predicted responses are very close to the true response, and will be large if for some of the observations, the predicted and true responses differ substantially.

RMSE is the square root of the MSE. Comparing to MSE that is measured in squared units of the response variable, RMSE is more often used because it is measured in the same units as the response variable. Thus, RMSE is used in this paper for the evaluations.

4.4. Dimensions Reduction of Representation Matrices

Five embedding algorithms in **Table 1** embed graphs into matrices with different dimensions. **Table 2** in Chapter 5 shows the results of the embedded features from each method. In order to better compare performances of these embeddings from the same level with the same

dimensions, dimension reduction will be applied first before the embeddings are fed to the linear model using Principle Component Analysis (PCA).

Dimension reduction involves projecting the n features(predictors) into a d dimensional subspace, where $d < n$. This is achieved by computing d different linear combinations, or projections of the variables. Then, these d projections are used as predictors to fit a linear regression model by least squares.

There are many algorithms existing to perform dimension reduction such as (Principle Component Analysis) PCA, (Singular Value Decomposition) SVD, (Linear Discriminant Analysis) LDA and many others. Different dimension reduction algorithms may affect the accuracies of linear models in the different levels, but we do not go deeper on this topic in this paper. Thus, one of the popularly used dimension reduction of PCA is used by convenience.

PCA is a multivariate technique that analyze data in which observations are described by several inter-correlated quantitative variables. Specifically, if a data set has n features, then PCA can be used to project these n features on to a d dimensional space. Dimensional reduction will improve prediction accuracy and computational efficiency. PCA is also very useful for data visualization. The calculations of PCA are described as below given a data set $\Omega = [x^{(1)}, x^{(2)}, \ldots, x^{(n)}], \Omega \in \mathbb{R}^{m \times n}, x^{(i)} \in \mathbb{R}^m$, where m is the number of observations and n is the number of features.

Step-1: evaluate $\mu_i = mean(x^{(i)}); \mu \in \mathbb{R}^n$

Step-2: replace $x^{(i)}$ with $x^{(i)} - \mu_i \times ones(m, 1)$

Step-3: now $x_{new}^{(i)} = x^{(i)} - \mu_i \times ones(m, 1)$, variance $\sigma_i = st.dev.(x_{new}^{(i)})$

Step-4: $x_{new}^{(i)} = x_{new}^{(i)} / \sigma_i$

Step-5: evaluate $\lambda_{i=1:n}$ and $e_{i=1:n}$ for Ω, which is taken through steps 1-4

Step-6: the eigenvalues should be ordered in descending order

Step-7: $Y_{m\times d} = \Omega_{m\times n} E_{n\times d}$,

where $E = [e_1,\ e_2,\ \ldots,\ e_d], e_{i=1:d} \in \mathbb{R}^n$ are the eigenvectors corresponding to $\lambda_1 \ldots \lambda_d$, which are the first d largest eigenvalues in descending order.

$Y_{m\times d}$ is the data in reduced dimensions PCA is not a feature selection method. Instead, it can provide with the directions in which the data has the maximum variance. The information along this direction is important as it actually captures the maximum variance in the data.

4.5. Volcano Plot for Catalysts Discoveries

Volcano Plot is derived from Sabatier's principle which states that an ideal catalyst should neither bind the products too strong nor bind the reactants too weak. As in modern volcano plots, a descriptor variable (e.g., the binding energy of hydrogen) is plotted along the x-axis and a measurement of catalytic activity (e.g., the experimental current density) is plotted as y-axis. Hence, in line with Sabatier's principle, the resulting volcano shape demonstrates a clear relationship between the value of the descriptor variable and catalytic activity. The Volcano Plot is provided by researchers in chemistry for any related machine learning tasks.

The volcano shape in **Figure 11** [1], can be subdivided into three regions: the left slope at Cat1 substrate interactions, where catalysts bind intermediates too strongly; the right slope at Cat4 and Cat5 substrate interactions, where catalysts bind intermediates too weakly; and finally an ideal binding region on the volcano plateau (or the peak) at Cat2 and Cat3 substrate interactions, where catalysts bind intermediates neither too strong nor too weak, according to the Sabatier's principle which can be regarded as ideal catalysts candidates.

Predictions of Linear Models is the Energy of the molecules, which corresponds to the x-axis in the Volcano Plot. If this value is located at the Plateau area, then this molecule can be an ideal catalyst candidate, otherwise, cat not.

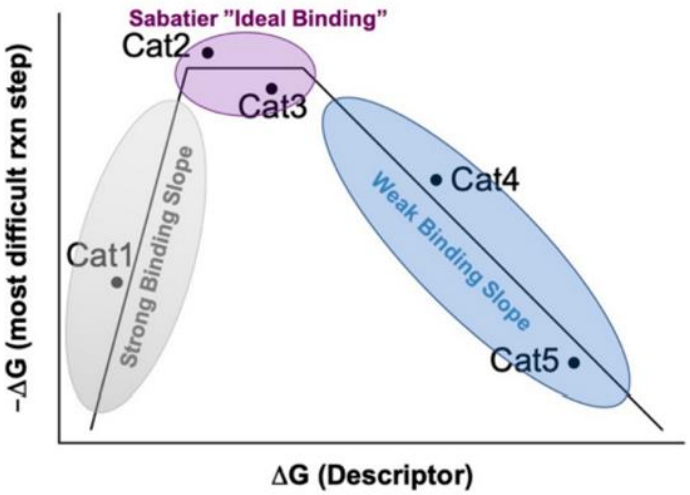


Figure 11: Volcano Plot (source from [1])[23]

4.6. Technical Requirements

Google Collaboratory[24] is used for all the implementations. The folder of dataset is uploaded in the google drive. By *from google.colab import drive*, we can get access to the folder of *"/content/drive"*. The dataset for this project is located in *"/content/drive/MyDrive/data"*. The following is a list of the python libraries that are used in this project:

qml[25]

xyz2graph[26]

[23] The descriptor value is plotted along the x axis and the negative of the free energy of the most difficult reaction step of the catalytic cycle (or another measure of catalytic activity) along the y axis.
[24] https://colab.research.google.com/?utm_source=scs-index
[25] http://www.qmlcode.org/
[26] https://github.com/zotko/xyz2graph

Graph2Vec in Karateclub[27]

StellarGraph[28]

Tensorflow

Keras

Networkx

ScikiLearn

Numpy

Pandas

Matplotlib

qml is used for the implementations of CM, BoB, SLATM.

xyz2graph is used to convert the compound objects to the graph objects and *Networkx* is used to transform graph objects to the Networkx graph type which is the commonly used type. *Graph2vec in Karateclub* is the package for the implementation of Graph2vec.

Stellargraph can transform Networkx graph into the type of Stellargraph. The Packages of *Stellargraph*, *tensorflow*, and *keras* are used for the implementation of UGRAPHEMB, which is implemented based on graph convolutional neural network.

ScikiLearn is for the linear models.

Numpy, *Pandas* and *Matplotlib* are the packages for the data process.

[27] https://karateclub.readthedocs.io/en/latest/_modules/karateclub/graph_embedding/graph2vec.html
[28] https://stellargraph.readthedocs.io/en/v0.8.3/quickstart.html

5. EXPERIMENTS AND RESULTS

Approaches for molecular representations are introduced in Chapter 3. Computational Details including the dataset, linear models, accuracy measurement, dimension reduction method, and computational tools are introduced in Chapter 4. In this chapter experiments and results are presented and discussed.

5.1. Experiments

The flows for experiments are as **Figure 12**. The rectangle boxes on the right side describes the basic steps of our experiments: featurization/representation of data, dimension reduction, and downstream tasks by linear models.

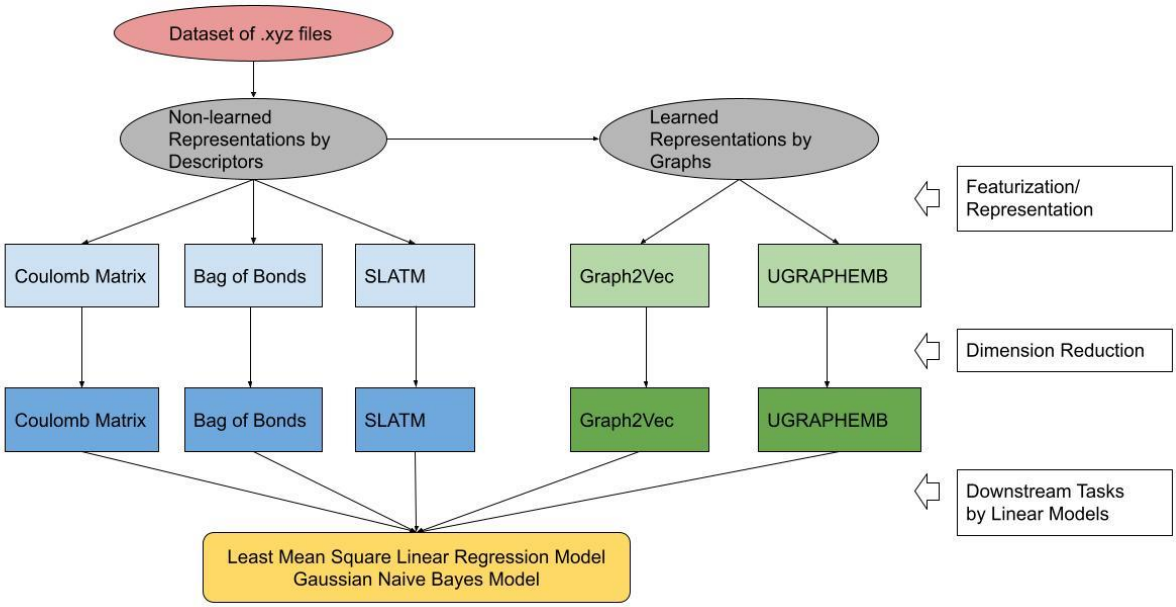


Figure 12: Experiment Flow

As for non-learned representations based on descriptors, the compound objects are constructed from the dataset of cartesian coordinate files by *qml* package. The compound objects can call the representation methods of CM, BoB and SLATM for molecular representations.

As for Learned Representations based on graphs, the compound objects are further converted into graph objects by *xyz2graph* and *Networkx*. Such graphs can be applied by graph machine learning algorithms of Graph2vec and UGRAPHEMB. **Table 2** shows the embedded dimensions of each representation method, m is the number of graphs used in the implementation, which is m = 985.

After all representations obtained, dimension reductions are applied on them for the preparation of downstream tasks. The first column of (D = 10, 30, 50, 70, 90) in **Table 3** and **Table 4** indicates the dimensions of representations by all the methods have been reduced to.

Table 2: Default Dimensions of Representation Methods

Non-learned Representations by Descriptors			Learned Representations by Graphs	
CM	BoB	SLATM	Graph2vec	UGRAPHEMB
(m, 276)	(m, 465)	(m, 10552)	(m, 128)	(m,96)

5.2. Results

The root-mean-square-error (RMSE) is a frequently used measure of the difference between values predicted by a model or an estimator and the values observed. The RMSE serves to aggregate the magnitudes of the errors in predictions and is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, which are also discussed in Section 4.3. Thus, RMSE is used as an assessment variable for comparing different representations when applying the same linear model for prediction tasks.

Table 3 shows the results of RMSE of Least Mean Square Linear Regression running on the embeddings, and **Table 4** shows the results of RMSE of Gaussian Naïve Bayes. In order to see how the dimensions of representations affect the performance, the number of dimensions is taken as a parameter for both algorithms of Linear Regression and Gaussian Naïve Bayes, with expectation to explore such effect.

Figure 13 is visualization for Linear Models with x-axis indicating the five representation methods and y-axis indicating RMSE, in which the top one is for Least Mean Square Linear Regression and the bottom one is for Gaussian Naïve Bayes. How each representation methods perform can be observed by this Figure. When one point at x-axis is selected, which indicate the representation method, how dimensions affect this representation can also be observed.

Figure 14 is visualization with x-axis indicating the dimensions and y-axis indicating RMSE, in which the top one is Least Mean Square Linear Regression and the bottom one is Gaussian Naïve Bayes. When one point at x-axis is selected, which indicate the dimension, how the five representation methods differ at this given dimension can be observed.

From **Figure 13**, we can see that when applying Least Mean Square Linear Regression, Coulomb Matrix (CM) performs best among all the five types of representations (CM, BoB, SLATM, Graph2vec, UGRAPHEMB). We can also see that Graph2vec and UGRAPHEMB, which are learned representations, perform stably regardless which linear models is used and regardless the dimension of the representations. Thus, we can say that the learned representations are more stable than the non-learned representations.

As for dimensions, by both **Figure 13** and **Figure 14**, dimensions affect the performance of BoB largely and SLATM slightly, but not the other ones. Dimensions do not affect the

performance of Graph2vec and UGRAPHEMB obviously, which can also verify that the learned representations perform in a more stable manner than non-learned representations.

Between the two algorithms of Least Mean Square Linear Regression and Gaussian Naïve Bayes that are used in this paper, Gaussian Naïve Bayes is more stable than Least Mean Square Linear Regression. Given Least Mean Square Linear Regression is applied, the choice of representation methods does matter. In the process of feature engineering, we prefer the featurizations that could perform in a stable manner for the downstream tasks regardless which conventional machine learning algorithms are selected.

Table 3: Results by Least Mean Square Linear Regression

RMSE	Non-learned Representations by Descriptors			Learned Representations by Graphs	
Linear Regression	CM	BoB	SLATM	Graph2vec	UGRAPHEMB
D = 10	63	137	68	260	252
D = 30	50	25722	48	263	230
D = 50	51	26497	183	265	230
D = 70	48	31583	185	282	225
D = 90	69	38219	33	275	230

Table 4: Results by Gaussian Naïve Bayes

RMSE	Non-learned Representations by Descriptors			Learned Representations by Graphs	
Linear Regression	CM	BoB	SLATM	Graph2vec	UGRAPHEMB
D = 10	297	387	289	314	313
D = 30	313	361	365	299	501
D = 50	317	416	418	306	279
D = 70	342	300	287	324	297
D = 90	306	277	306	315	327

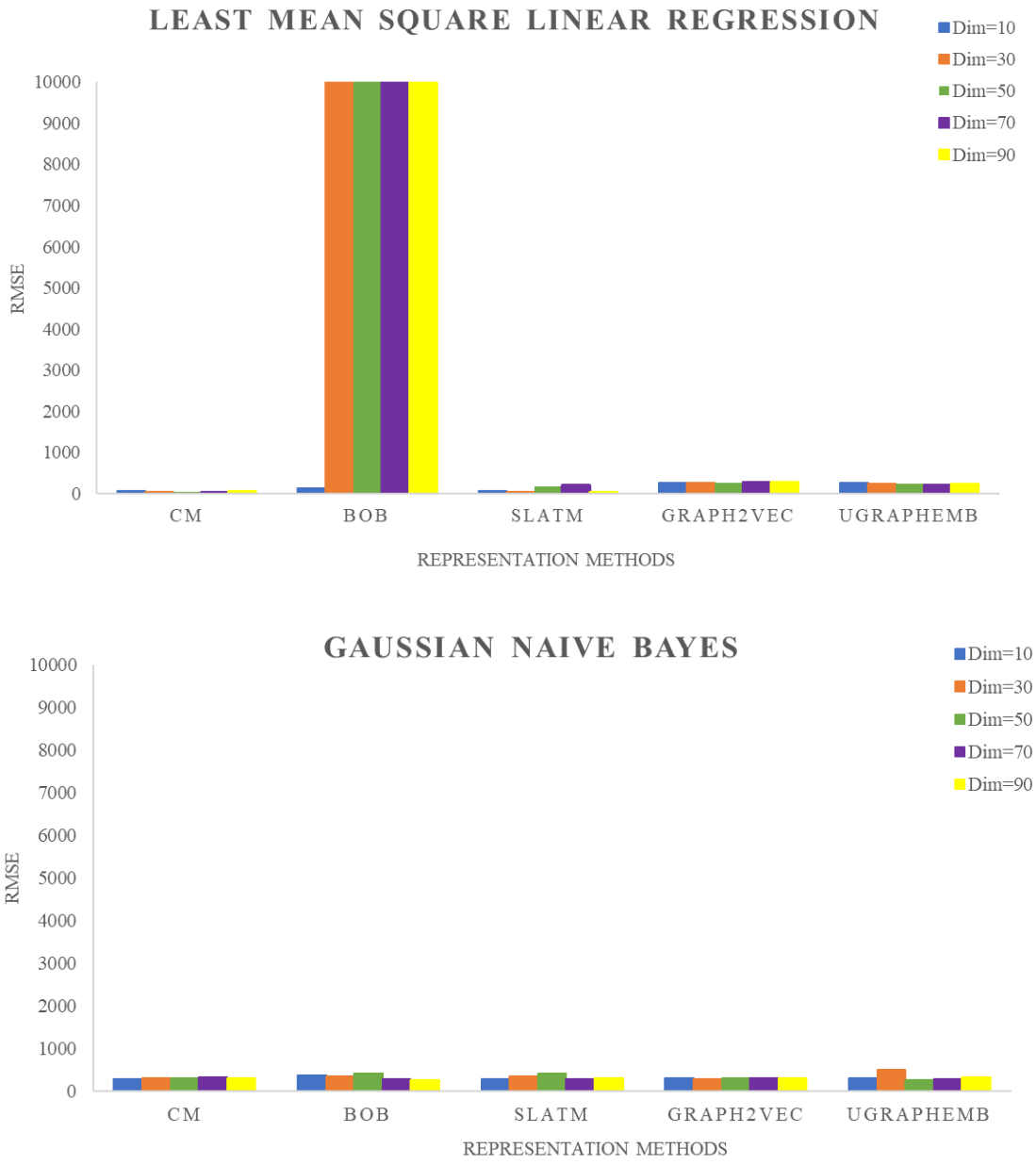


Figure 13: Representation Methods for Linear Models[29]

[29] X-axis in Figure 13 is representation methods and this figure shows how representation methods perform when one Linear Mode is chosen.

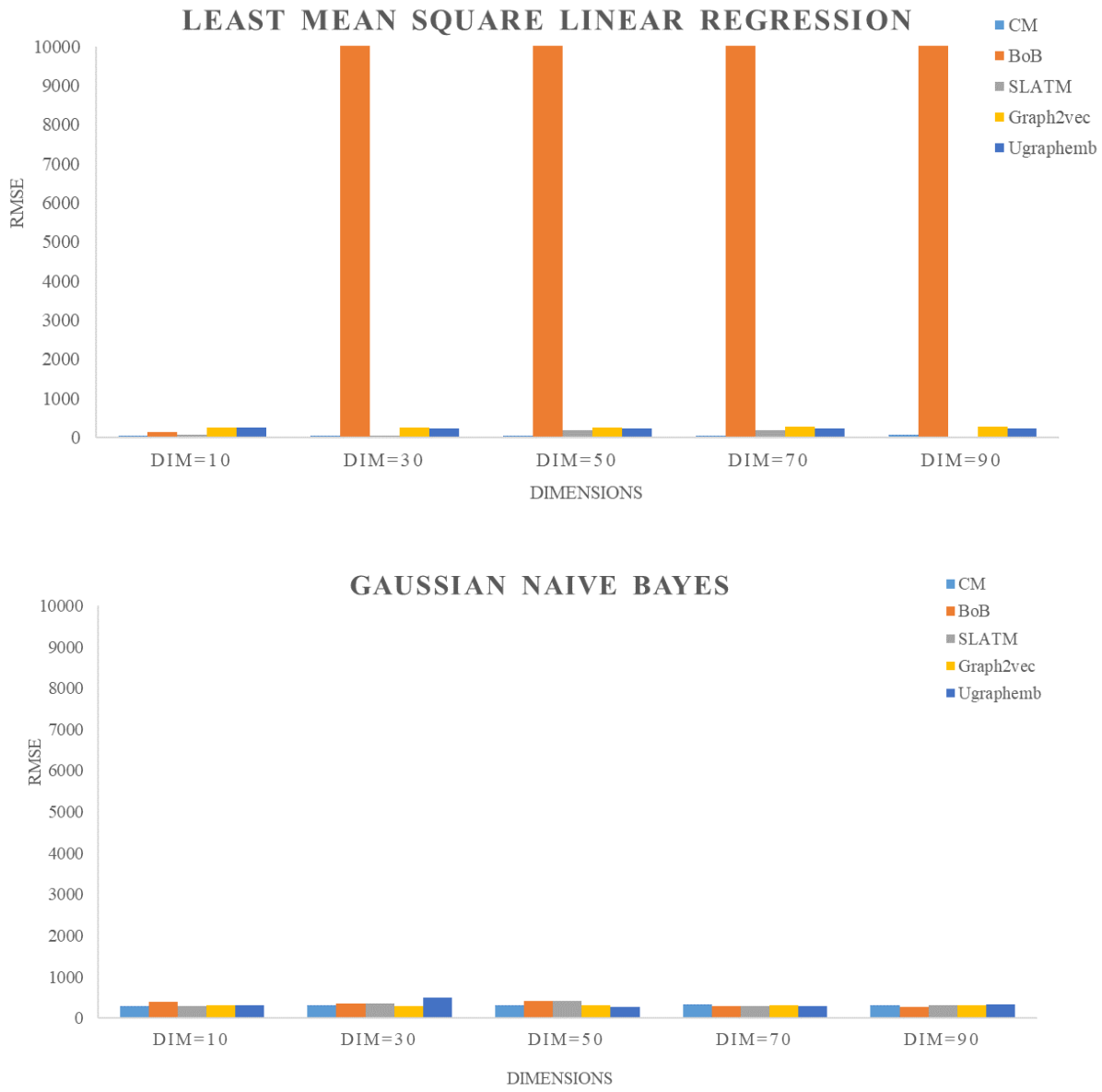


Figure 14: Dimensions for Linear Models[30]

[30] X-axis in Figure 14 is dimensions and this figure can better show how dimensions affect the performance of each representation methods when one Linear Mode is chosen.

6. CONCLUSION AND FUTURE WORK

Catalyst discovery is crucial to solve climate change and energy scarcity. Machine learning is used to predict the properties of catalyst candidates which are calculated by Density Functional Theory. To represent molecules into the format that can be used by machine learning algorithms is the initial task. There are many existing representation methods, so to compare how different representation methods perform is the main purpose of this paper. Thus, non-learned representation methods based on descriptors by CM, BoB and SLATM, and learned representation methods based on graphs by Graph2vec and UGRAPHEMB are implemented. Dimension reduction techniques are applied to the represented feature vectors to prepare them with the same dimensions before they are fed to linear models.

The results show that learned representations based on Graphs of Graph2vec and UGRAPHEMB perform in a more stable manner than non-learned representations based on descriptors, which means that there is richer and more comprehensive latent features learned automatically by such graph machine learning algorithms, which is preferred. Thus, the promising development of graph machine learning is worth the effort to be applied on molecular representations. Dimensions of the representations generally have no obvious effect on the performances of learned representations, thus we can say that learning the latent features is more important than the dimension reduction on the embeddings.

In this paper, only Least Mean Square Linear Regression and Gaussian Naïve Bayes are applied to explore how these five types of representations perform. Which linear models to choose affect non-learned representations greatly. However, linear models do not affect how the representations perform if they are learned. During the implementations, the used libraries and packages are published by other researchers, which are publicly available. Thus, there is no

customized usage, and this stopped us to further investigate the reason. However, the results still helped us to better understand the performances of the representation methods.

As for future work, by the results obtained in Chapter 5, we can conclude that the learned representations based on graphs perform in a more stable way than non-learned representations based on descriptors. Although non-learned featurizations are powerful and have held up well in the literature, they often do not capture important patterns in the data, which cause some questions for researchers: What features should be generated for the data? What patterns in the data should features capture? Should all features be used, or only a selection, and if only a selection, which ones? Do features sufficiently encode the input or is useful signal unintentionally discarded? Should expensive quantum mechanical calculations be included, or can they be avoided? How can we be certain features correlate with the output and not just add noise?

All these questions have motivated researchers to apply deep learning with the primary goal of learning the feature extraction process as part of the model, instead of relying on external preprocessing for feature generation. Compared with traditional approaches, incorporating featurization as part of the model-building process unlocks the ability to learn the featurization in a data-driven manner, granting significantly improved model flexibility at reduced engineering complexity and potentially lower total computational cost. Such data-driven featurizations can produce more informative representations, leading to improved model performance. Based on the experiments in this paper, we can also see that learned representations have more stable performances than non-learned representations, which can further prove that learned representations contain richer information based on original data.

Thus, to learn molecular representations using deep learning algorithms is an intriguing research topic.

7. REFERENCES

[1] M. D. Wodrich, B.Sawatlon, M. Busch, and C. Corminboeuf, “The Genesis of Molecular Volcano Plots”, Published as part of the Accounts of Chemical Research special issue “Data Science Meets Chemistry”, 2021.

[2] B. Meyer, B. Sawatlon, S. Heinen, O. A. Lilienfeld, and C. Corminboeuf, “Machine Learning meets volcano plots: computational discovery of cross-coupling”, Rsc.li/chemical-science. DOI: 10.1039/c8sc01949e, 2018.

[3] S. Raghunathan and U. Priyakumar, “Molecular representations for machine learning applications in chemistry”, Int J Quantum Chem, 2021.

[4] Anderson, E., G. Veith, and D. Weininger, “SMILES: A line notation and computerized interpreter for chemical structures”, U.S. Environmental Protection Agency, Washington, D.C., EPA/600/M-87/021 (NTIS PB88130034), 1987.

[5] S. R. Heller, A. McNaught, I. Pletnev, S. Stein, and D. Tchekhovskoi, "InChI, the IUPAC International Chemical Identifier". Journal of Cheminformatics. 7: 23. doi:10.1186/s13321-015-0068-4. PMC 4486400, PMID 26136848, 2015.

[6] Rogers and Hahn, “Extended-connectivity fingerprints. Journal of chemical information and modeling”, 50(5): 742-754, 2010.

[7] M. Rupp, A. Tkatchenko, K. Muller, and O. A. Lilienfeld, “Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning”, Physical Review Letters 108, 058301, 2012.

[8] K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. Lilienfeld, K. Muller, and A. Tkatchenko, “Machine Learning Predictions of Molecular Properties: Accurate Many-Body Potentials and Nonlocality in Chemical Space”, The journal of Physical Chemistry Letters, 2015.

[9] B. Huang, N. O. Symonds, O. A. Lilienfeld, “The fundamentals of quantum machine learning”, arXiv:1807.04259v2, 2018.

[10] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, “Convolutional networks on graphs for learning molecular fingerprints”, In advances in neural information processing systems, 2224-2232, 2015.

[11] H.Altae-Tran, B. Ramsundar, A. S. Pappu, and V. Pande, “Low data drug discovery with one-shot learning”, ACS central science 3(4):283-293, 2017.

[12] K. T. Schutt, F. Arbabzadah, S. Chmiela, K. R. Muller, and A. Tkatchenko, “Quantum-chemical insights from deep tensor neural networks”, Nature communications 8:13890, 2017.

[13] J. Gomes, B. Ramsundar, E. N. Feinberg, and V. S. Pande, “Atomic convolutional networks for predicting protein-ligand binding affinity”, arXiv: 1703. 10603, 2017.

[14] B. Ramsundar, S. Kearnes, P. Riley, D. Webster, D. Konerding, and V. Pande, “Massively multitask networks for drug discovery”, arXiv: 1502. 02072, 2015.

[15] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, et al., “Moleculenet: A benchmark for molecular machine learning”, arXiv: 1703. 00564, 2017.

[16] W. L. Hamilton, R. Ying, and J. Leskovec, “Representation Learning on Graphs: Methods and Applications”, IEEE, 2018.

[17] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Re, and K. Murphy, “Machine Learning on Graphs: A Model and Comprehensive Taxonomy”, arXiv: 2005. 03675v2, 2021.

[18] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, “Graph Learning: A Survey”, arXiv: 2105. 00696v1, 2021.

[19] Z. Zhang, P. Cui, and W. Zhu,“Deep Learning on Graphs: A Survey”, arXiv: 1812. 04202v1, 2018.

[20] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A Comprehensive Survey on Graph Neural Networks”, arXiv: 1901.00596v4, 2019.

[21] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications”, AI Open, 57-81, 2020.

[22] T. Mikolov. K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space”, arXiv: 1301. 3781v3, 2013.

[23] B. Perozzi, R. Al-Rfou, S. Skiena, “DeepWalk: Online Learning of Social Representations”, arXiv: 1403. 6652v2, 2014.

[24] A. Grover and J. Leskovec, “Node2vec: Scalable Feature Learning for Networks”, arXiv: 1607. 00653v1, 2016.

[25] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: Large-scale Information Network Embedding”, arXiv: 1503. 03578v1, 2015.

[26] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, “Network Representation Learning with Rich Text Information”, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015), 2015.

[27] A. Sperduti and A. Starita, “Supervised Neural Networks for the Classification of Structures”, IEEE Transactions on Neural Networks. vol. 8, no. 3, 1997.

[28] P. Frasconi, M. Gori, and A. Sperduti, “A general framework for adaptive processing of data structures”, IEEE. TNN 9, 768-786, 1998.

[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbucher, and G. Monfardini, “The graph neural network model”, IEEE TNN 20, 61-80, 2009.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, In Proceedings of the IEEE. 86, pp. 2278-2324, 1998.

[31] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, “Geometric deep learning: going beyond Euclidean data”, IEEE SPM 34, 18-42, 2017.

[32] M. M. Li, K. Huang, and M. Zitnik, “Representation Learning for Networks in Biology and Medicine: Advancements, Challenges, and Opportunities”, arXiv: 2104. 04883v1, 2021.

[33] H. Yi, Z. You, D. Huang, and C. K. Kwoh, “Graph representation learning in bioinformatics: trends, methods and applications”, Briefings in Bioinformatics, 00(00), 1-16, 2021.

[34] K. V. Chuang, L. M. Gunsalus, and M. J. Keiser, “Learning Molecular Representations for Medicinal Chemistry”, Journal of Medicinal Chemistry, 63, 8705-8722, 2020.

[35] O. Wieder, S. Kohlbacher, M. Kuenemann, A. Garon, P. Ducrot, T. Seidel, and T. Langer, “A compact review of molecular property prediction with graph neural networks”, Drug Discovery Today: Technologies, 2020.

[36] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. A. Orgun, L. Cao, F. Ricci, and P. S. Yu, “Graph Learning based Recommender Systems: A Review”, arXiv: 2105. 06339v1, 2021.

[37] A. Derrow-Pinion, J. She, D. Wong, O. Lange, T. Hester, L. Perez, M. Nunkesser, and et al., “ETA Prediction with Graph Neural Network in Google Maps”, arViv: 2108. 11482v1, 2021.

[38] A. Monken, F. Haberkorn, M. Gopinath, L. Freeman, and F. A. Batarseh, “Graph Neural Networks for Modeling Causality in International Trade”, Association for the Advancement of Artificial Intelligence, 2021.

[39] A. Tsitsulin, J. Palowitch, B. Perozzi, and E. Muller, “Graph Clustering with Graph Neural Networks”, arXiv: 2006. 16904v1, 2020.

[40] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, and S. Jaiswal, “graph2vec: Learning Distributed Representations of Graphs”, arXiv: 1707.05005v1, 2017.

[41] Y. Bai, H. Ding, Y. Qiao, A. Marinovic, K. Gu, T. Chen, Y. Sun, W. Wang, “Unsupervised Inductive Graph-Level Representation Learning via Graph-Graph Proximity”, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), 2019.

[42] “DScribe: Library of Descriptors for Machine Learning in Materials Science”, Computer Physics Communications, 2020. [Online]. Available: https://singroup.github.io/dscribe/latest/tutorials/descriptors/coulomb_matrix.html

[43] C. Stamile, A. Marzullo, E. Deusebio, “Graph Machine Learning”, Packt Publishing, ch. 2, 2021.

[44] “Machine Learning Descriptors for Molecules”, ChemIntelligence, 2021. [Online]. Available: https://chemintelligence.com/blog/machine-learning-descriptors-molecules

[45] S. A. Ludwig, “Comparison of Data Fusion Methods Applied to Epileptic Seizure Recognition”, Journal of Artificial Intelligence and Soft Computing Research, vol. 12, no. 1, pp. 5-17, 2022.