ON THE USE OF LARGE LANGUAGE MODEL FOR VIRTUAL SCREENING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ İLKER SIĞIRCI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2023

Approval of the thesis:

ON THE USE OF LARGE LANGUAGE MODEL FOR VIRTUAL SCREENING

submitted by **ALİ İLKER SIĞIRCI** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. M. Volkan Atalay
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Tunca Doğan
Computer Engineering, Hacettepe University _____

Prof. Dr. M. Volkan Atalay
Computer Engineering, METU _____

Assist. Prof. Dr. Aybar C. Acar
Graduate School Of Informatics, METU _____

Date:11.09.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: ALİ İLKER SIĞIRCI

Signature :

ABSTRACT

ON THE USE OF LARGE LANGUAGE MODEL FOR VIRTUAL SCREENING

SIĞIRCI, ALİ İLKER
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. M. Volkan Atalay

September 2023, 79 pages

Due to the abundance of drug candidates, conducting in-lab experiments to find an effective compound for a given target is a costly and time-consuming task in drug discovery. This thesis aims to reduce the number of drug candidates during early drug discovery by clustering the compounds. ChemBERTa, a Bidirectional Encoder Representation from Transformers (BERT) model, is employed to extract the descriptors for a compound. The compounds are clustered with respect to the learned features, and several clustering algorithms, including the k-means clustering algorithm and the Butina algorithm, are used. Finally, obtained clusters are evaluated by measures such as the Silhouette Score and Homogeneity Score. Our empirical findings show that using learned descriptors of ChemBERTa produces results that are comparable with traditional and graph-based models, as shown by metrics of cluster accuracy and computing runtime.

Keywords: drug-target interaction, compound descriptors, representation learning, natural language processing, clustering

ÖZ

DOĞAL DİL İŞLEME MODELİNİN SANAL TARAMADA KULLANIMI

SIĞIRCI, ALİ İLKER
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. M. Volkan Atalay

Eylül 2023 , 79 sayfa

İlaç adaylarının bolluğu nedeniyle, belirli bir hedef için etkili bir bileşiği bulmak için laboratuvar deneyleri yapmak maliyetli ve zaman alıcı bir ilaç keşfi sürecidir. Bu tez, erken ilaç keşfi sırasında bileşikleri kümeleyerek ilaç adaylarının sayısını azaltmayı amaçlamaktadır. Bir bileşiğin tanımlayıcı özelliklerini çıkarmak için Yönlü Kodlayıcı Temsili Dönüşümlerden (BERT) bir model olan ChemBERTa kullanılır. K-ortalama kümeleme algoritması ve Butina algoritması gibi çeşitli kümeleme algoritmaları, bileşiklerin öğrenilmiş özelliklerine göre kümelendirilir. Son olarak, elde edilen kümeler Siluet ve Homojenlik Skoru kriterlerine göre değerlendirilir. ChemBERTa modelinin çıktılarının kullanımının, hesaplama süresi ve kümeleme doğruluğu gibi metriklerle gösterilen geleneksel ve grafik tabanlı modellerle karşılaştırılabilir sonuçlar ürettiği, deneylerimizde gösterilmiştir.

Anahtar Kelimeler: ilaç-hedef etkileşimi, bileşik öz vektörleri, temsil öğrenimi, doğal dil işleme, kümeleme

To my family.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Prof.Dr. Volkan Atalay, for his continuous guidance, support, and invaluable insights throughout my research journey. His mentorship has been instrumental in shaping the trajectory of this thesis. I would also like to express my sincere appreciation to the esteemed members of the jury, Assoc. Prof. Dr. Tunca Doğan, and Assist. Prof. Dr. Aybar C. Acar for their time, expertise, and invaluable contributions in reviewing and improving this work.

I sincerely appreciate all of the *in-silico* group's members. In particular, Ataberk Dönmez and Alperen Dalkıran. Your collective knowledge and welcoming manner have enhanced my research experience by providing an enjoyable atmosphere for thinking.

My heartfelt appreciation goes to my family for their endless encouragement, patience, and unwavering belief in my abilities. Their love and support have been my constant motivation. I am lucky to have them in my life, namely my father Hamit Sığırcı, my mother Hatice Sığırcı, my sister İrem Gül Sığırcı, and my aunt Selime Başkurt.

To my esteemed colleagues, I am profoundly grateful for your understanding, flexibility, and support during my academic pursuits while balancing professional responsibilities. Your encouragement has been invaluable.

I would also like to extend my thanks to all the professors, peers, and friends who have contributed to my academic and personal growth during this remarkable journey.

This thesis is a testament to the collective efforts of these individuals, and their contributions truly humble me.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vi

ACKNOWLEDGMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xiii

LIST OF FIGURES xv

LIST OF ABBREVIATIONS xix

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Added Improvements 2

1.3 The Outline of the Thesis 3

2 BACKGROUND INFORMATION AND RELATED WORK 5

2.1 Compounds and Proteins 5

2.1.1 Computational Compound Representation 6

2.2 ChemBERTa 7

2.3 Dimensionality Reduction 9

2.3.1 PCA 10

2.3.2 Uniform Manifold Approximation and Projection (UMAP) . . 10
2.4 Clustering Algorithms 10
2.4.1 k-means 10
2.4.2 Hierarchical Agglomerative Clustering(HAC) 11
2.4.3 Butina 12
2.4.4 Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) 13
2.5 Compound Distance Measures 13
2.6 Clustering Evaluation Metrics 14
2.6.1 Silhouette Coefficient and Silhouette Score 15
2.6.2 Calinski-Harabasz Index 15
2.6.3 Davies-Bouldin Index 16
2.6.4 Homogeneity Score 16
2.6.5 Mutual Information Score 17
2.6.6 Adjusted Rand Index (ARI) 18
2.7 Related Work on Compound Representation and Clustering 18
3 METHOD 21
3.1 Choosing of the Learned Compound Descriptor 23
3.1.1 Calculating the Learned Compound Descriptors 23
3.2 Choosing the Dimensionality Reduction Method 25
3.3 Choosing the Clustering Metrics 26
4 EXPERIMENTS 27
4.1 DATASETS 27

4.1.1 ChEMBL 27
4.1.1.1 ChEMBL Version 27 28
4.1.1.2 ChEMBL Version 29 28
4.1.2 Directory of Useful Decoys, Enhanced (DUD-E) 29
4.1.3 ZINC 15 30
4.1.4 Johnson M. Tuberculosis 30
4.2 Ablation Studies 31
4.2.1 Finetuning the ChemBERTa model 31
4.2.2 Ablation of Dimensionality Reduction Method 33
4.2.3 Ablation of the Computing Resource 38
4.3 Experimental Setup 39
4.3.1 Selected Parameters of the Clustering Algorithms 41
4.3.2 Hyperparameters of the Clustering Algorithms 44
4.4 Clustering on Johnson et al. Dataset 44
4.5 Clustering 6 Protein Families from ChEMBL_29 47
4.6 Clustering on ChEMBL_27 50
4.7 Clustering on DUD-E 53
4.8 Clustering on ZINC 55
5 CONCLUSION AND FUTURE WORK 59
5.1 CONCLUSION AND DISCUSSION 59
5.2 FUTURE WORK 60
REFERENCES 61

APPENDICES

A.1 Figures of Clustering Results on 6 Protein Families Dataset 69

B.2 Figures of Clustering Results on Chembl_27 Dataset 71

B.2.1 Abl1 71

B.2.2 Renin 72

B.2.3 Thb 73

C.3 Figures of Clustering Results on DUD-E Dataset 75

C.3.1 Abl1 75

C.3.2 Renin 76

C.3.3 Thb 77

D.4 Figures of Clustering Results on ZINC Dataset 79

LIST OF TABLES

TABLES

Table 4.1 Summary of the 3 subset datasets of ChEMBL_27 database and their respective active and inactive compound counts. 28

Table 4.2 Summary of the protein families and their respective active and inactive compound counts. 29

Table 4.3 Summary of the protein families and their respective active and inactive compound counts. 30

Table 4.4 Performance evaluation for classification of the fine-tuned ChemBERTa model on 6 protein families dataset. 33

Table 4.5 k-means clustering results evaluated with silhouette score(↑) for choosing dimensionality reduction method on the subset of 6 protein families dataset. 35

Table 4.6 k-means clustering results evaluated with homogeneity score(↑) for choosing dimensionality reduction method on the subset of 6 protein families dataset. 35

Table 4.7 Selected Hyperparameter Ranges for each Dataset. 44

Table 4.8 k-means clustering results on the Johnson et al. Dataset. 45

Table 4.9 The results of the clustering algorithms for 3 descriptors using UMAP(16) dimensionality reduction and with the best hyperparameter on the 6 Protein Families dataset 47

Table 4.10 The results of the clustering algorithms for 3 descriptors using UMAP(16) dimensionality reduction and with the best hyperparameter on ChEMBL_27 ABL1 dataset 51

Table 4.11 The results of the clustering algorithms for 3 descriptors using UMAP(16) dimensionality reduction and with the best hyperparameter on DUD-E ABL1 dataset 54

Table 4.12 The running time comparison of the end-to-end clustering pipeline for 3 descriptors using UMAP(16) dimensionality reduction on 900,000 compound subset of ZINC15 database. Single running time refers to clustering analysis conducted using a singular value of the hyperparameter, whereas total running time refers to clustering analysis performed across a range of hyperparameter values. 57

Table 4.13 The running time percentages for each step of the end-to-end clustering for each descriptors on ZINC 15 database 57

LIST OF FIGURES

FIGURES

Figure 2.1 Different type of representations for Caffeine molecule. (A) Two-dimensional graph notation where each vertex is an atom and each edge is a bond. (B) A three-dimensional version of (A) where the molecule depth is also shown. (C) SMILES notation. (D) 2048-sized binary ECFP4 descriptor (fingerprint). 7

Figure 2.2 Visual representation of ChemBERTa pre-training approaches; MLM and MTR (adapted from [1]). 8

Figure 3.1 Visual summary of the proposed method 22

Figure 3.2 The process of calculating learned compound descriptors from ChemBERTa model 25

Figure 4.1 End-to-End ChemBERTa model fine-tuning pipeline using dagster tool on 6 protein families datasets on ChEMBL_29 database. 32

Figure 4.2 Detailed version of k-means silhouette score results of ChemBERTa model for choosing dimensionality reduction method on the subset of 6 protein families dataset 36

Figure 4.3 Detailed version of k-means silhouette score results of Chemprop model for choosing dimensionality reduction method on the subset of 6 protein families dataset 37

Figure 4.4 Detailed version of k-means silhouette score results of ECFP4 model for choosing dimensionality reduction method on the subset of 6 protein families dataset 37

Figure 4.5 The end-to-end k-means clustering running time comparison between CPU and GPU on ZINC database with ranging from 100 to 80,000 molecules 39

Figure 4.6 Silhouette scores of all models using UMAP(16) with different n_clusters on the Johnson et al. Dataset 45

Figure 4.7 Calinski-Harabasz scores of all models using UMAP(16) with different n_clusters on the Johnson et al. Dataset 46

Figure 4.8 Davies Bouldin scores of all models using UMAP(16) with different n_clusters on the Johnson et al. Dataset 46

Figure 4.9 Silhouette scores of k-means and HAC results using UMAP(16) with different n_clusters on the 6 Protein Families dataset. 49

Figure 4.10 ARI values of k-means and HAC results using UMAP(16) with different n_clusters on the 6 Protein Families dataset. 49

Figure 4.11 Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the ChEMBL_27 ABL1 Dataset 52

Figure 4.12 Homogeneity scores of k-means and HAC for 3 descriptors with different n_clusters on the ChEMBL_27 ABL1 Dataset 52

Figure 4.13 Silhouette scores of k-means and HAC for all models with different n_clusters on the DUD-E ABL1 Dataset 54

Figure 4.14 Homogeneity scores of k-means and HAC for all models with different n_clusters on the DUD-E ABL1 Dataset 55

Figure 4.15 Silhouette scores of k-means and HAC for all models with different n_clusters on the ZINC15 Database with 900,000 compounds. . . 56

Figure .1 Silhouette scores of Butina using UMAP(16) with different threshold on the 6 Protein Families Dataset 69

Figure .2 ARI values of Butina using UMAP(16) with different threshold on the 6 Protein Families Dataset 69

Figure .3 Silhouette scores of HDBSCAN using UMAP(16) with different min_cluster_size on the 6 Protein Families Dataset 70

Figure .4 ARI values of HDBSCAN using UMAP(16) with different min_cluster_size on the 6 Protein Families Dataset 70

Figure .5 Silhouette scores of Butina clustering for 3 descriptors with different threshold on the Chembl_27 ABL1 Dataset 71

Figure .6 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the Chembl_27 ABL1 Dataset 71

Figure .7 Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the Chembl_27 Renin Dataset 72

Figure .8 Silhouette scores of Butina clustering for 3 descriptors with different threshold on the Chembl_27 Renin Dataset 72

Figure .9 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the Chembl_27 Renin Dataset 73

Figure .10 Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the Chembl_27 THB Dataset 73

Figure .11 Silhouette scores of Butina clustering for 3 descriptors with different threshold on the Chembl_27 THB Dataset 74

Figure .12 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the Chembl_27 THB Dataset 74

Figure .13 Silhouette scores of Butina clustering for 3 descriptors with different threshold on the DUD-E ABL1 Dataset 75

Figure .14 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the DUD-E ABL1 Dataset 75

Figure .15 Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the DUD-E Renin Dataset 76

Figure .16 Silhouette scores of Butina clustering for 3 descriptors with different threshold on the DUD-E Renin Dataset 76

Figure .17 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the DUD-E Renin Dataset 77

Figure .18 Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the DUD-E THB Dataset 77

Figure .19 Silhouette scores of Butina clustering for 3 descriptors with different threshold on the DUD-E THB Dataset 78

Figure .20 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the DUD-E THB Dataset 78

Figure .21 Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the ZINC15 Database with 900,000 compounds. . . 79

LIST OF ABBREVIATIONS

AE	Autoencoder
AUROC	Area Under the Receiver Operating Characteristics
AUPRC	Area Under Precision-Recall Curve
BERT	Bidirectional Encoder Representations from Transformers
BPE	Byte-Pair Encoding
CPU	Center Processing Unit
D-MPNN	Directed Message Passing Neural Network
DTI	Drug-Target Interaction
ECFP	Extended Connectivity Fingerprints
GCN	Graph convolutional network
GPCR	G-Protein Coupled Receptor
GPU	Graphics Processing Unit
HAC	Hierarchical Agglomerative Clustering
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
MLM	Masked Language Modeling
MTR	Multi Task Regression
PCA	Principal Component Analysis
RAM	Random Access Memory
RoBERTa	Robustly Optimized BERT Pretraining Approach
SELFIES	Self-Referencing Embedded Strings
SMILES	Simplified molecular-input line-entry system
UMAP	Uniform Manifold Approximation and Projection
VAE	Variational Autoencoder

VS	Virtual Screening
QSAR	Quantitative Structure Activity Relationship

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Discovery and development of novel drugs have the potential to treat a variety of diseases, lengthen people's lives, and improve their general well-being. Finding an effective compound (drug) in the conventional experimental approach needs a laboratory environment. Researchers must test each compound to find the appropriate drug-target interaction pairs for the problem. Since there are too many candidate compounds, the average time it takes to get an effective drug to market is around 15 years, and the total cost is around 2 billion dollars [2, 3]. The emergence of computational methods significantly reduced the challenges associated with time and financial resources. Researchers can simulate and analyze compound-target interactions in a virtual environment using these techniques. This process is called Virtual screening(VS), a computational method for drug discovery, identifying binding structures of the small molecules to target proteins [4].

The public databases consist of chemical molecules and their target activities. Unfortunately, the target activities of many chemical molecules aren't known, which means they are many unlabeled drug-candidate compounds. Hence, the traditional virtual screening approaches need significant computing power. To reduce the high computing power requirement during the early stages of the drug discovery process, the number of candidate compounds needs to be reduced to a reasonable number for further processing. For this purpose, learned compound descriptors can be extracted from a BERT model, and the compounds can be divided into different groups based on these descriptors. Each clustered group is expected to consist of compounds with similar target activities. Based on this intuition, target protein-based virtual screening

methods can be applied in the reduced number of each cluster.
Our motivation in the thesis is to find the effects of the learned compound descriptors from the ChemBERTa model on the accuracy and speed of compound clustering.

1.2 Added Improvements

The main contribution of this thesis is to utilize the learned compound descriptors of the BERT model in compound clustering and compare the results with traditional descriptors. The subsequent enhancements are as follows.

- Unlike traditional compound descriptors like ECFP4, we use the learned compound descriptors from the ChemBERTa-2 model [1] to cluster compounds to screen them against a given target protein. The use of the latent space of BERT models for compound clustering hasn't been explored in the literature, and our study shows that using the pre-trained ChemBERTa-2 model for compound clustering gives competitive results compared to traditional descriptors and graph-based models.

- We apply dimensionality reduction techniques to reduce the running time of compound clustering.

- We compare traditional and learned compound descriptors for the prediction of drug-target interaction. This comparison shows that learned ChemBERTa model descriptors give competitive cluster accuracy compared to traditional ones. The main advantage of using learned descriptors is their clustering speed and, thus, reduced running time.

- Additionally, we demonstrate that the extraction of learned compound descriptors is more efficient compared to graph-based descriptors and equally as efficient as standard descriptor approaches in terms of computational efficiency.

1.3 The Outline of the Thesis

This thesis consists of a total of 5 chapters. Chapter 2 gives necessary information regarding compounds, proteins, Drug-Target Interaction, ChemBERTa model architecture, and clustering techniques. Chapter 3 describes the proposed method for extracting and clustering compound descriptors from the BERT model. Moreover, different approaches to get the learned compound descriptors are discussed. Chapter 4 explains the experiment setup, used datasets, and the results of the experiments. Finally, Chapter 5 summarizes the thesis and discusses possible future work.

CHAPTER 2

BACKGROUND INFORMATION AND RELATED WORK

2.1 Compounds and Proteins

A substance made up of two or more elements that are chemically linked together in a certain proportion is referred to as a chemical compound. Chemical bonds, which can be covalent or ionic, are used to hold these elements together, creating a distinct complex molecule with certain features [5]. In the drug-target interaction context, they are referred to as drugs.

Proteins are substantial, complex macromolecules essential for developing, maintaining, and controlling every living organism. They are made up of peptide bonds that connect chains of amino acid units, which are smaller building blocks. [5]. In the drug-target interaction context, they are referred to as drug targets.

Drug-target interaction describes the chemical interactions and binding between a drug molecule and its biological target, generally a protein such as an enzyme or receptor. By interacting with a target, a compound can cause activation or inhibition of the functionality of its target. Throughout the thesis, a compound and a target are said to interact if the $pChEMBL \geq 7$ (or $XC50 \leq 10\mu$M) [6]. Furthermore, even though the thesis uses the interaction term as a binary value of 0 or 1, there is also a term called affinity. It represents a continuous binding strength value [7, 8].

2.1.1 Computational Compound Representation

A key component of computational chemistry and drug discovery is compound (molecular) representation. A compound representation transforms detailed chemical structures into numerical representations that computers can understand. For a variety of computational tasks, such as determining protein binding, virtual screening, and quantitative structure-activity relationship (QSAR), it is important that compounds are accurately and effectively represented [9, 10, 11].

Figure 2.1 shows different human-readable representations of the Caffeine molecule, where 2D and 3D representations are created using RDKit [12]. Graph-based 2D and 3D representations use graph vertices as atoms and edges as atomic bonds [13]. With these attributes, a graph-based deep learning model can be trained, or a CNN model can be trained by using the visual representation of the compounds [14, 15, 16, 17]. The Simplified Molecular Input Line Entry System (SMILES) [18] and Self-Referencing Embedded Strings(SELFIES) [19] notations are in the form of one-dimensional strings and best suited for Natural Language Processing(NLP) based models like Bidirectional Encoder Representations from Transformers (BERT) [20, 1, 21, 22].

The latent space of the machine/deep learning models trained with the aforementioned notations is called learned compound descriptors [11]. In other words, the compounds are represented by the model in the form of a vector, and it is called a learned descriptor. There are also other approaches for compound descriptors that do not use any machine/deep learning model, which are called traditional descriptors/fingerprints. The most used traditional descriptor is ECFP4 [23], which represents the compound with a 2048-sized binary vector.

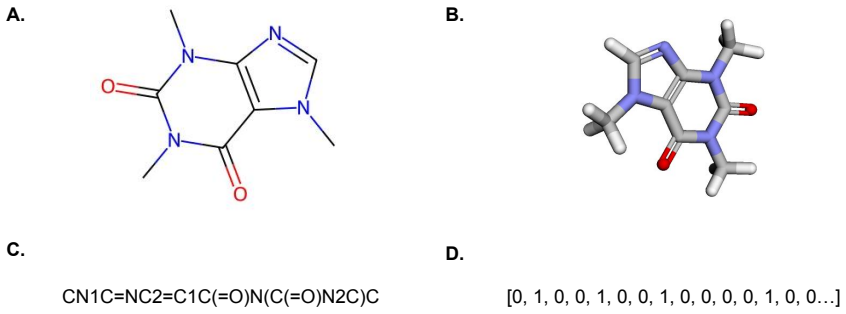

Figure 2.1: Different type of representations for Caffeine molecule. (A) Two-dimensional graph notation where each vertex is an atom and each edge is a bond. (B) A three-dimensional version of (A) where the molecule depth is also shown. (C) SMILES notation. (D) 2048-sized binary ECFP4 descriptor (fingerprint).

2.2 ChemBERTa

The ChemBERTa is a Robustly Optimized BERT Pretraining Approach (RoBERTa) [24] based Natural Language Processing (NLP) model. It is utilized for creating learned compound descriptors, which is explained in detail in Section 3.1.1.

The transformer-based language model known as Bidirectional Encoder Representations from Transformers (BERT) is intended to comprehend natural language [25]. In order to understand complicated meanings, it learns contextualized word representations by taking into account words on both sides. BERT understands sentence relationships through Next Sentence Prediction (NSP) while being trained on a large text corpus and uses Masked Language Modeling (MLM) to predict missing words. By fine-tuning BERT with task-specific layers on labeled data, the knowledge that has been pre-trained is applied to particular tasks. The multilayer structure of BERT's transformer architecture with self-attention captures linguistic abstraction and long-range dependencies. BERT has revolutionized natural language processing benchmarks by outperforming other systems in a variety of language-understanding tasks despite the computational demands of the task.

RoBERTa [24] improves BERT by focusing on the pre-training strategy. A deeper understanding of language is achieved by using larger batch sizes, more training data,

and longer training times. Additionally, RoBERTa eliminates Next Sentence Prediction (NSP) during pretraining in favor of just Masked Language Modeling (MLM). This improves its capacity to recognize relationships between words and contextual information. The performance of RoBERTa outperforms that of BERT, achieving state-of-the-art performance on a variety of natural language understanding tasks.
In the context of ChemBERTa, it is a RoBERTa model that is utilized for the chemical field. Its goal is to classify whether given compounds and their targets interact. The model is trained on unlabeled 77 million SMILES compounds from PubChem dataset. Despite being trained on a substantial dataset, the model's categorization as a large language model is insufficient due to its limited parameter size. The Masked Language Modeling (MLM) task is performed by masking some part of the SMILES input. Then, the model is trained to predict masked input. In their paper [21], Multi-Task Regression (MTR) model is also trained. Around 200 compound features are initially extracted from SMILES notation using RDKit [12]. The model is then trained to learn these features simultaneously. Although MTR approach is slower due to the feature size, its performance outperforms that of the MLM. Pre-training pipeline of the ChemBERTa model is visually presented in Figure2.2
ChemBERTa-2 model is the improved version of ChemBERTa that is pre-trained on the larger dataset. Throughout the thesis, ChemBERTa and ChemBERTa-2 are used interchangeably and hence, we always refer to the latest model, which is ChemBERTa-2.

Figure 2.2: Visual representation of ChemBERTa pre-training approaches; MLM and MTR (adapted from [1]).

2.3 Dimensionality Reduction

Dimensionality reduction strategies are employed in order to address the difficulties presented by datasets with a high number of dimensions. These techniques help in the simplification and conversion of complex datasets into representations with fewer dimensions. The process of reduction improves computational efficiency, enables 2D visualization of the data, and prevents the curse of dimensionality problems.
In terms of computational efficiency, it is observed that more dimensional data requires a greater amount of memory and space. By employing dimensionality reduction methodologies, one can efficiently reduce memory and space consumption. The curse of dimensionality is characterized by the increased sparsity of high-dimensional data [26]. The presence of sparsity in the data results in increased computational complexity during the computation of distance metrics and extended durations for execution. Furthermore, in the context of high-dimensional data, it can be observed that all data points show a tendency to have similar distances from one another. The computation of the distance matrix becomes inefficient by this factor [26]. The application of dimensionality reduction techniques allows for a reduction in the computing complexity associated with the data. Moreover, by using dimensionality reduction techniques, the accuracy and performance of clustering algorithms can be improved [27].

There are also possible disadvantages of using dimensionality reduction methods. Their results may be deceptive, as they might display cluster formations that may not exist in the underlying data or depict observations as being distant from one another in the projected space despite their proximity in the original space [28].

There exist two primary techniques for dimensionality reduction, namely Principal Component Analysis (PCA), which operates in a linear manner, and Uniform Manifold Approximation and Projection (UMAP), which operates in a non-linear manner.

2.3.1 PCA

Principal Component Analysis (PCA) is a linear dimensionality reduction method that preserves the variation by removing the redundancy in the data [29]. PCA identifies a collection of new orthogonal axes, referred to as principal components, which effectively capture the highest amount of variance present in the dataset. The initial principal component denotes the direction in which the data exhibits the greatest variability, followed by the subsequent principal components in a sequential manner. However, PCA assumes the underlying relationships in the data are linear. This implies that it might not efficiently capture complex non-linear patterns that exist within the data.

2.3.2 Uniform Manifold Approximation and Projection (UMAP)

Uniform Manifold Approximation and Projection (UMAP) is a non-linear dimensionality reduction method that preserves the local and global structure of the data [30]. This is achieved by representing data points as a weighted graph, where edges with higher weights connect related data points. UMAP then optimizes a low-dimensional representation of the data in a manner that closely approximates the graph topology observed in the original high-dimensional data. The distinguishing feature of UMAP is its capacity to effectively handle non-linear interactions, hence differentiating it from linear dimensionality reduction methods such as PCA. Furthermore, it is known for its computing efficiency compared to other non-linear algorithms such as t-SNE [31].

2.4 Clustering Algorithms

2.4.1 k-means

The main goal of k-means clustering is to divide a given dataset into a predetermined number of clusters, taking into consideration the similarity between data points [32].

k-means can be more accurately classified as a partitioning algorithm rather than a clustering algorithm. The k-means algorithm does not discover clusters but rather divides the dataset into a specified number of partitions by minimizing the distances within each partition. The process begins by randomly picking k initial cluster centroids from the dataset, where k represents the required number of clusters. Subsequently, the algorithm proceeds by assigning each individual data point to the centroid that is closest to it, utilizing a selected distance measure. Then, iteratively updates the centroids by re-calculating the arithmetic mean of the data points belonging to each cluster. This process repeats until the centroids are no longer changed. Although k-means is a commonly used and computationally efficient algorithm, it is subject to certain assumptions regarding the shape of clusters, sensitivity to initial conditions, and the requirement of a predetermined number of clusters. To address these issues, various techniques have been proposed. One example of such a method is the k-means++ initialization approach [33]. It changes the selection of the initial centroid location, resulting in faster convergence and improved clustering outputs. Moreover, it helps reduce the sensitivity of the original algorithms, hence increasing the likelihood of converging to the global optimum solution. In addition, one notable benefit of the k-means algorithm comes in its computational efficiency, making it the only feasible clustering approach for datasets of significant sizes[1]. The time complexity of the algorithm is $O(n \cdot k \cdot I \cdot d)$, where: n is the number of data points, k is the number of clusters, I is the number of iterations required for convergence, and d is the dimension of the data.

2.4.2 Hierarchical Agglomerative Clustering(HAC)

It is a hierarchical clustering method that follows a bottom-up strategy [34]. Initially, each data point is treated as an individual cluster. Afterward, the algorithm proceeds by repeatedly merging the clusters that are closest to each other until a predefined stopping criterion is satisfied. The identification of the nearest clusters is achieved through the use of a distance metric (*linkage*), single, complete, and average. The algorithm generates a hierarchical arrangement of clusters, which can be represented visually using dendrograms. Dendrograms are a tree-like structure, illustrating the

[1] https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html#k-means

process of merging clusters at each iteration. This enables data analysts to comprehend the hierarchical organization of clusters visually. Moreover, the algorithm provides *n_clusters* parameter, which cuts the created dendrogram at different heights to get the specified cluster number.

The calculation of each *linkage* criteria as follows:

- In the Single Linkage method, the distance between two clusters is defined as the minimum distance observed between any pair of individuals belonging to the two respective clusters.
- In the Complete Linkage, the measure of distance between two clusters is determined by the maximum distance observed between any two members belonging to the respective clusters.
- In the Average Linkage, the measure of distance between two clusters is computed as the average of all distances between individual members belonging to the respective clusters.

2.4.3 Butina

It is an unsupervised, non-hierarchical clustering algorithm, which is an adapted variant of the Sphere Exclusion algorithm [35]. It is also referred as Taylor clustering [36]. Although the procedures have similarities, Taylor did not employ them for the purpose of grouping. It is only developed for compound clustering and its use-cases. At first, the algorithm calculates cluster centroids. A good cluster centroid is the point that has the most neighbor points connected to it. For each element, their neighbors are calculated and sorted in descending order. A distance metric is used to determine the distances between each member in a sorted list and all the remaining elements. The *threshold* criteria determine the selection of distances within the same cluster. The process of iteratively repeating until it is applied to all remaining elements that have not yet been assigned as members of another cluster.
The distance metric is generally Tanimoto (Jaccard) for traditional ECFP4 descriptors

and Euclidean for learned compound descriptors [37]. Moreover, the *threshold* is a user-defined parameter, and its best value is selected using a grid-search mechanism.

2.4.4 Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) is a density-based clustering method, which is a hierarchical extension of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [38, 39]. Its primary objective is to detect clusters that vary in shape and size within a given dataset while also distinguishing and labeling noise spots. It calculates the mutual reachability distance among the data points. A hierarchy is formed by constructing a minimum spanning tree based on these distances. Subsequently, the hierarchy is compressed into a cluster tree. In conclusion, the process involves extracting clusters from the tree by identifying related components that satisfy certain requirements, such as a *minimum_cluster_size*. An important feature of the algorithm is its ability to handle varying sizes of clusters on non-uniform data, which enables its use on complex datasets. Additionally, due to its nature as a density-based approach, it exhibits improved performance when utilized in combination with dimensionality reduction techniques that effectively reduce the data points to a significantly denser set of dimensions[2]

2.5 Compound Distance Measures

A compound distance measure is employed to calculate the distance between two compounds in terms of compound descriptors. It offers a quantitative evaluation of the degree of difference between each pair of compounds. By employing these methods, it is possible to construct a distance matrix that can afterward be utilized in clustering algorithms.

Euclidean distance and Tanimoto coefficient are chosen for the distance calculation task of compound descriptors.

[2] https://umap-learn.readthedocs.io/en/latest/clustering.html

Tanimoto coefficient measure, which is also referred to as the Jaccard coefficient [40], evaluates the degree of overlap between two binary sets by calculating the ratio of the intersection size to the union size [35]. It is generally used with ECFP4 descriptors since they work better with binary data [37]. Euclidean measures the spatial distance between two points in a multi-dimensional Euclidean space, hence denoting their dissimilarity. Their respective formulations are shown in Equation 2.1 and Equation 2.2.

$$\text{Tanimoto}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

In Equation 2.1, A and B denote vectors that consist of elements with binary values.

$$\text{Euclidean}(x, y) = \frac{1}{1 + d(x, y)} \tag{2.2}$$

In Equation 2.2, $d(x, y)$ represents the Euclidean distance between x and y points.

2.6 Clustering Evaluation Metrics

Clustering evaluation metrics offer quantitative measurements such as cluster homogeneity, compactness, and separation to determine how well a clustering algorithm groups data points into clusters. Through the utilization of clustering metrics, one can conduct comparisons across the results of various clustering algorithms, optimize the hyperparameter values of clustering algorithms, and determine the optimized algorithm for the particular data.

Certain evaluation metrics need the presence of ground truth labels, while others can be effectively employed in datasets that lack such labels. In our study, three metrics have been selected for the evaluation of the resulting clustering for an unlabeled dataset: Silhouette Score, Calinski-Harabasz Index, and Davies-Bouldin Index. Three metrics, namely the Homogeneity Score, Mutual Information Score, and Adjusted Rand Index, are used for a dataset that has ground truth labels.

The implementations of the metrics used in this study are taken from scikit-learn [41] and RAPIDS [42] python libraries.

2.6.1 Silhouette Coefficient and Silhouette Score

The silhouette value is a metric that quantifies an object's cohesion (similarity to its own cluster) in relation to its separation from other clusters. [43]. It is within the range of -1 to 1. A silhouette value in near proximity to 1 indicates that the object exhibits a strong correlation with its respective cluster while displaying a weak correlation with neighboring clusters. This reflects a distinct separation between clusters. On the other, a number in close proximity to -1 indicates the possibility of misassignment of the item to a cluster. Conversely, values near 0 suggest the presence of overlapping clusters or data points that are in close proximity to cluster boundaries.

$$\text{Silhouette}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{2.3}$$

In Equation 2.3, $a(i)$ represents the average distance of point i to other points within the same cluster, and $b(i)$ represents the smallest average distance of point i to points in any other cluster (excluding the cluster it belongs to). The silhouette value for each data point indicates the quality of its cluster assignment.

2.6.2 Calinski-Harabasz Index

Calinski-Harabasz Index evaluates the degree of separation among clusters by examining the ratio of variation in inter-clusters to variance in intra-clusters. The objective is to identify higher values that demonstrate well-defined and visually separated clusters.

$$\text{CH} = \frac{\text{Tr(B)}}{\text{Tr(W)}} \times \frac{N - k}{k - 1} \tag{2.4}$$

Equation 2.4 involves the trace of the between-cluster scatter matrix, denoted as Tr(B), and the trace of the within-cluster scatter matrix, denoted as Tr(W). N represents the total number of data points, while k represents the number of clusters.

2.6.3 Davies-Bouldin Index

Davies-Bouldin Index evaluates both the separation and cohesion of clusters at the same time. It measures the mean similarity between each cluster and its most similar cluster, taking into account both the variation within clusters and the distances between clusters. Clusters containing lower values are a sign of better clustering, and they represent well-separated and cohesive clusters.

$$\text{DB} = \frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d(C_i, C_j)}\right) \tag{2.5}$$

In Equation 2.5, the variable k represents the number of clusters. The symbol σ_i denotes the average distance between data points inside cluster C_i and its centroid. Similarly, σ_j represents the average distance within cluster C_j. Lastly, $d(C_i, C_j)$ represents the distance between the centroids of clusters C_i and C_j.

2.6.4 Homogeneity Score

The Homogeneity Index evaluates the clusters, where each cluster is presumed to consist only of data points from a single class. In terms of the distribution of real classes within clusters, it measures the degree of cluster purity. A high homogeneity index value indicates that each cluster primarily corresponds to a different true class, reflecting a clear and consistent data partitioning. The range of the index value is from 0 to 1.

Moreover, this measure is particularly useful when one wants to cluster binary labeled data such as active and inactive molecules. Since the goal of our study is to cluster active molecules from inactive ones, employing the homogeneity index shows the degree to which the clusters have only active molecules or inactive molecules.

$$H = 1 - \frac{H(C|K)}{H(C)} \tag{2.6}$$

In Equation 2.6

- H represents the homogeneity score.
- $H(C|K)$ is the conditional entropy of the cluster assignments given the ground truth labels.
- $H(C)$ is the entropy of the ground truth labels.

2.6.5 Mutual Information Score

The Mutual Information Index quantifies the shared knowledge between the two sets of labels, demonstrating the extent to which understanding of one set aids in the prediction of the other. Greater mutual information values imply that the clustering algorithm effectively captures the underlying structure of the data.

$$\text{MutualInfo}(C,T) = \sum_{i=1}^{k}\sum_{j=1}^{l} \frac{|C_i \cap T_j|}{n} \log\left(\frac{n \cdot |C_i \cap T_j|}{|C_i| \cdot |T_j|}\right) \tag{2.7}$$

In Equation 2.7:

- C_i represents the ith cluster;
- T_j represents the jth true class;
- $|C_i \cap T_j|$ is the number of common points between C_i and T_j;
- $|C_i|$ is the number of points in cluster C_i;
- $|T_j|$ is the number of points in true class T_j;
- n is the total number of data points;
- k is the number of clusters;
- l is the number of true classes.

.

2.6.6 Adjusted Rand Index (ARI)

Adjusted Rand Index (ARI) quantifies the degree of similarity between the cluster labels generated by a clustering algorithm and the actual cluster labels. The evaluation metric takes into account the differences for both true positive and true negative instances and normalizes the resulting score to a range between -1 and 1. Higher values on this scale indicate a stronger level of clustering accuracy. Moreover, it penalizes unnecessary splits which become useful when the total number of clustering sizes is known.

$$\text{ARI} = \frac{\text{RI} - \text{Expected RI}}{\text{Max RI} - \text{Expected RI}} \tag{2.8}$$

In Equation2.8, RI represents the Rand Index, Expected RI denotes the expected value of the Rand Index assuming randomness, and Max RI signifies the upper limit of the Rand Index.

2.7 Related Work on Compound Representation and Clustering

An essential part of the drug(compound) discovery process is predicting the interactions between compounds and targets. In drug-target interaction, compound representation is a crucial step. Machine learning methods for compound representation are popularly based on Variational AutoEncoders (VAE), graphs, and techniques for Natural Language Processing (NLP). Numerous prior studies have employed compound representations and clustered compounds with the aim of identifying the appropriate drug for the target protein.

Hadipour et al. introduce a novel molecular descriptor approach by combining PCA and VAE to integrate local and global molecular attributes [44]. The effectiveness of these descriptors is evaluated on the Mycobacterium tuberculosis dataset [45]. In this study, the superiority of VAE-based descriptors over AE-based ones is demonstrated through several clustering analyses that include k-means, AE-assisted k-means, VAE-assisted k-means, and BIRCH algorithms. They state that their best scored method

uses both local and global molecular features, which are created with RDKit [12]. The created molecular features are made input for VAE model to create a latent space. This approach is claimed to reduce virtual screening times on large datasets significantly. However, the computational complexity of VAE models is very high and requires a huge amount of training time [46]. Furthermore, alternative methods, such as UMAP, are shown to give faster and more accurate results than the autoencoder one [30].

Yang et al. employ a graph-based deep learning model called Chemprop, which learns the molecule representation through a message-passing network over the neighboring information of the graph [16]. In their benchmark on MoleculeNet dataset [47], Chemprop outperforms most baseline methods for classification and regression tasks. Another study also constructs a similar graph model called PotentialNet [48], which utilizes multi-task graph convolution. PotentialNet is reported to improve the performance over traditional machine learning models. However, training graph-based models needs a huge computational power [49]. Descriptor-based machine learning models that are easy to train and computationally efficient are shown to beat the graph-based models on property prediction tasks [49]. Hence, it can be stated that the use of graph models for molecule property prediction on large chemical databases is slow and inefficient.

Donmez utilizes Chemprop model to create learned compound descriptors of 25 dimensions [50]. Based on these descriptors, the compounds are clustered and the resulting clusterings are evaluated with several clustering accuracy metrics on 3 different datasets. The performance is then compared with the traditional compound descriptor, ECFP4. The results show that using Chemprop compound descriptors gives faster running time and efficient storage space than ECFP4 descriptors. Moreover, Chemprop descriptors give better results on one dataset, while in others, ECFP4 outperforms Chemprop.

Cassar discusses the importance of a fast clustering algorithm [51]. A distributed

clustering mechanism is proposed where the clustering job is shared among more than one worker computer. Two novel algorithms are implemented, D-Butina and DLSH-Butina. The former uses the SPARK framework for task distribution while the latter uses a novel approximation technique called Locality-sensitive hashing(LSH) to find the neighboring molecule. The accuracy of the distributed clustering is evaluated and its speed is compared with the traditional BUTINA and k-means clustering algorithms. The proposed approach demonstrates a significant improvement in processing speed, ranging from 2 to 8 times while ensuring cluster accuracy.

CHAPTER 3

METHOD

The main goal is to use learned compound descriptors for clustering and choose representative compounds from each cluster to decrease the number of compounds for virtual screening to a manageable number. The compounds are expected to be clustered based on their activity property which means that the compounds in the same cluster would interact with a similar target protein or protein family. The researchers can then use this reduced number of compounds for virtual screening and also for their in-lab experiments, which effectively decreases time and money resources.
The proposed method has 3 main steps and they are summarized below. Also, the high-level step-by-step visual representation is shown in Figure 3.1

- Extract learned compound descriptors from the pre-trained BERT model, ChemBERTa.
- Apply dimensionality reduction techniques such as PCA and UMAP onto the learned compound descriptors to get efficient storage size and faster clustering speed.
- Group compounds based on their activity property using the descriptors obtained in the previous step by using clustering techniques such as k-means, HAC, and Butina.

In the following three sections, the way of selecting the best methods for each step in our approach is given. In the first section, the reasons for choosing the ChemBERTa model for learned compound descriptors are given. In the second section, the selection of the dimensionality reduction algorithm is shown. In the final section, the selection of the computing resource is explained.

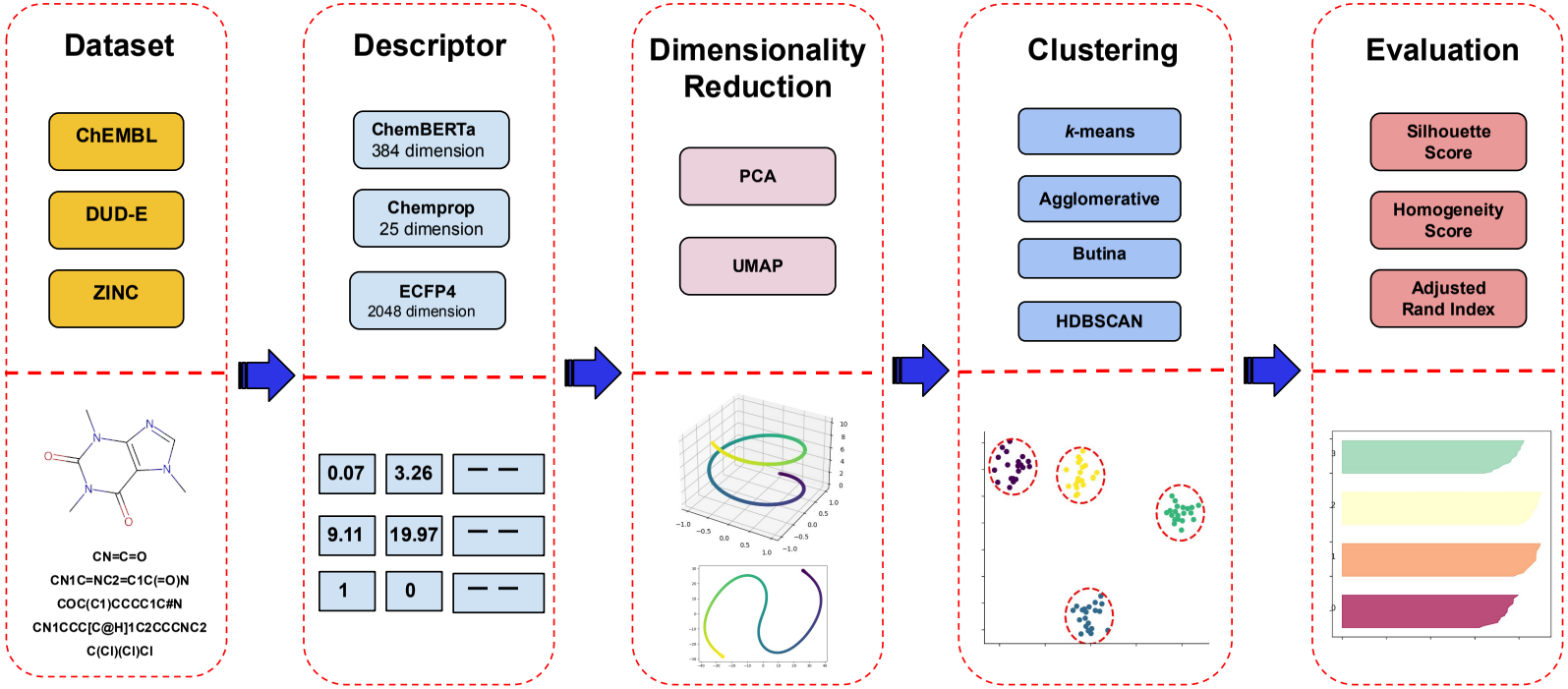

Figure 3.1: Visual summary of the proposed method

3.1 Choosing of the Learned Compound Descriptor

In Section 2.1.1, the computational compound representation methods are explained. Mainly, there are two approaches, traditional ECFP4 descriptors(fingerprints) and learned model descriptors from the deep learning models. Since the goal of the thesis is to analyze the effect of learned compound descriptors on the clustering process, a deep-learning model needed to be selected. To this end, the ChemBERTa model is chosen.

Although previous studies have used models such as Variational Autoencoder(VAE) [44] and Directed Message Passing Neural Network(D-MPNN) [50] for learned descriptors for clustering compounds, the use of BERT models for this purpose hasn't been explored to the best of our knowledge. The intuition behind selecting the ChemBERTa model is that BERT models are proven effective in the diverse fields of NLP tasks, such as question-answering, next-sentence prediction, and text clustering [52]. This means that the text representation of the model is highly accurate and usable for other text-based tasks. Given that our datasets are composed of only text-based representations of compounds, denoted as SMILES, the BERT model can be leveraged and utilized to calculate learned compound descriptors for the clustering algorithms.

BERT models can be utilized in two steps: fine-tuning the pre-trained model and calculating fixed-sized representations from the pre-trained model. Fine-tuning part is not needed for calculating the learned compound descriptors, and it is explained in detail in Ablation Studies 4.2.1.

3.1.1 Calculating the Learned Compound Descriptors

The feature-based technique utilizes the pre-trained BERT model to extract fixed vectors. This approach is alternatively referred to as contextualized word embedding [53, 52]. The process involves mapping each word to a vector space. Consequently, words with comparable meanings exhibit proximity in the aforementioned vector space. This approach offers two distinct advantages in comparison to the direct fine-tuning of the BERT model. One primary benefit is the ability to incorporate a problem-specific design, as not all natural language processing (NLP) problems can be effectively addressed using a transformer encoder architecture. Another benefit

is the enhanced computational efficiency resulting from the execution of computationally intensive pre-training representation only once. Hence, once computed, this representation can be utilized across multiple experiments. Moreover, the utilization of a feature-based technique in conjunction with a pre-trained BERT model renders it highly scalable, making it suitable for the analysis of extensive datasets.

The pre-trained BERT models contain a latent space of the vector embeddings, which we call in the context of the thesis, learned compound descriptors. Once the ChemBERTa model has been selected, we must extract learned compound descriptors from the model. The learned compound descriptors can be extracted from the model through the following process:

- Using the pre-trained Byte-Pair-Encoding tokenizer of ChemBERTa, tokenize each SMILES compound. Also, add special tokens *[CLS]*, *[SEP]* and *[PAD]*
- Initialize pre-trained ChemBERTa model with tokenized inputs.
- Get atom-level descriptors from the last hidden state of the initialized model.
- Get the 384-dimensional compound descriptors by mean averaging atom-level descriptors

The BERT tokenizer handles the process of converting the input compound such that the BERT model can utilize them. However, the dimension of the tokenized output for each compound is determined dynamically. For instance, given a SMILES notation compound, the tokenizer returns a vector with size [62, 384], but for another SMILES notation compound, it can return a vector with size [60, 384]. These vectors reflect the model at the atomic level; each vector dimension represents an atom. Since the clustering algorithms require compound-level descriptors, descriptors are converted from atom-level to compound-level by mean averaging. The process can be seen in Figure 3.2. Throughout the thesis, the final vector size, which has 384 dimensions, is referred to as the ChemBERTa model's learned compound descriptors.

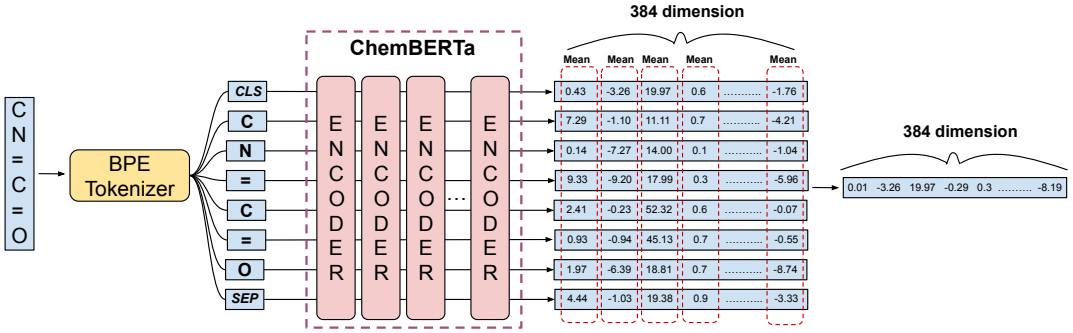

Figure 3.2: The process of calculating learned compound descriptors from ChemBERTa model

3.2 Choosing the Dimensionality Reduction Method

The compound descriptors have high dimensions, with ECFP4 having 2048 dimensions and ChemBERTa having 384 dimensions. In order to reduce the computational complexity and address the challenges caused by the curse of dimensionality, dimensionality reduction techniques need to be applied to the compound descriptors. Furthermore, the utilization of dimensionality reduction techniques has been shown to increase the effectiveness and accuracy of clustering algorithms[27]. As seen in our methodology steps given in Figure 3.1, clustering is applied after dimensionality reduction. Further elaborations on these issues can be found in Section 2.3

As for the dimensionality reduction method, UMAP method with 16 dimension is selected for following reasons:

- It can work with non-linear and complex data
- It is more accurate than PCA
- It is faster than other non-linear reduction algorithms

Detailed information about the selection process of the dimensionality reduction method can be found in ablation studies section 4.2.2

3.3 Choosing the Clustering Metrics

For unlabeled analyses of datasets, silhouette score is chosen. The motivation for this as follows:

- Visual silhouette plots are a suitable complement to this technique since they offer an enhanced understanding of the distribution of individual data points within clusters, hence allowing a full evaluation of the clustering outcomes.
- The interpretability of its value range is straightforward. A value of -1 indicates the presence of misassigned clusters, while a value of 0 signifies the existence of overlapping clusters. Conversely, a value of 1 denotes the presence of well-defined clusters.
- It is also widely used in the clustering process of the bioinformatics field [44, 54, 50].

In the context of analyzing labeled datasets, the selection of the homogeneity score is preferred above metrics such as the Adjusted Rand Index (ARI). The Homogeneity Score quantifies the extent to which each cluster exclusively consists of data points belonging to a singular ground truth class. Additionally, it does not impose penalties for additional divisions in the datasets, unlike ARI. This feature proved to be especially useful for our datasets, as they exclusively consist of binary labels, specifically active and inactive. Given the objective of our study, which is to differentiate active molecules from inactive ones effectively, it is more appropriate to utilize the homogeneity score.

CHAPTER 4

EXPERIMENTS

4.1 DATASETS

In this thesis, datasets are employed for several objectives:

- finding the hyperparameter values of the dimensionality reduction method,
- determining a model for learned compound descriptors through fine-tuning, and
- executing the experiments.

The datasets are derived from four primary sources:

- ChEMBL [6],
- Directory of Useful Decoys Enhanced (DUD-E) [55],
- ZINC Is Not Commercial 15 (ZINC 15) [56], and
- Johnson et al. M. Tuberculosis [45].

4.1.1 ChEMBL

ChEMBL database consists of over 20 million curated bio-activity measurements, over 2 million distinct compounds, and roughly 15,000 protein targets [6]. The bio-activity measurements offer information on the effectiveness and activity of various compounds against protein targets that allows the investigation of drug-target interaction.

We have used the subsets of datasets from versions 27 and 29 of the ChEMBL database in our experiments. We identify an active compound in these subsets if its pChEMBL ≥ 7 value corresponds to its target protein; this value is obtained during the filtering process [57, 50].

4.1.1.1 ChEMBL Version 27

Three subset datasets of the ChEMBL version 27 database are used in the experiments: ChEMBL1862 (ABL1), ChEMBL286 (Renin), ChEMBL1947 (THB). Generation of these datasets is based on the methodologies outlined in [57, 50]. A summary of the methodologies is as follows: datasets are produced with shared similarities to DUD-E. DUD-E is an extensive database including both active and inactive chemicals that exhibit similarity to the active molecules. The active compounds associated with a certain target are identified. Subsequently, a total of 60 inactive compounds are added for each active compound. The aforementioned process is carried out for ABL1, Renin, and THB datasets, which are the selected sub-datasets of DUD-E [51]. Each dataset is represented in a final form that has two columns, the SMILES notation, and its corresponding binary label. The label "1" denotes that the compound has interacted with the target, whereas the label "0" signifies that there is no interaction.

Table 4.1: Summary of the 3 subset datasets of ChEMBL_27 database and their respective active and inactive compound counts.

Dataset	**Active Compounds**	**Inactive Compounds**	**Total Compounds**
ABL1	1632	103,946	105,578
Renin	2667	167,330	169,997
THB	418	25,612	26,030

4.1.1.2 ChEMBL Version 29

The filtered 6 subset datasets of the ChEMBL version 29 database are used in the experiments, GPCR, Ion Channel, Kinase, Nuclear Receptor, Protease, Transporter.

The same filtering steps from [58, 59] are applied. The main goal of the filtering is to get mutually exclusive active compounds based on each target. The process basically eliminates compounds that have no target, no taxonomy, no pChEMBL value, and more than one interacted target. The final filtered dataset numbers are in Table 4.3

The utilization of these datasets can be seen in two primary experiments within our work, namely the finetuning of the ChemBERTa model 4.2.1 and the clustering experiment discussed in this chapter. For the model finetuning part, both active and inactive compounds are used for each dataset.
The experiments conducted in this chapter only utilize active molecules, which are combined from each protein family. Throughout the thesis, this data set is commonly referred to as the 6 protein families dataset. The rationale behind employing this approach comes from the fact that the six subsets of the dataset were initially generated for the purpose of binary classification, as stated in the study by Dalkiran [58]. However, in order to utilize them for the purpose of clustering, it is necessary to alter their shape. Therefore, with the utilization of only active compounds taken from each dataset, a total of six distinct clusters are obtained.

Table 4.2: Summary of the protein families and their respective active and inactive compound counts.

Protein family	**Active Compounds**	**Inactive Compounds**	**Total Compounds**
GPCR	36,924	31,085	68,009
Ion Channel	5,996	14,167	20,163
Kinase	35,531	30,778	66,309
Nuclear Receptor	5,099	6,668	11,767
Protease	15,718	19,518	35,236
Transporter	3,666	5,898	9,564

4.1.2 Directory of Useful Decoys, Enhanced (DUD-E)

Directory of Useful Decoys, Enhanced (DUD-E) database has a total of 102 target proteins together with their corresponding active and inactive compounds. The term

"decoy" refers to inactive chemicals that share similar chemical characteristics to active compounds. This particular characteristic introduces a higher level of complexity to the clustering operation [55]. Among the 102 target proteins, three specific protein datasets, namely ABL1, Renin, and THB, are selected for further analysis as described in the study by Cassar [51]. These three datasets, similar to the subset of ChEMBL datasets, consist only of two columns: SMILES notation and binary label that indicate the activity of the compounds.

Table 4.3: Summary of the protein families and their respective active and inactive compound counts.

Dataset	Active Compounds	Inactive Compounds	Total Compounds
ABL1	152	10,750	10,902
THB	103	7,450	7,553
Renin	104	6,958	7,062

4.1.3 ZINC 15

ZINC 15 database is one of the biggest and most comprehensive chemical databases accessible for computational drug discovery. It has a collection of more than 800 million different SMILES strings [56]. 2021 iteration of the ZINC15 database is used similarly to the previous work [50]. Then, Flagments subset of the database is selected. This subset comprises 900,000 compounds characterized by a molecular weight ranging from 250 to 350 daltons and a *logP* value ranging from -1 to 3.5. This final dataset is utilized to demonstrate the scalability and effectiveness of our method.

4.1.4 Johnson M. Tuberculosis

Johnson M. Tuberculosis dataset contains 47,217 unlabeled compounds represented in SMILES [45]. It is created by conducting a comprehensive chemical-genetic screen targeting the bacteria Mycobacterium tuberculosis. This is achieved by generating chemical-genetic interaction profiles (CGIPs) through the utilization of mutant strains of M. tuberculosis. This dataset is used to compare the effectiveness of our method.

4.2 Ablation Studies

In order to determine the optimal approach for each step in our methodology, a series of experiments are conducted. These experiments included the fine-tuning of the ChemBERTa model, investigations into the selection of an appropriate dimensionality reduction method, and the identification of the most suitable computing device.

4.2.1 Finetuning the ChemBERTa model

In the fine-tuning process, the best model indicated by ChemBERTa-2 paper is used, which is the Multi-Task Regression (MTR) version of the model trained on 77 million unlabeled SMILES compounds [1]. There is also the Masked Language Modeling(MLM) version of the model, but the authors indicate that it is only useful for their development, architecture selection setup, and its results are worse than the MTR version. Hence, the pre-trained *ChemBERTa-77M-MTR* model is downloaded from huggingface [1] and fine-tuned with our dataset for the binary classification task. For the fine-tuning task, we use the 6 protein families dataset on ChEMBL_29, which is explained in detail on 4.1.1. Each protein dataset is composed of a list of compounds and their respective binary labels, which indicate whether the compounds interact with the target protein. For the whole fine-tuning process, dagster[2], a data pipeline tool, is used, and its resulting pipeline is shown in Figure 4.1. The pipeline is composed of 3 main parts: data splitting, fine-tuning the model, and evaluating the fine-tuned model.

[1] https://huggingface.co/DeepChem/ChemBERTa-77M-MTR
[2] https://dagster.io/

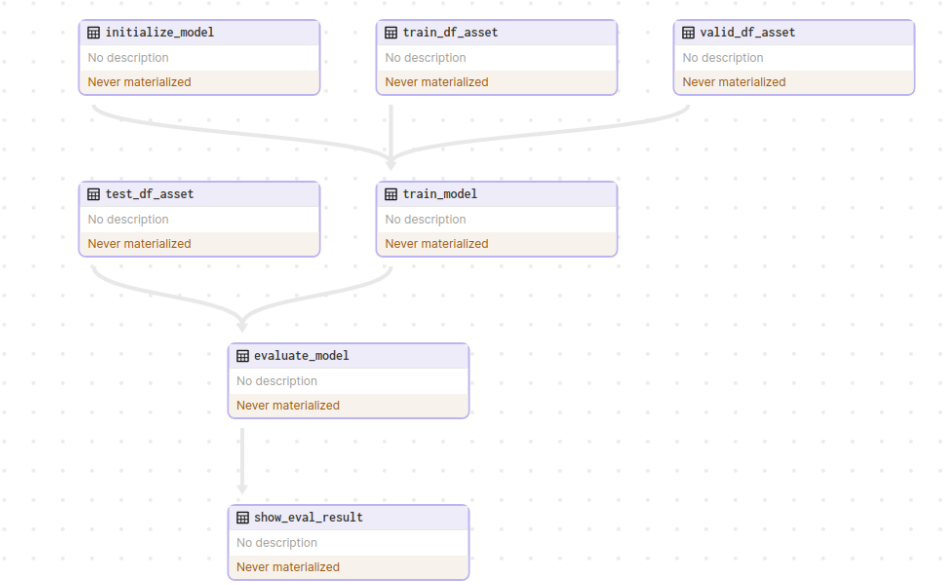

Figure 4.1: End-to-End ChemBERTa model fine-tuning pipeline using dagster tool on 6 protein families datasets on ChEMBL_29 database.

The step-by-step version is as follows:

- Data is split randomly into 3 main parts; training(80%), validation(10%) and test(10%).
- Byte-Pair Encoding (BPE) tokenizer is used to convert SMILES to model inputs.
- The model is fine-tuned on the training set for 10 epochs.
- Model evaluated with Accuracy, Area Under the Receiver Operating Characteristics(AUROC), and Area Under Precision-Recall Curve(AUPRC) metrics on the test dataset.

In order to prove the learned descriptors of the ChemBERTa model are useful, we evaluate the classification accuracy of the model. The fine-tuning evaluation results for each 6 protein families are presented in Table 4.4. The table shows the ChemBERTa model classifies active and inactive compounds with more than 0.75 AUROC score for each protein family. Moreover, in the Result section of the original

ChemBERTa-2 paper [1], the authors show the MoleculeNet benchmark scores of the ChemBERTa-2 model with other state-of-the-art models, and they claim that their model outperforms Chemprop model (D-MPNN) on 6 out of 8 fine-tuning tasks. By combining our experiment results and the benchmark scores of the original paper, we conclude that the ChemBERTa-2 model is able to learn from SMILES text representations of the compounds and give competitive results.

Table 4.4: Performance evaluation for classification of the fine-tuned ChemBERT-a model on 6 protein families dataset.

Protein Type	**AUROC** ↑	**AUPRC** ↑	**ACC** ↑	**Loss** ↓
GPRC	0.77	0.75	0.68	0.47
Ion channel	0.78	0.79	0.71	0.35
Kinase	0.80	0.81	0.72	0.36
Nuclear receptor	0.81	0.76	0.74	0.40
Protease	0.79	0.74	0.73	0.41
Transporter	0.79	0.73	0.72	0.46

4.2.2 Ablation of Dimensionality Reduction Method

PCA and UMAP are selected as candidate dimensionality reduction methods for ablation study. PCA is a linear and fast algorithm which widely used in the Drug-Target Interaction processes [60, 61]. The UMAP algorithm has been shown to be efficient in processing non-linear data [30]) and has recently been applied in the field of bioinformatics [62]. Although t-SNE can be used for non-linear data [57, 63], its computational complexity is high and requires longer running times than UMAP [28]. Therefore, t-SNE has not been chosen as a dimensionality reduction technique in our study.

The dimensions of 16 and 32 are used for dimensionality reduction, as suggested in

the UMAP documentation [3]. The reasoning behind this is that these dimensions are considered to maintain a balance between computing efficiency and the explainability of the underlying data. It is also noteworthy that the Chemprop descriptors, which have 25 dimensions, are only reduced to 16 dimensions. In addition, the GPU implementations of PCA[4] and UMAP[5] [64] from NVIDIA RAPIDS framework [42] are used for the experiments.

The k-means technique is chosen for clustering in the experimental setup. Given the fixed number of clusters, which is 6, the k-means algorithm can be utilized effectively. Moreover, it is used in similar tasks in the bioinformatics field [44].

To show the results of different dimensionality reduction methods, a series of experiments are conducted on a subset of the CheEMBL_29 database (see Section 4.1.1.2). This subset consisted of 6 protein families and contained a total of 6,000 SMILES molecules, with 1,000 molecules attributed to each protein family. The studies were conducted for all three descriptors, namely ChemBERTa, Chemprop, and ECFP4. The application of PCA and UMAP techniques effectively reduces the dimensionality of the descriptors to 16 and 32 dimensions, respectively. Following that, the reduced dimensional descriptors are subjected to clustering using the k-Means, and their performance is evaluated using the Silhouette Score.
The experiments are conducted using 10 different *random_seed* values, their mean and standard deviation results are presented in Table 4.5. The analysis reveals that the utilization of UMAP with 16 reduced dimensions yields the highest silhouette score. While UMAP with 32 dimensions shows comparable scores to UMAP with 16 dimensions, it requires larger storage space and increased computing complexity. Therefore, UMAP with a dimensionality of 16 is chosen for utilization in the following experiments in Chapter 4.

As previously mentioned in Section 2.3, dimensionality reduction techniques have a tendency to show data points at relatively close distances to one another. This might

[3] https://umap-learn.readthedocs.io/en/latest/clustering.html
[4] https://docs.rapids.ai/api/cuml/stable/api/#principal-component-analysis
[5] https://docs.rapids.ai/api/cuml/stable/api/#umap

Table 4.5: k-means clustering results evaluated with silhouette score(↑) for choosing dimensionality reduction method on the subset of 6 protein families dataset.

Method	**Dimension**	**ChemBERTa**	**Chemprop**	**ECFP4**
No Reduction	-	0.14 ± 0.03	0.13 ± 0.05	-0.15 ± 0.10
PCA	16	0.19 ± 0.05	0.16 ± 0.01	0.04 ± 0.15
PCA	32	0.16 ± 0.03	-	0.05 ± 0.10
UMAP	16	$\mathbf{0.36 \pm 0.07}$	$\mathbf{0.32 \pm 0.04}$	$\mathbf{0.28 \pm 0.08}$
UMAP	32	0.33 ± 0.04	-	0.31 ± 0.10

be the reason for the observed increase in silhouette scores. Therefore, given that our dataset is labeled, we additionally assess the homogeneity score of the experiments to ensure the validity of the dimensionality reduction methods, which results are shown in Table 4.6. Like Silhouette experiments, 10 different *random_seed* values, their means and standard deviations are calculated. The results indicate that the utilization of UMAP has led to an improvement in the scores for all three descriptors. The use of Principal Component Analysis (PCA) does not yield significant changes in our data due to its linear nature, which is not well-suited for handling the complexity of our dataset.

Table 4.6: k-means clustering results evaluated with homogeneity score(↑) for choosing dimensionality reduction method on the subset of 6 protein families dataset.

Method	**Dimension**	**ChemBERTa**	**Chemprop**	**ECFP4**
No Reduction	-	0.39 ± 0.20	0.35 ± 0.15	0.41 ± 0.10
PCA	16	0.42 ± 0.15	0.48 ± 0.07	0.44 ± 0.10
PCA	32	0.40 ± 0.10	-	0.42 ± 0.08
UMAP	16	0.58 ± 0.04	$\mathbf{0.55 \pm 0.02}$	0.51 ± 0.08
UMAP	32	$\mathbf{0.60 \pm 0.09}$	-	$\mathbf{0.52 \pm 0.06}$

The findings related to the hyperparameter *n_clusters* are presented in Figure 4.2,

Figure 4.3, and Figure 4.4. It is evident that both PCA with 16 dimensions and 32 dimensions yield comparable outcomes to the non-reduced version. This implies that the dataset at present exhibits a non-linear structure, making the application of PCA, a linear dimensionality reduction technique, ineffective in understanding the underlying patterns within the data. Furthermore, it is seen that lower scores are obtained when no reduction is carried out. This observation suggests that in the setting of higher dimensional data, the experiments suffer from the curse of dimensionality.

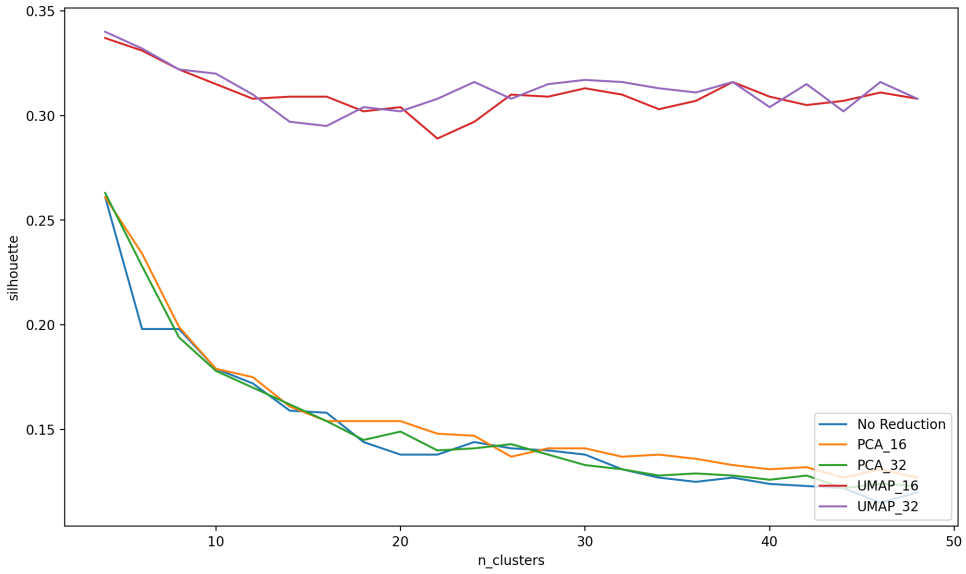

Figure 4.2: Detailed version of k-means silhouette score results of ChemBERTa model for choosing dimensionality reduction method on the subset of 6 protein families dataset

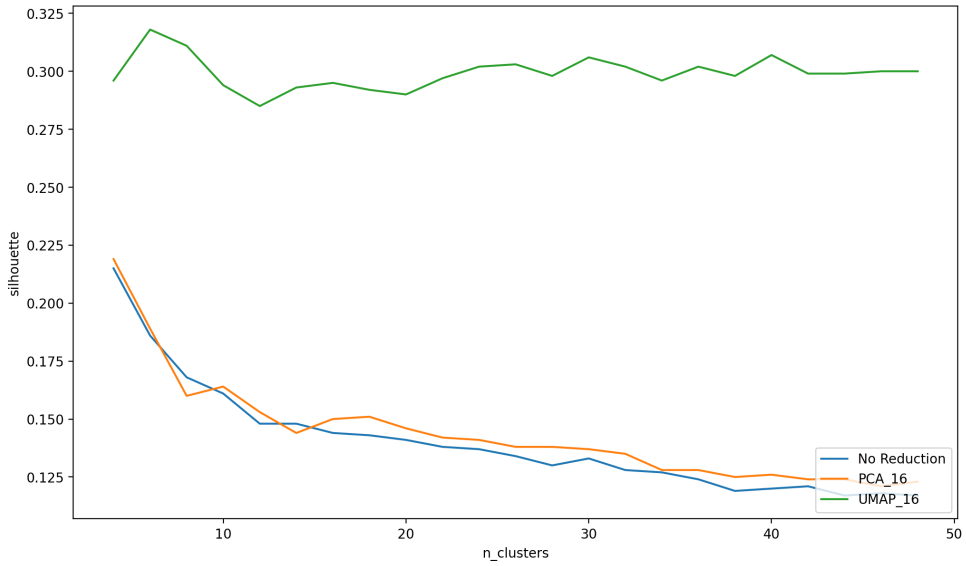

Figure 4.3: Detailed version of k-means silhouette score results of Chemprop model for choosing dimensionality reduction method on the subset of 6 protein families dataset

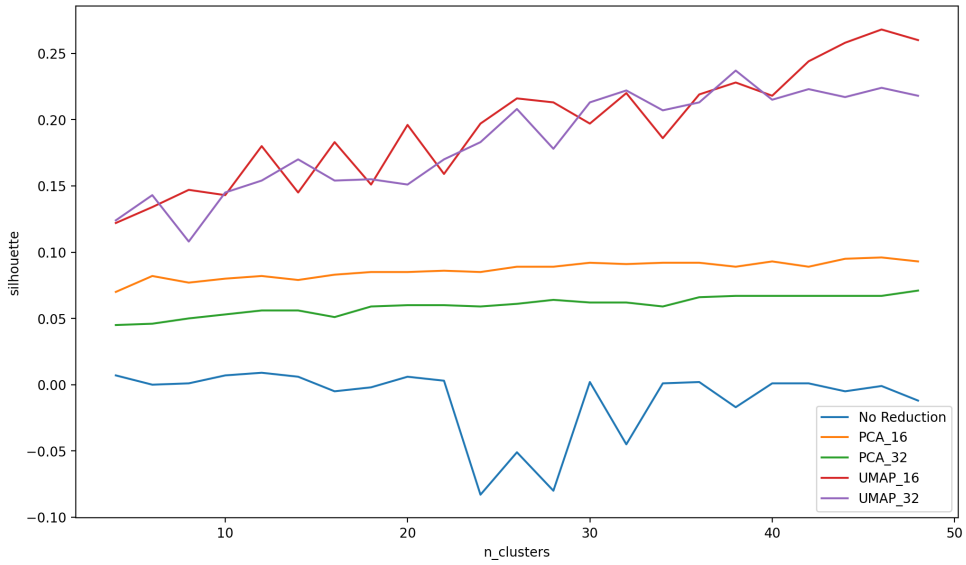

Figure 4.4: Detailed version of k-means silhouette score results of ECFP4 model for choosing dimensionality reduction method on the subset of 6 protein families dataset

4.2.3 Ablation of the Computing Resource

Since the faster running times are one of the crucial aspects for clustering large molecules, we put extra work into this subject. To this end, we utilize GPU instead of CPU since it is widely used in machine learning and deep learning pipelines for speeding up execution times [65]. We implement the experiment codes with maximum GPU utilization using the pytorch library [66]. Also, we integrate GPU-compatible clustering algorithms and evaluation methods by using the NVIDIA RAPIDS framework [42]. In the RAPIDS framework, mainly the cuML [6] library is used. It can be seen as the GPU-accelerated version of sklearn [41] for clustering algorithms. Using all aforementioned methods together in an end-to-end clustering pipeline, we get from 8x to 20x speed up. To show the difference in running time of CPU and GPU, we run an example k-means clustering experiment on a subset of the ZINC database. We increase the molecule number from 100 to 80,000 and observe the running time. Figure 4.5 shows the results of this experiment in terms of CPU and GPU running times. We see that as the molecule number increases, CPU running times follow a linear trend with respect to it. However, GPU running times show a logarithmic increase instead of a linear one.

Moreover, as the clustering molecule number increases, we need more RAM to be able to finish experiments without facing any out-of-memory errors. Since GPU VRAMs are generally less than the computer's main RAM, we implement an automatic fallback mechanism. If more than 1,000,000 compounds are clustered(the number depends on the GPU used for the experiments), we switch to CPU for all the calculations. Since none of our datasets exceeds this limit, we only use GPU for all experiments without encountering any problems. However, this switching mechanism is necessary to be able to cluster large compounds even though it increases the running time.

[6] https://docs.rapids.ai/api/cuml/

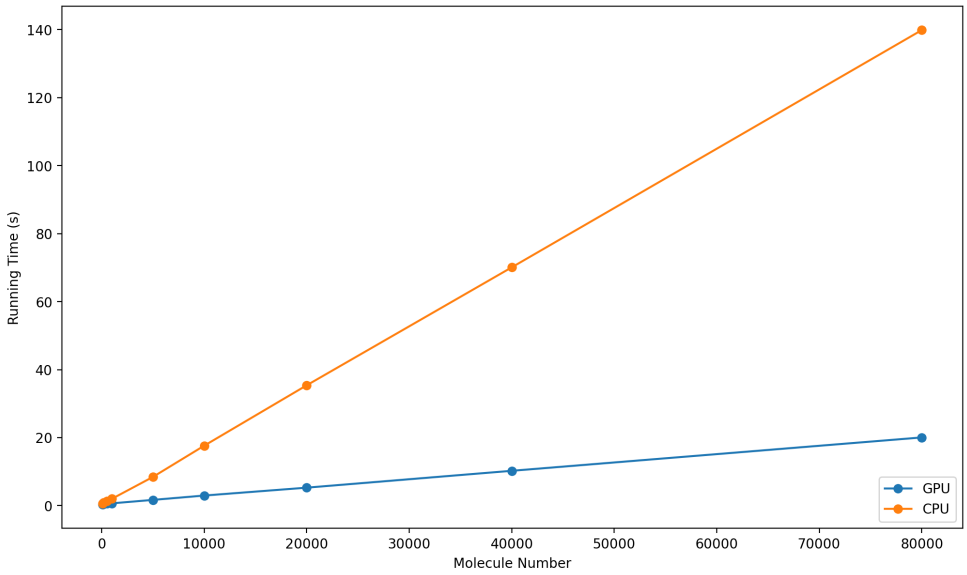

Figure 4.5: The end-to-end k-means clustering running time comparison between CPU and GPU on ZINC database with ranging from 100 to 80,000 molecules

4.3 Experimental Setup

All experiments run on a desktop computer with an AMD 5800x CPU, 32 GB RAM, and NVIDIA RTX 3070 8 GB Graphics Card.

In order to simplify the monitoring of the experiment, we utilize dagster [7], a data pipeline tool, in conjunction with Weight-and-Biases [8], a software designed for experiment tracking. In addition, all experiments are conducted with a fixed *random_seed* value, which ensures consistent generation of random elements for each run of the experiment. By integrating all of these elements, we maximize the traceability and replicability of our experiments. In addition, the rdkit [12] library is used for molecule-related operations such as rendering, substructure searching and similarity analysis. The library is a cheminformatics software toolkit that is extensively utilized in the field of chemical informatics. It is an open-source platform that offers a wide range of functionalities for both Python and C++ programming languages. Furthermore, all

[7] https://github.com/dagster-io/dagster
[8] https://wandb.ai/site

of the experiment codes can be found on github[9]

For each dataset, experiments are run using the combinations of different clustering methods, descriptors, and clustering evaluation methods.

- Clustering Methods: k-means, HAC, Butina, HDBSCAN
- Descriptors: ChemBERTa, Chemprop, ECFP4
- Dimensionality Reduction Method: UMAP with 16 dimensions
- Evaluation Methods: Silhouette Score, Homogeneity Score

The selection of dimensionality reduction method and clustering evaluation methods are explained in the previous chapter, 3.3, 3.2.

Moreover, grid-search is applied to different hyperparameter configurations for each clustering algorithm to find the best hyperparameter. The hyperparameters can be seen in the below. The effect of *n_clusters* hyperparameter is shown in the Figures of this chapter, while the details of other hyperparameters are shown in the Appendices.

- k-means: *n_clusters*
- HAC: *n_clusters*
- Butina: *threshold*
- HDBSCAN: *min_cluster_size*

In the hyperparameter figures, there might be empty values which means that for that hyperparameter value the evaluation score calculation can't be applied. This is generally due to two main reasons, either the memory limit exceeded or the used clustering algorithm doesn't cluster any points.

Additionally, we downloaded the Chemprop model that was trained using the Multi-task learning method on a subset dataset of the ChEMBL version 27 database [50] in

[9] https://github.com/ilkersigirci/thesis-work

order to obtain the Chemprop learned descriptors.

4.3.1 Selected Parameters of the Clustering Algorithms

For the k-means algorithm, cuML implementation [10] is used. As mentioned in cuML documentation, it is 10x-50x faster than sklearn CPU implementation for large datasets. Detailed information about each argument can be found in cuML documentation. Experiments are run with the following parameters:

- *init_method*: k-means++
- *n_init*: 1
- *max_sample_per_batch*: 32,768
- *max_iter*: 300
- *n_clusters*: dynamic with respect to dataset (hyperparameter)

The most important parameter of k-means is *n_clusters* since the final cluster size of the dataset is determined by it. Given the lack of prior knowledge of the exact number of clusters in the datasets, the grid-search approach is used to determine the optimal value for the *n_clusters* parameter for each dataset. The range of values for grid search varies for each dataset, and it is explained in the respective subsections dedicated to each dataset. Each range value is evaluated with different metrics like silhouette score. By looking at the metrics, it is observed that the scores tend to converge as the *n_clusters* increase to large values. The parameter *init_method* is selected as the k-means++ algorithm, which has several benefits as discussed in Section 2.4.1. The parameter *max_sample_per_batch* holds significance in our analysis due to the utilization of the large datasets. It determines the matrix dimensions for the pairwise distance calculation. The equivalent parameter, *batch_size*, is the counterpart parameter of sklearn's implementation of Mini-Batch k-means[11]. According to the cuML

[10] https://docs.rapids.ai/api/cuml/stable/api/#k-means-clustering
[11] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html

documentation, it is recommended to set a lower number for *max_sample_per_batch* when dealing with big *n_clusters* values. Nevertheless, during the course of our testing, we do not experience any issues. Therefore, the parameter is set to the default value.

In the Hierarchical Agglomerative Clustering (HAC), the implementation by Nolet et al. [67] is used. A similar approach with k-means *n_clusters* parameter is followed and its best value is determined by grid-search for each dataset. The parameters used for the experiments are as follows.

- *affinity*: Euclidean
- *linkage*: single
- *connectivity*: knn
- *n_clusters*: dynamic with respect to dataset (hyperparameter)

Detailed information about all parameters can be found in cuML documentation [12]. The parameter *connectivity* is for computing the pairwise distance between each point. k-nearest-neighbors (KNN) *connectivity* is chosen because, as the documentation states, it has superior speed compared to alternative methods. However, due to the limitation of cuML, Euclidean *affinity(linkage)* is used, which is the only supported *affinity* with KNN *connectivity*.

For Butina clustering, RDKit implementation is used[13]. Butina Clustering algorithm shows significantly slower performance compared to other algorithms that are tested since it doesn't utilize GPU and only works on CPU. Nonetheless, since it is widely used in the bioinformatics field for clustering compounds, the experiments are conducted with the following parameters:

- *distFunc*: Euclidean or Tanimoto (Jaccard)

[12] https://docs.rapids.ai/api/cuml/stable/api/#agglomerative-clustering
[13] https://www.rdkit.org/docs/source/rdkit.ML.Cluster.Butina.html

- *isDistData*: True
- *threshold*: (hyperparameter)

The parameter *threshold* defines the criteria for determining which points classify as neighbors. The optimal value is determined by conducting a grid search on each dataset, selecting from a range of values including 0.2, 0.35, 0.5, and 0.8. The *distFunc* parameter is responsible for calculating the distance between two points, and its value is dynamically determined based on the descriptor being used. For ECFP4 descriptors, the Tanimoto (Jaccard) function is used since it works better for binary values [37]. For learned model descriptors, Euclidean is used. *isDistData* parameter represents whether the data is in distance matrix form or not. In the experiments, the distance matrix is calculated from the descriptors before being given to the Butina algorithm; hence, its value is set to True.
In the field of bioinformatics, the often used method for computing the distance matrix is the *BulkTanimotoSimilarity*[14] [68, 69]. However, it is observed to be slow in our experiment runs. Therefore, implementation of *pdist* from the scipy library is utilized[15]. This approach results in improved computational efficiency and reduced memory requirements.

In HDBSCAN clustering, the cuML implementation is used[16]. In cuML documentation, it is stated that it runs 300 times faster than the original hdbscan library implementation[17]. The parameters used are the following:

- *metric*: euclidean or tanimoto(jaccard)
- *connectivity*: knn
- *min_cluster_size*: (hyperparameter)

The *min_cluster_size* determines the minimum threshold for the number of points required to be classified as clusters. The optimal value is selected by grid-search on

[14] https://www.rdkit.org/docs/source/rdkit.DataStructs.cDataStructs.html
[15] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.pdist.html
[16] https://docs.rapids.ai/api/cuml/stable/api/#hdbscan
[17] https://hdbscan.readthedocs.io/en/latest/index.html

each dataset, as detailed in their respective subsections. *metric* and *connectivity* are selected with the same strategy in the HAC and Butina clustering algorithms.

4.3.2 Hyperparameters of the Clustering Algorithms

The hyperparameter ranges are changed with respect to each dataset. Table 4.7 displays the actual values for each range in detail. The range notation of the table is the same one as Python. For instance, in [x, y, z], x is the beginning number, y is the ending number, and z is the increase value at each step. By tracking the convergence of the clustering evaluation metrics, the ending number of the ranges is chosen.
We only perform a k-means experiment for the Johnson et al. dataset in order to create a comparable experiment with their work [44]. Hence, only *n_clusters* are tested as a result. For the 6 Protein Families dataset, since the actual number of clusters is known in advance to be 6, the *n_clusters* ranges are selected so as to avoid deviating too much from 6. When choosing ranges for the ChEMBL and DUD-E datasets, convergence is taken into consideration. For the subset dataset of ZINC 15, since it contains a large number of compounds, the ranges for *n_clusters* and *min_cluster_size* have been increased to 900,000 in order to accommodate that.

Table 4.7: Selected Hyperparameter Ranges for each Dataset.

Dataset	**n_clusters**	**threshold**	**min_cluster_size**
Johnson et al.	[5, 100, 5]	-	-
6 Protein Families	[5, 50, 2]	[0.2, 0.8, 0.2]	[5, 100, 5]
ChEMBL_27	[10, 500, 10]	[0.2, 0.8, 0.2]	[5, 100, 5]
DUD-E	[10, 500, 10]	[0.2, 0.8, 0.2]	[5, 100, 5]
ZINC 15	[10, 1000, 20]	[0.2, 0.8, 0.2]	[5, 500, 10]

4.4 Clustering on Johnson et al. Dataset

Our goal for this experiment is to compare the learned compound descriptors of ChemBERTa model + UMAP with the selected best model in Hadipour et al. [44].

We use the same experiment setup with the paper. To this end, we downloaded the dataset they used in [45] and evaluated the clusters with their clustering metrics: the Silhouette Score, Calinski–Harabasz Index and Davies–Bouldin Index. As shown in Table 4.8, our method outperforms their best model on all metrics. More information about all trained models can be seen in Figure 4.6, Figure 4.7 and Figure 4.8.

Table 4.8: k-means clustering results on the Johnson et al. Dataset.

Model + Dimensionality Reduction	**Cluster Size**	**Silhouette ↑**	**Calinski–Harabasz ↑**	**Davies–Bouldin ↓**
ChemBERTa + UMAP (16)	35	**0.416**	**34,646**	0.718
ChemBERTa + UMAP (32)	25	0.414	30,161	**0.697**
ChemBERTa + UMAP (64)	60	0.392	29,478	0.819
Local&Global Features + VAE (32)	50	0.286	10,112	0.999

In this experimental setup, we only cluster compounds with the k-means clustering method, and we don't use the clustering combinations explained in the previous section, since we want a direct one-to-one comparison with [44], and we follow their exact experiment setup for that.

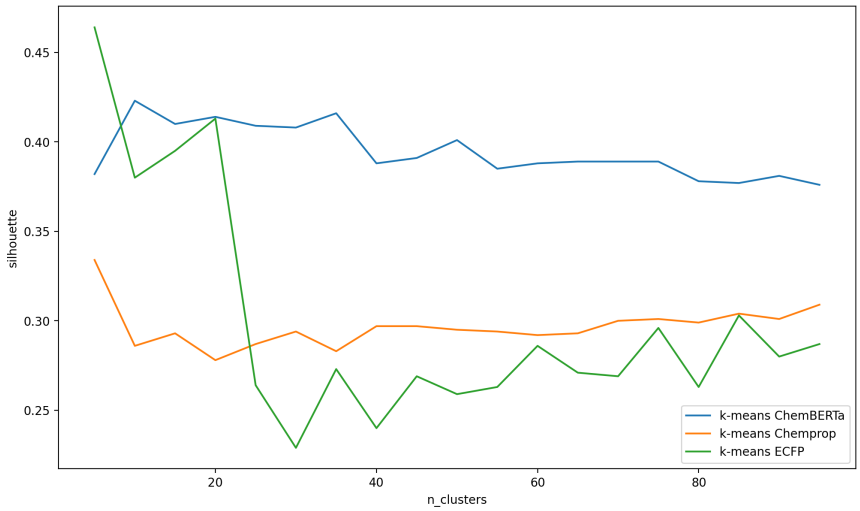

Figure 4.6: Silhouette scores of all models using UMAP(16) with different n_clusters on the Johnson et al. Dataset

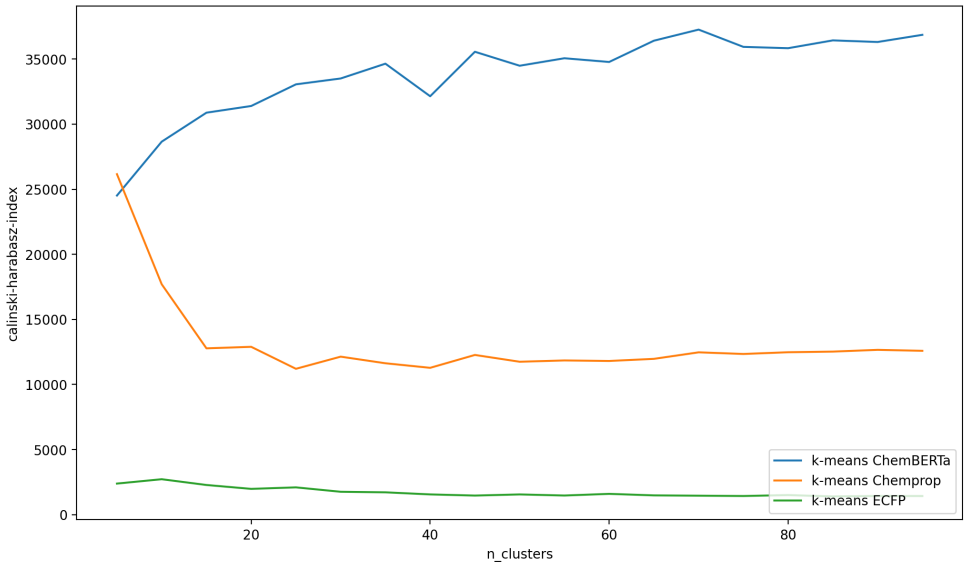

Figure 4.7: Calinski-Harabasz scores of all models using UMAP(16) with different n_clusters on the Johnson et al. Dataset

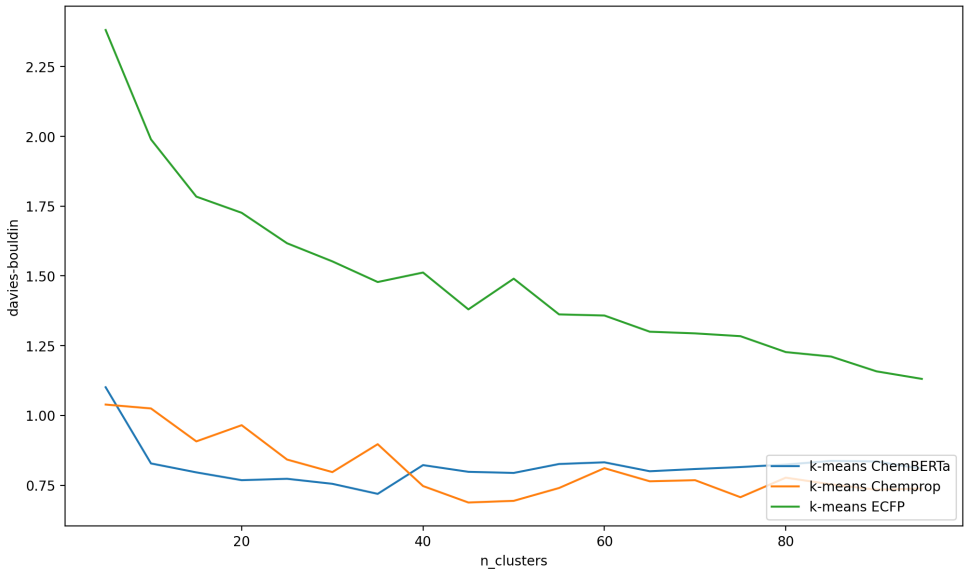

Figure 4.8: Davies Bouldin scores of all models using UMAP(16) with different n_clusters on the Johnson et al. Dataset

4.5 Clustering 6 Protein Families from ChEMBL_29

The experimental configuration for Clustering 6 Protein Families from ChEMBL_29 has 4 clustering algorithms and 3 descriptors, which results in 12 different experiment runs. On top of this, clustering algorithms are tested with 74 different hyperparameter setups. This makes a total of 888 experiments. Only the results of k-means and HAC shown as figures. Other results are shown in the Appendices A.1.

- k-means - *n_clusters*: From 2 to 50 with 2 step size, a total of 25 different values.
- HAC - *n_clusters*: From 2 to 50 with 2 step size, a total of 25 different values.
- Butina *threshold*: [0.2, 0.35, 0.5, 0.8], a total of 4 different values.
- HDBSCAN *min_cluster_size*: From 5 to 100 with 5 step size, a total of 20 different values.

Table 4.9: The results of the clustering algorithms for 3 descriptors using UMAP(16) dimensionality reduction and with the best hyperparameter on the 6 Protein Families dataset

Clustering method	**Model**	**Hyperparameter**	**Value**	**Silhouette ↑**	**ARI ↑**
k-means	ChemBERTa	n_clusters	**6**	**0.31**	0.15
k-means	Chemprop	n_clusters	6	0.28	0.11
k-means	ECFP4	n_clusters	20	0.20	0.09
HAC	ChemBERTa	n_clusters	**6**	**0.2**	0.01
HAC	Chemprop	n_clusters	6	- 0.4	0.01
HAC	ECFP4	n_clusters	6	- 0.6	0.01
Butina	ChemBERTa	threshold	0.8	- 0.21	0.1
Butina	Chemprop	threshold	0.5	- 0.27	0.1
Butina	ECFP4	threshold	0.5	- 0.15	0.2
HDBSCAN	ChemBERTa	min_cluster_size	40	- 0.05	0.01
HDBSCAN	Chemprop	min_cluster_size	20	**0.4**	0.01
HDBSCAN	ECFP4	min_cluster_size	70	0.2	0.03

Since the dataset consists of active compounds on 6 Protein Families, it should have an optimal cluster size of 6. Hence, unlike other experiments, the actual cluster number is known beforehand. Therefore, the experiment configuration is modified to reflect it. The grid-search range for the parameter *n_clusters* is selected to be around the optimal size. Furthermore, the Adjusted Rand Index (ARI) is used instead of the Homogeneity Index for this dataset. This choice is motivated by the fact that the dataset consists of six labels of equal size, and we aim to penalize unnecessary divisions, as explained in Section 2.6.6.

For each clustering algorithm, the best scores and corresponding hyperparameter values are shown in Table 4.9. ChemBERTa descriptors perform better than others at 6 clusters for the k-means algorithm. Surprisingly, ECFP4 descriptors exhibit optimal performance at 20 clusters, despite their low values. This could indicate that they are not fully aware of the underlying dataset. The silhouette scores and ARI values in the HAC, Butina and HDBSCAN clustering algorithms are extremely low for all 3 descriptors. This suggests that those clustering algorithms are not able to cluster the dataset.

Figure 4.9 and Figure 4.10 have a black vertical line that indicates *n_clusters* = 6. In Figure 4.9, in the case of HAC, the scores decrease after *n_clusters* = 6 for all 3 models while the best descriptors are from Chemprop. However, since the scores are close to zero, it can be concluded that HAC does not effectively cluster the compounds. For k-means, the silhouette scores of all models are around 0.3, and they follow a straight line. It is an interesting finding since one would expect to see higher silhouette scores when *n_clusters* = 6.
In Figure 4.10, ChemBERTa descriptors outperform other ones for k-means. A higher ARI value is seen *n_clusters* = 6, as expected. For HAC, none of the models give statistically significant values.

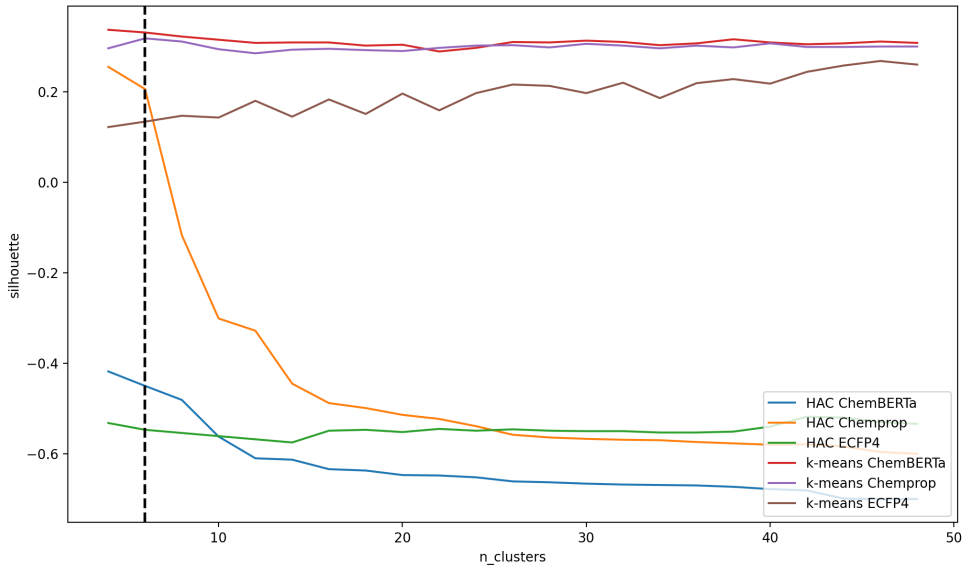

Figure 4.9: Silhouette scores of k-means and HAC results using UMAP(16) with different n_clusters on the 6 Protein Families dataset.

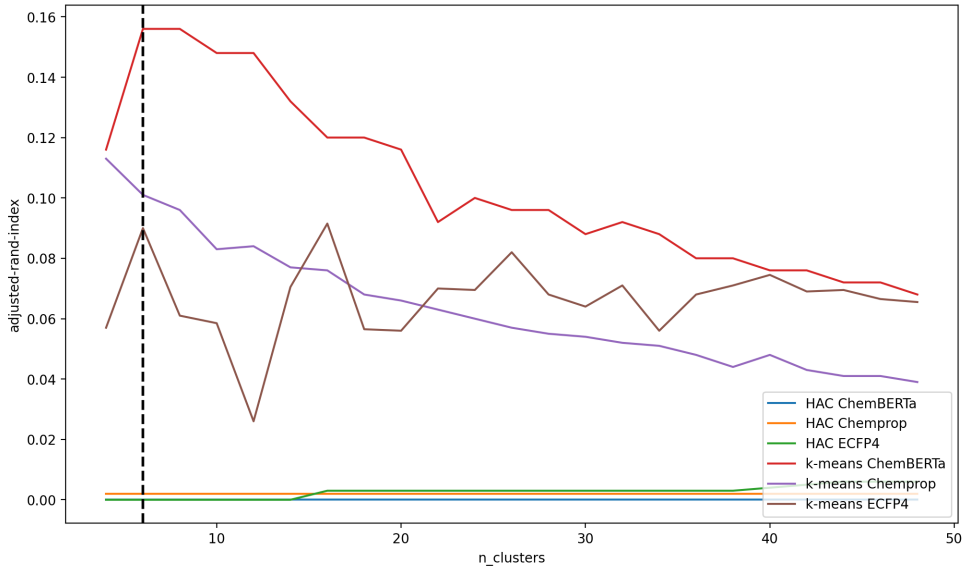

Figure 4.10: ARI values of k-means and HAC results using UMAP(16) with different n_clusters on the 6 Protein Families dataset.

4.6 Clustering on ChEMBL_27

The experimental configuration for clustering on the ChEMBL_27 has 4 clustering algorithms and 3 descriptors, which results in 12 different experiment runs. On top of that, to be able to choose the best hyperparameters, clustering algorithms are tested with 64 different hyperparameter setups. This makes a total of 1488 experiments. Only the results of k-means and HAC for ABL1 dataset are shown. The results of Renin and THB datasets can be found in the Appendices B.2.

- k-means - *n_clusters*: From 10 to 500 with 10 step size, a total of 50 different values.
- HAC - *n_clusters*: From 10 to 500 with 10 step size, a total of 50 different values.
- Butina *threshold*: [0.2, 0.35, 0.5, 0.8], a total of 4 different values.
- HDBSCAN *min_cluster_size*: From 5 to 100 with 5 step size, a total of 20 different values.

The experiments performed on the ChEMBL_27 ABL1 dataset contain 169,997 SMILES compounds. Table 4.11 presents the performance evaluation for these experiments, showing only the results of the best hyperparameter values for each clustering method. Based on both silhouette and homogeneity score, ChemBERTa descriptors perform better in k-means clustering than the other two models. In HAC, the ECFP4 descriptor performs significantly better than the others. The observed value indicates that Butina clustering is unable to effectively cluster compounds, resulting in the assignment of random cluster labels to each molecule. Nevertheless, the ChemBERTa descriptors outperform the other models by a narrow margin. For HDBSCAN clustering, ECFP4 descriptors provide comparatively higher silhouette and homogeneity scores. It indicates that traditional descriptors are a good fit for hierarchical clustering since they are also perform better with HAC.

Figure 4.11 and Figure 4.12 show the effect of choosing various hyperparameter (n_clusters) values on the clustering score for k-means and HAC. We see that the per-

formance of the ChemBERTa descriptors increases with increasing clustering numbers in k-means. However, the score of other descriptors are almost constant. Furthermore, it can be seen that in contrast to traditional ECFP4 descriptors, both learned descriptors perform badly with HAC.

Table 4.10: The results of the clustering algorithms for 3 descriptors using UMAP(16) dimensionality reduction and with the best hyperparameter on ChEMBL_27 ABL1 dataset

Clustering method	**Model**	**Hyperparameter**	**Value**	**Silhouette ↑**	**Homogeneity ↑**
k-means	ChemBERTa	n_clusters	110	**0.316**	**0.8**
k-means	Chemprop	n_clusters	130	0.262	0.4
k-means	ECFP4	n_clusters	30	0.162	0.6
HAC	ChemBERTa	n_clusters	90	- 0.507	0.1
HAC	Chemprop	n_clusters	70	- 0.518	0.1
HAC	ECFP4	n_clusters	190	**0.223**	**0.5**
Butina	ChemBERTa	threshold	0.35	**-0.094**	**0.2**
Butina	Chemprop	threshold	0.35	-0.314	0.1
Butina	ECFP4	threshold	0.5	-0.129	0.1
HDBSCAN	ChemBERTa	min_cluster_size	40	0.254	0.1
HDBSCAN	Chemprop	min_cluster_size	50	0.106	0.1
HDBSCAN	ECFP4	min_cluster_size	50	**0.549**	**0.4**

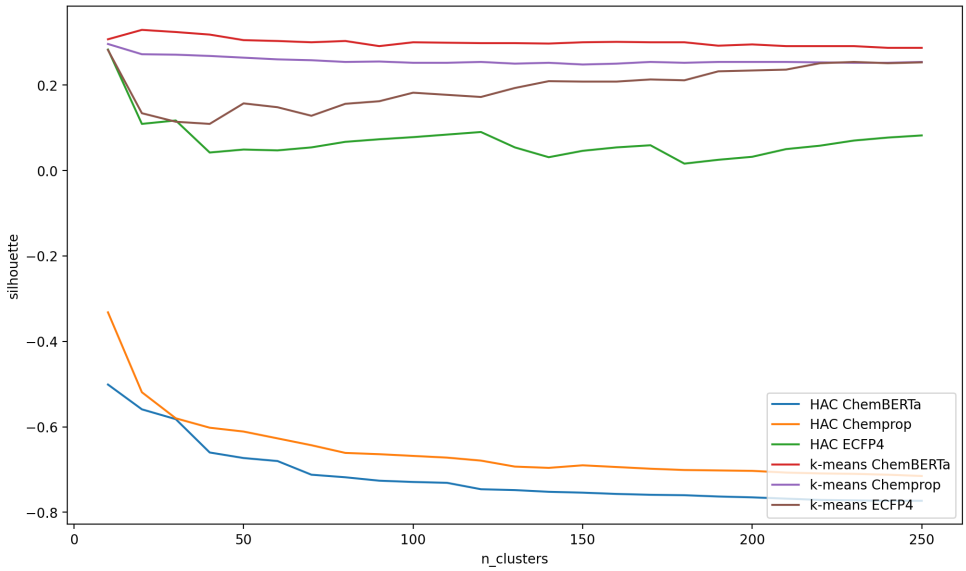

Figure 4.11: Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the ChEMBL_27 ABL1 Dataset

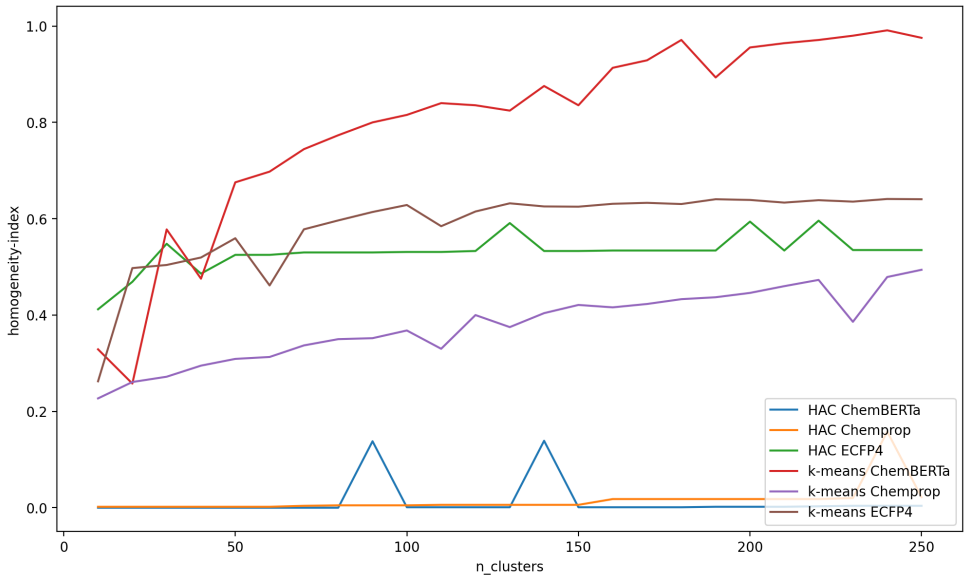

Figure 4.12: Homogeneity scores of k-means and HAC for 3 descriptors with different n_clusters on the ChEMBL_27 ABL1 Dataset

4.7 Clustering on DUD-E

The experiment configuration has 4 clustering algorithms and 3 descriptors, which results in 12 different experiment runs. On top of that, to be able to choose the best hyperparameters, clustering algorithms are tested with 124 different hyperparameter setups. This makes a total of 1488 experiments. Only the results of k-means and HAC for the ABL1 dataset are shown. The results of Renin and THB datasets can be found in the Appendices C.3.

- k-means - *n_clusters*: From 10 to 500 with 10 step size, total 50 different values.
- HAC - *n_clusters*: From 10 to 500 with 5 step size, total 50 different values.
- Butina *threshold*: [0.2, 0.35, 0.5, 0.8], total 4 different values.
- HDBSCAN *min_cluster_size*: From 5 to 100 with 5 step size, total 20 different values.

There are 10,902 SMILES compounds in the experiments performed on the DUD-E ABL1 dataset. The performance evaluation for these studies is shown in Table 4.11, which displays the outcomes of the optimal hyperparameter settings for each clustering technique. ECFP4 performs better than other descriptors for k-means in terms of both silhouette and homogeneity scores. ChemBERTa and Chemprop descriptors for HAC provide near-zero homogeneity scores and negative silhouette values, indicating an inability to cluster compounds well. However, ECFP4 descriptors perform quite well, indicating that there may still be value in traditional methods. All 3 descriptors for Butina and HDBSCAN provide values that are close to zero, indicating unsatisfactory clustering performance.

The impact of selecting various n_clusters values on the clustering score for k-means and HAC is shown in Figures 4.13, 4.14. All 3 descriptors perform at a silhouette score of about 0.3 for k-means, and their performance is nearly consistent for values of n_clusters. Furthermore, HAC's clustering performance is quite poor, with the exception of ECFP4 descriptors. Moreover, we can observe that k-means clustering performs well for all 3 descriptors when looking at the homogeneity score.

Table 4.11: The results of the clustering algorithms for 3 descriptors using UMAP(16) dimensionality reduction and with the best hyperparameter on DUD-E ABL1 dataset

Clustering method	**Model**	**Hyperparameter**	**Value**	**Silhouette ↑**	**Homogeneity ↑**
k-means	ChemBERTa	n_clusters	60	0.29	0.5
k-means	Chemprop	n_clusters	70	0.32	0.6
k-means	ECFP4	n_clusters	110	**0.44**	**0.8**
HAC	ChemBERTa	n_clusters	20	- 0.53	0.05
HAC	Chemprop	n_clusters	150	- 0.62	0.05
HAC	ECFP4	n_clusters	40	**0.25**	**0.7**
Butina	ChemBERTa	threshold	0.35	0.06	0.03
Butina	Chemprop	threshold	0.35	0.04	0.02
Butina	ECFP4	threshold	0.5	**0.09**	**0.03**
HDBSCAN	ChemBERTa	min_cluster_size	60	- 0.05	0.04
HDBSCAN	Chemprop	min_cluster_size	40	- 0.12	0.02
HDBSCAN	ECFP4	min_cluster_size	70	**0.32**	**0.2**

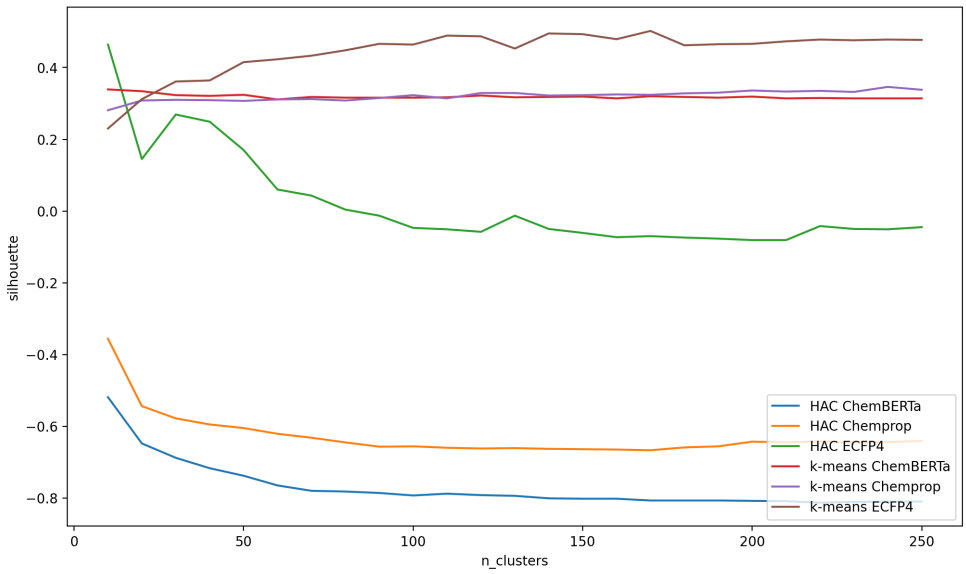

Figure 4.13: Silhouette scores of k-means and HAC for all models with different n_clusters on the DUD-E ABL1 Dataset

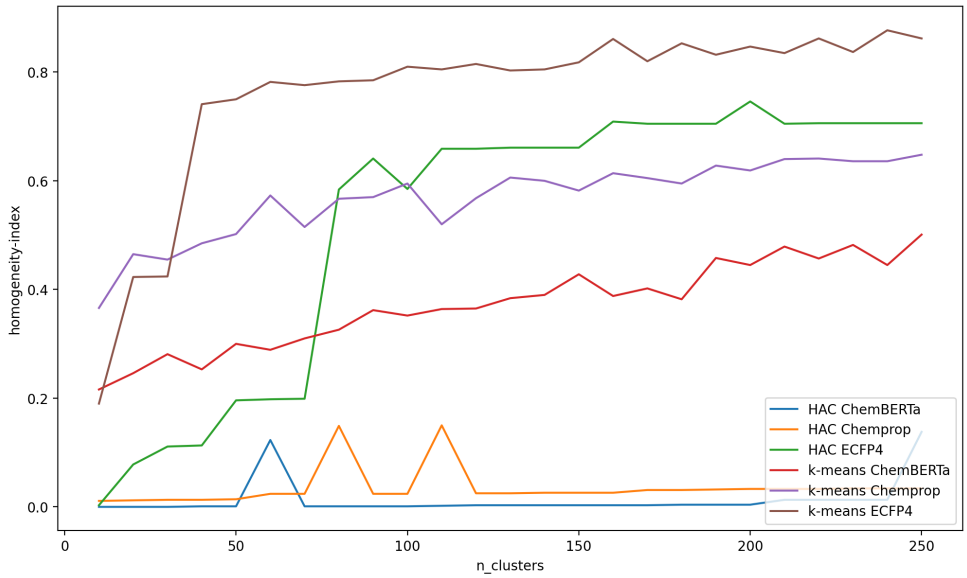

Figure 4.14: Homogeneity scores of k-means and HAC for all models with different n_clusters on the DUD-E ABL1 Dataset

4.8 Clustering on ZINC

The experimental configuration for clustering on ZINC has 4 clustering algorithms and 3 desciptors, which results in 12 different experiment runs. On top of that, to be able to choose the best hyperparameters, clustering algorithms are tested with 154 different hyperparameter setups. This makes a total of 1848 experiments. Only the results of k-means and HAC are shown. The other results can be found in the Appendices D.4. Furthermore, the size of the dataset prevents the BUTINA clustering experiments from being carried out. The justification for this is that they require more RAM than our experiment machine and they are extremely slow to compote for large datasets.

- k-means - *n_clusters*: From 10 to 1000 with 20 step size, a total of 50 different values.
- HAC - *n_clusters*: From 10 to 1000 with 20 step size, a total of 50 different values.
- Butina *threshold*: [0.2, 0.35, 0.5, 0.8], a total of 4 different values.

- HDBSCAN *min_cluster_size*: From 5 to 500 with 10 step size, a total of 50 different values.

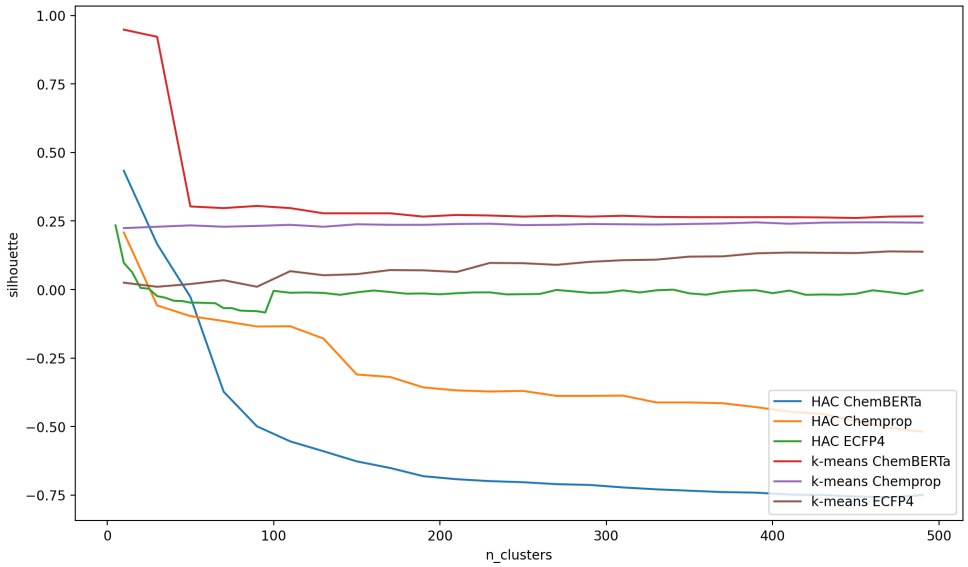

Figure 4.15: Silhouette scores of k-means and HAC for all models with different n_clusters on the ZINC15 Database with 900,000 compounds.

Since the ZINC database is unlabeled, only the silhouette metric is calculated. Moreover, *n_clusters* range is increased to handle effectively the large size of the database. However, due to the saturation after cluster size of 500, Figure 4.15 is clipped at that point. Similar to other experiments, HAC results are negative, which indicates inefficient clustering. For k-means clustering, ChemBERTa descriptors score the highest, though Chemprop descriptors are very close behind. Traditional ECFP4 descriptors show poor results, which can mean that traditional methods aren't scalable for large datasets.

As shown in Table 4.12, our approach exhibits a high degree of scalability when applied to large datasets while simultaneously providing clustering results and computational efficiency. The computational efficiency of ChemBERTa learned descriptors outperforms the Chemprop graph-based model because of the longer duration required for descriptor calculation in the Chemprop model. Despite the relatively slower running time of ChemBERTa compared to ECFP4, its higher silhouette score

Table 4.12: The running time comparison of the end-to-end clustering pipeline for 3 descriptors using UMAP(16) dimensionality reduction on 900,000 compound subset of ZINC15 database. Single running time refers to clustering analysis conducted using a singular value of the hyperparameter, whereas total running time refers to clustering analysis performed across a range of hyperparameter values.

Model	**Single Running Time**	**Total Running Time**
ChemBERTa	2m 45s	3h 36m
Chemprop	5m 15s	5h 10m
ECFP4	2m 30s	3h 7m

makes it a viable alternative to conventional methodologies. Moreover, the comprehensive comparison of running time percentages for each step can be seen in Table 4.13. It is evident that the calculation of descriptors serves as the crucial factor in distinguishing among the three descriptors.

Table 4.13: The running time percentages for each step of the end-to-end clustering for each descriptors on ZINC 15 database

Model	**Calculate Descriptors**	**Clustering**	**Evaluation**
ChemBERTa	10	85	5
Chemprop	25	70	5
ECFP4	8	87	5

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION AND DISCUSSION

This thesis addresses the difficulties brought on by the huge variety of possible drug candidates in the field of drug discovery. The main goal was to use a compound clustering strategy to reduce the resource and time-intensive nature of laboratory testing. The work effectively collected compound descriptors using ChemBERTa, a BERT-based model, allowing for the future use of clustering techniques. Notably, the effectiveness of the suggested methodology was shown by the use of several clustering algorithms, such as k-means and Butina. The learned compound descriptors of the ChemBERTa model are shown to create accurate and effective compound groups, as seen by the examination of the resulting clusters using metrics like Silhouette Score. These results highlight the potential of ChemBERTa descriptors to accelerate compound grouping while improving its precision, opening up a possible path for advancing early drug discovery efforts.

The methodology employed in our study demonstrates a notable level of scalability when implemented on extensive datasets, while concurrently delivering clustering outcomes and computational efficacy. Furthermore, the utilization of a BERT-based model in our method enables the effective utilization of extensive and unlabeled datasets for the compound clustering task. Our findings demonstrate that utilizing a pre-trained model on a substantial dataset of 77 million SMILES yields comparable results to conventional fingerprint methodologies in terms of performance. This would potentially enable models trained with greater datasets to surpass traditional approaches in performance across all tasks.

5.2 FUTURE WORK

- A larger pre-trained model can be used for learned compound descriptors since pre-training size generally affects the model accuracy as stated in [21, 22]. An example model is MoLFormer [70], which is pre-trained on 1,1 billion compounds from the ZINC database. It outperforms its counterpart models in the BACE task of the MoleculeNet benchmark dataset.

- Instead of SMILES, SELFIES notation of the compounds can be used based on the intuition from the SELFORMER model [22]. Although the authors of the ChemBERTa-2 paper made experiments with SELFIES instead of SMILES and found no significant boost in the model performance [21], the result of the SELFORMER paper shows the opposite. The SELFORMER model outperforms both the ChemBERTa-2 model and other graph-based models like chemprop on most of the MoleculeNet classification tasks. As a future study, the learned descriptors of SELFORMER model can be extracted and used for compound clustering.

REFERENCES

[1] W. Ahmad, E. Simon, S. Chithrananda, G. Grand, and B. Ramsundar, "ChemBERTa-2: Towards chemical foundation models," *arXiv [cs.LG]*, Sept. 2022.

[2] S. P. Leelananda and S. Lindert, "Computational methods in drug discovery," *Beilstein J. Org. Chem.*, vol. 12, pp. 2694–2718, Dec. 2016.

[3] J. Vamathevan, D. Clark, P. Czodrowski, I. Dunham, E. Ferran, G. Lee, B. Li, A. Madabhushi, P. Shah, M. Spitzer, and S. Zhao, "Applications of machine learning in drug discovery and development," *Nat. Rev. Drug Discov.*, vol. 18, pp. 463–477, June 2019.

[4] T. I. Oprea, "Virtual screening in lead discovery: A viewpoint," *Molecules : A Journal of Synthetic Chemistry and Natural Product Chemistry*, vol. 7, p. 51, Jan. 2002.

[5] F. Albert Cotton, G. Wilkinson, C. A. Murillo, and M. Bochmann, *Advanced Inorganic Chemistry*. John Wiley & Sons, Apr. 1999.

[6] D. Mendez, A. Gaulton, A. P. Bento, J. Chambers, M. De Veij, E. Félix, M. P. Magariños, J. F. Mosquera, P. Mutowo, M. Nowotka, M. Gordillo-Marañón, F. Hunter, L. Junco, G. Mugumbate, M. Rodriguez-Lopez, F. Atkinson, N. Bosc, C. J. Radoux, A. Segura-Cabrera, A. Hersey, and A. R. Leach, "ChEMBL: towards direct deposition of bioassay data," *Nucleic Acids Res.*, vol. 47, pp. D930–D940, Jan. 2019.

[7] C. Dalvit, P. Pevarello, M. Tatò, M. Veronesi, A. Vulpetti, and M. Sundström, "Identification of compounds with binding affinity to proteins via magnetization transfer from bulk water," *J. Biomol. NMR*, vol. 18, pp. 65–68, Sept. 2000.

[8] H. Öztürk, A. Özgür, and E. Ozkirimli, "DeepDTA: deep drug–target binding affinity prediction," *Bioinformatics*, vol. 34, pp. i821–i829, Sept. 2018.

[9] R. Todeschini and V. Consonni, "Molecular descriptors," *Recent Advances in QSAR Studies*, pp. 29–102, 2010.

[10] F. Grisoni, V. Consonni, and R. Todeschini, "Impact of molecular descriptors on computational models," *Methods Mol. Biol.*, vol. 1825, pp. 171–209, 2018.

[11] K. V. Chuang, L. M. Gunsalus, and M. J. Keiser, "Learning molecular representations for medicinal chemistry," *J. Med. Chem.*, vol. 63, pp. 8705–8722, Aug. 2020.

[12] G. Landrum, "RDKit: A software suite for cheminformatics, computational chemistry, and predictive modeling." `http://www.rdkit.org/RDKit_Overview.pdf`, 2013. Accessed: 2023-7-7.

[13] A. T. Balaban, "Applications of graph theory in chemistry," *J. Chem. Inf. Comput. Sci.*, vol. 25, pp. 334–343, Aug. 1985.

[14] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, V. M. Tran, A. Chiappino-Pepe, A. H. Badran, I. W. Andrews, E. J. Chory, G. M. Church, E. D. Brown, T. S. Jaakkola, R. Barzilay, and J. J. Collins, "A deep learning approach to antibiotic discovery," *Cell*, vol. 181, pp. 475–483, Apr. 2020.

[15] J. Peng, Y. Wang, J. Guan, J. Li, R. Han, J. Hao, Z. Wei, and X. Shang, "An end-to-end heterogeneous graph representation learning-based framework for drug-target interaction prediction," *Brief. Bioinform.*, vol. 22, Sept. 2021.

[16] K. Yang, K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, and R. Barzilay, "Analyzing learned molecular representations for property prediction," *J. Chem. Inf. Model.*, vol. 59, pp. 3370–3388, Aug. 2019.

[17] E. Heid and W. H. Green, "Machine learning of reaction properties via learned representations of the condensed graph of reaction," *J. Chem. Inf. Model.*, vol. 62, pp. 2101–2110, May 2022.

[18] D. Weininger, "SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules," *J. Chem. Inf. Comput. Sci.*, vol. 28, pp. 31–36, Feb. 1988.

[19] M. Krenn, F. Häse, A. Nigam, P. Friederich, and A. Aspuru-Guzik, "Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation," *Mach. Learn. : Sci. Technol.*, vol. 1, p. 045024, Oct. 2020.

[20] S. Wang, Y. Guo, Y. Wang, H. Sun, and J. Huang, "SMILES-BERT: Large scale unsupervised Pre-Training for molecular property prediction," in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, BCB '19, (New York, NY, USA), pp. 429–436, Association for Computing Machinery, Sept. 2019.

[21] S. Chithrananda, G. Grand, and B. Ramsundar, "ChemBERTa: Large-scale self-supervised pretraining for molecular property prediction," *arXiv [cs.LG]*, Oct. 2020.

[22] A. Yüksel, E. Ulusoy, A. Ünlü, and T. Doğan, "SELFormer: Molecular representation learning via SELFIES language models," *arXiv [q-bio.QM]*, Apr. 2023.

[23] D. Rogers and M. Hahn, "Extended-connectivity fingerprints," *J. Chem. Inf. Model.*, vol. 50, pp. 742–754, May 2010.

[24] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv [cs.CL]*, July 2019.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv [cs.CL]*, Oct. 2018.

[26] L. Van Der Maaten, E. O. Postma, H. J. van den Herik, and Others, "Dimensionality reduction: A comparative review," *J. Mach. Learn. Res.*, vol. 10, no. 66-71, p. 13, 2009.

[27] M. Allaoui, M. L. Kherfi, and A. Cheriet, "Considerably improving clustering algorithms using UMAP dimensionality reduction technique: A comparative study," in *Image and Signal Processing*, pp. 317–325, Springer International Publishing, 2020.

[28] Y. Wang, H. Huang, C. Rudin, and Y. Shaposhnik, "Understanding how dimension reduction tools work: an empirical approach to deciphering t-SNE, UMAP, TriMap, and PaCMAP for data visualization," *J. Mach. Learn. Res.*, vol. 22, pp. 9129–9201, Jan. 2021.

[29] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intellig. Lab. Syst.*, vol. 2, pp. 37–52, Aug. 1987.

[30] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," *arXiv [stat.ML]*, Feb. 2018.

[31] L. Gmail and G. Hinton, "Visualizing data using t-SNE." `https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbcl`, 2008. Accessed: 2023-9-1.

[32] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A K-Means clustering algorithm," *J. R. Stat. Soc. Ser. C Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.

[33] D. Arthur and S. Vassilvitskii, "K-means++ the advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, forge.agroparistech.fr, 2007.

[34] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," Sept. 2011.

[35] D. Butina, "Unsupervised data base clustering based on daylight's fingerprint and tanimoto similarity: A fast and automated way to cluster small and large data sets," *J. Chem. Inf. Comput. Sci.*, vol. 39, pp. 747–750, July 1999.

[36] R. Taylor, "Simulation analysis of experimental design strategies for screening random compounds as potential new drugs and agrochemicals," *J. Chem. Inf. Comput. Sci.*, vol. 35, pp. 59–67, Jan. 1995.

[37] D. Bajusz, A. Rácz, and K. Héberger, "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?," *J. Cheminform.*, vol. 7, p. 20, May 2015.

[38] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A Density-Based algorithm for

discovering clusters in large spatial databases with noise," *Knowledge Discovery and Data Mining*, 1996.

[39] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *J. Open Source Softw.*, vol. 2, p. 205, Mar. 2017.

[40] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," in *Proceedings of the international multiconference of engineers and computer scientists*, vol. 1, pp. 380–384, iaeng.org, 2013.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in python," *arXiv [cs.LG]*, pp. 2825–2830, Jan. 2012.

[42] S. Raschka, J. Patterson, and C. Nolet, "Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence," *Information*, vol. 11, p. 193, Apr. 2020.

[43] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.

[44] H. Hadipour, C. Liu, R. Davis, S. T. Cardona, and P. Hu, "Deep clustering of small molecules at large-scale via variational autoencoder embedding and k-means," *BMC Bioinformatics*, vol. 23, p. 132, Apr. 2022.

[45] E. O. Johnson, E. LaVerriere, Office, Emma, M. Stanley, E. Meyer, T. Kawate, J. E. Gomez, R. E. Audette, N. Bandyopadhyay, N. Betancourt, K. Delano, I. Da Silva, J. Davis, C. Gallo, M. Gardner, A. J. Golas, K. M. Guinn, S. Kennedy, R. Korn, J. A. McConnell, C. E. Moss, K. C. Murphy, R. M. Nietupski, K. G. Papavinasasundaram, J. T. Pinkham, P. A. Pino, M. K. Proulx, N. Ruecker, N. Song, M. Thompson, C. Trujillo, S. Wakabayashi, J. B. Wallach, C. Watson, T. R. Ioerger, E. S. Lander, B. K. Hubbard, M. H. Serrano-Wu, S. Ehrt, M. Fitzgerald, E. J. Rubin, C. M. Sassetti, D. Schnappinger,

and D. T. Hung, “Large-scale chemical–genetics yields new m. tuberculosis inhibitor classes,” *Nature*, vol. 571, pp. 72–78, June 2019.

[46] M. R. Karim, O. Beyan, A. Zappa, I. G. Costa, D. Rebholz-Schuhmann, M. Cochez, and S. Decker, “Deep learning-based clustering approaches for bioinformatics,” *Brief. Bioinform.*, vol. 22, pp. 393–415, Jan. 2021.

[47] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande, “MoleculeNet: a benchmark for molecular machine learning,” *Chem. Sci.*, vol. 9, pp. 513–530, Jan. 2018.

[48] E. N. Feinberg, D. Sur, Z. Wu, B. E. Husic, H. Mai, Y. Li, S. Sun, J. Yang, B. Ramsundar, and V. S. Pande, “PotentialNet for molecular property prediction,” *ACS Cent Sci*, vol. 4, pp. 1520–1530, Nov. 2018.

[49] D. Jiang, Z. Wu, C.-Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu, and T. Hou, “Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models,” *J. Cheminform.*, vol. 13, p. 12, Feb. 2021.

[50] A. Donmez, “Large scale machine learning based virtual screening for early drug discovery,” Master’s thesis, Middle East Technical University, Aug. 2021.

[51] J. Cassar, *A big data approach for clustering large chemical datasets*. PhD thesis, 2019.

[52] A. Subakti, H. Murfi, and N. Hariadi, “The performance of BERT as data representation of text clustering,” *J Big Data*, vol. 9, p. 15, Feb. 2022.

[53] R. K. Kaliyar, “A multi-layer bidirectional transformer encoder for pre-trained word embedding: A survey of BERT,” in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 336–340, Jan. 2020.

[54] E. Fenoy, A. A. Edera, and G. Stegmayer, “Transfer learning in proteins: evaluating novel protein learned representations for bioinformatics tasks,” *Brief. Bioinform.*, vol. 23, July 2022.

[55] M. M. Mysinger, M. Carchia, J. J. Irwin, and B. K. Shoichet, "Directory of useful decoys, enhanced (DUD-E): better ligands and decoys for better benchmarking," *J. Med. Chem.*, vol. 55, pp. 6582–6594, July 2012.

[56] T. Sterling and J. J. Irwin, "ZINC 15–ligand discovery for everyone," *J. Chem. Inf. Model.*, vol. 55, pp. 2324–2337, Nov. 2015.

[57] A. Donmez, A. S. Rifaioglu, A. Acar, T. Doğan, R. Cetin-Atalay, and V. Atalay, "iBioProVis: interactive visualization and analysis of compound bioactivity space," *Bioinformatics*, vol. 36, pp. 4227–4230, Aug. 2020.

[58] A. Dalkıran, A. Atakan, A. S. Rifaioğlu, M. J. Martin, R. Ç. Atalay, A. C. Acar, T. Doğan, and V. Atalay, "Transfer learning for drug–target interaction prediction," *Bioinformatics*, vol. 39, pp. i103–i110, June 2023.

[59] A. S. Rifaioglu, E. Nalbat, V. Atalay, M. J. Martin, R. Cetin-Atalay, and T. Doğan, "DEEPScreen: high performance drug–target interaction prediction with convolutional neural networks using 2-D structural compound representations," *Chem. Sci.*, vol. 11, no. 9, pp. 2531–2557, 2020.

[60] Q. Ye, C.-Y. Hsieh, Z. Yang, Y. Kang, J. Chen, D. Cao, S. He, and T. Hou, "A unified drug–target interaction prediction framework based on knowledge graph and recommendation system," *Nat. Commun.*, vol. 12, pp. 1–12, Nov. 2021.

[61] Y.-B. Wang, Z.-H. You, S. Yang, H.-C. Yi, Z.-H. Chen, and K. Zheng, "A deep learning-based method for drug-target interaction prediction based on long short-term memory neural network," *BMC Med. Inform. Decis. Mak.*, vol. 20, p. 49, Mar. 2020.

[62] E. Becht, L. McInnes, J. Healy, C.-A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using UMAP," *Nat. Biotechnol.*, vol. 37, pp. 38–44, Dec. 2018.

[63] A. S. Rifaioğlu, *Deep learning for prediction of drug-target interaction space and protein functions*. PhD thesis, 2020.

[64] C. J. Nolet, V. Lafargue, E. Raff, T. Nanditale, T. Oates, J. Zedlewski, and J. Patterson, "Bringing UMAP closer to the speed of light with GPU acceleration," *AAAI*, vol. 35, pp. 418–426, May 2021.

[65] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking TPU, GPU, and CPU platforms for deep learning," *arXiv [cs.LG]*, July 2019.

[66] "Pytorch: An imperative style, high-performance deep learning library." `https://proceedings.neurips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`. Accessed: 2023-8-20.

[67] C. J. Nolet, D. Gala, A. Fender, M. Doijade, J. Eaton, E. Raff, J. Zedlewski, B. Rees, and T. Oates, "cuSLINK: Single-linkage agglomerative clustering on the GPU," June 2023.

[68] S. Amilpur and R. Bhukya, "Predicting novel drug candidates against covid-19 using generative deep neural networks," *J. Mol. Graph. Model.*, vol. 110, p. 108045, Jan. 2022.

[69] M. K. Thirunavukkarasu, U. Suriya, T. Rungrotmongkol, and R. Karuppasamy, "In silico screening of available drugs targeting Non-Small cell lung cancer targets: A drug repurposing approach," *Pharmaceutics*, vol. 14, Dec. 2021.

[70] J. Ross, B. Belgodere, V. Chenthamarakshan, I. Padhi, Y. Mroueh, and P. Das, "Large-Scale chemical language representations capture molecular structure and properties," *arXiv [cs.LG]*, June 2021.

A.1 Figures of Clustering Results on 6 Protein Families Dataset

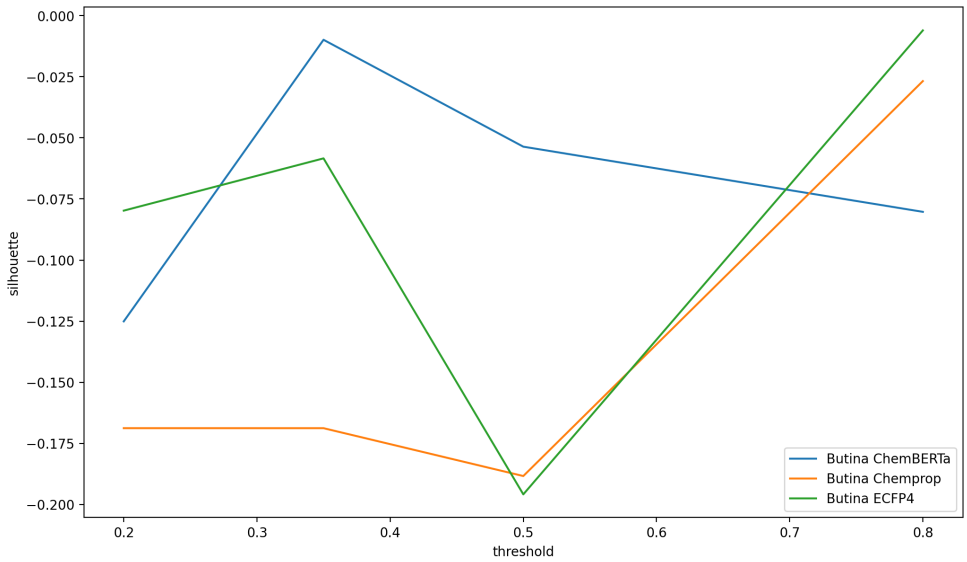

Figure .1: Silhouette scores of Butina using UMAP(16) with different threshold on the 6 Protein Families Dataset

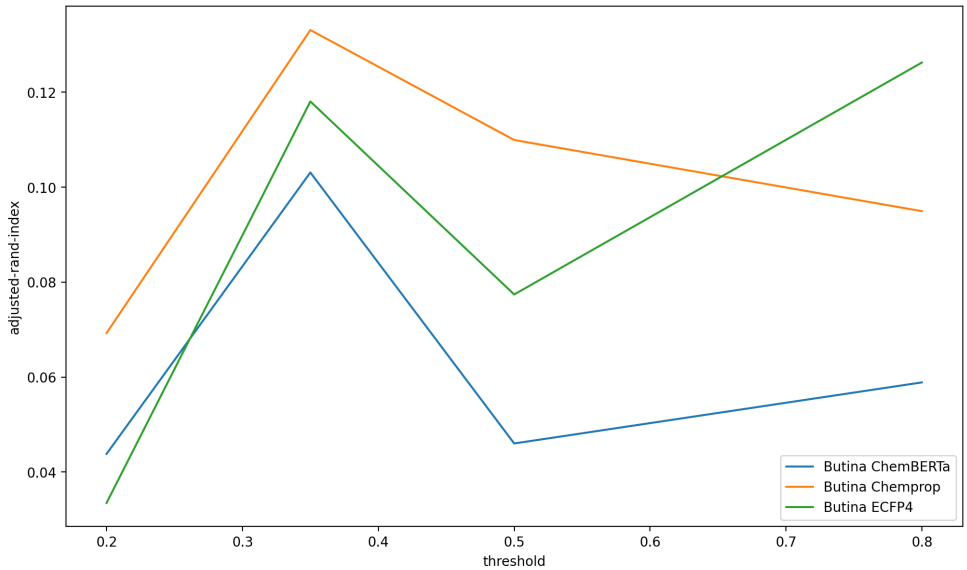

Figure .2: ARI values of Butina using UMAP(16) with different threshold on the 6 Protein Families Dataset

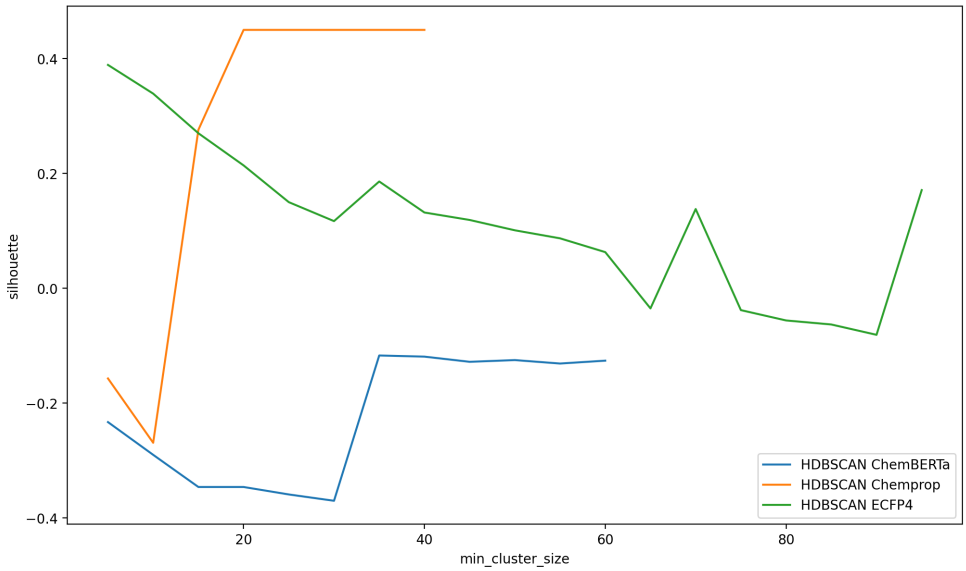

Figure .3: Silhouette scores of HDBSCAN using UMAP(16) with different min_cluster_size on the 6 Protein Families Dataset

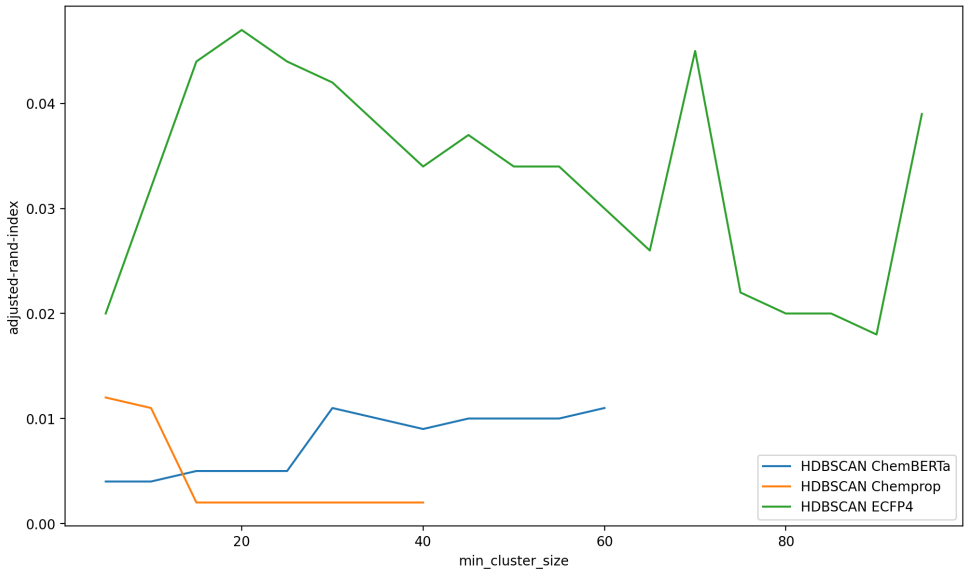

Figure .4: ARI values of HDBSCAN using UMAP(16) with different min_cluster_size on the 6 Protein Families Dataset

B.2 Figures of Clustering Results on Chembl_27 Dataset

B.2.1 Abl1

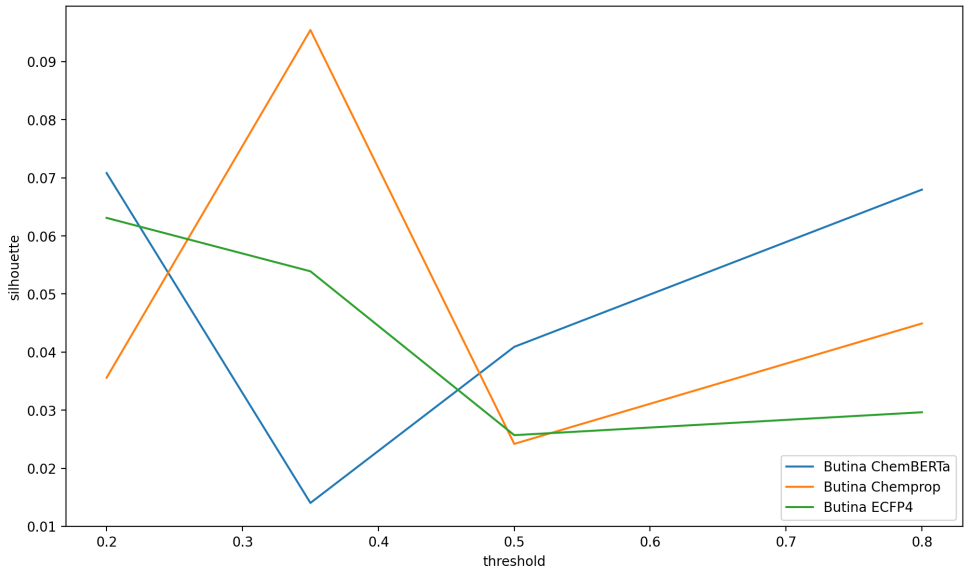

Figure .5: Silhouette scores of Butina clustering for 3 descriptors with different threshold on the Chembl_27 ABL1 Dataset

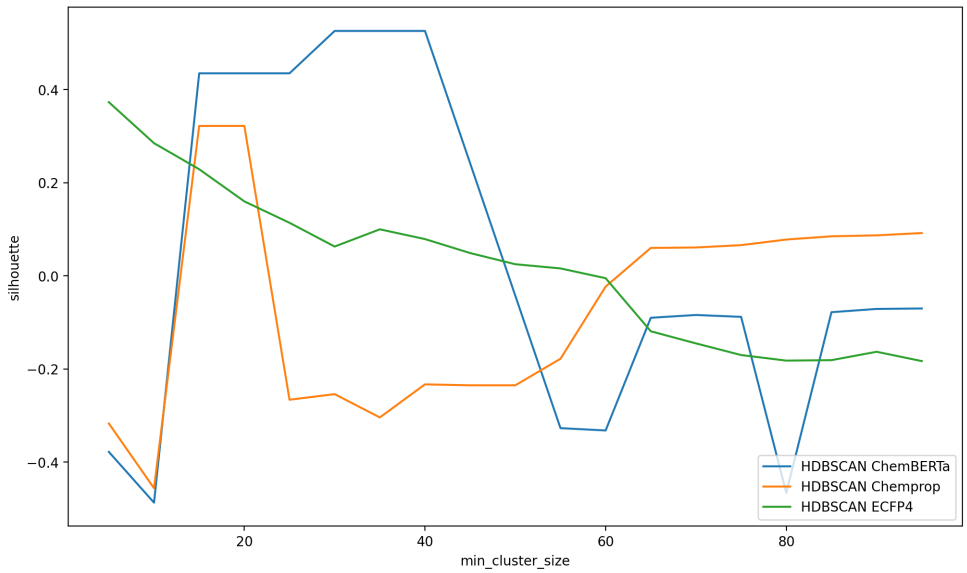

Figure .6: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the Chembl_27 ABL1 Dataset

B.2.2 Renin

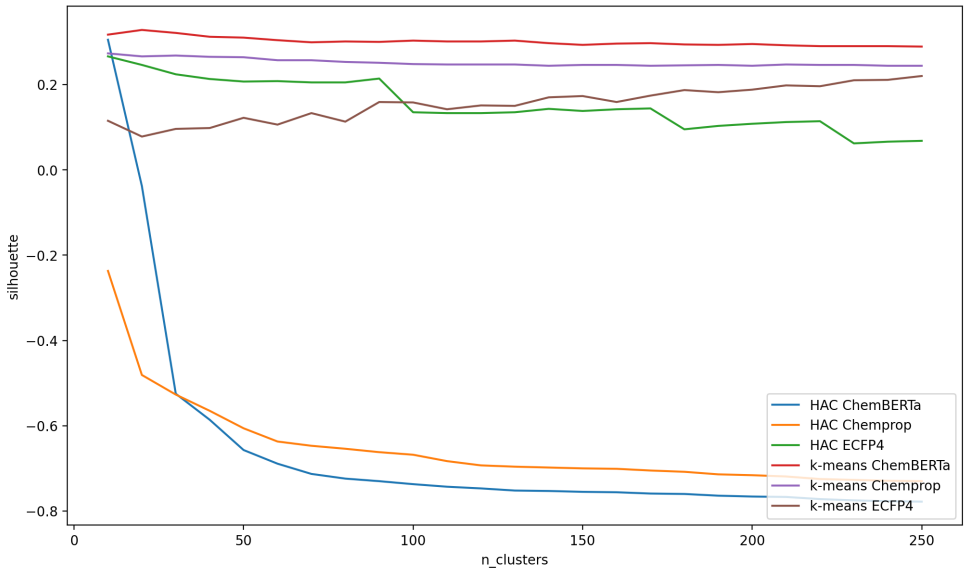

Figure .7: Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the Chembl_27 Renin Dataset

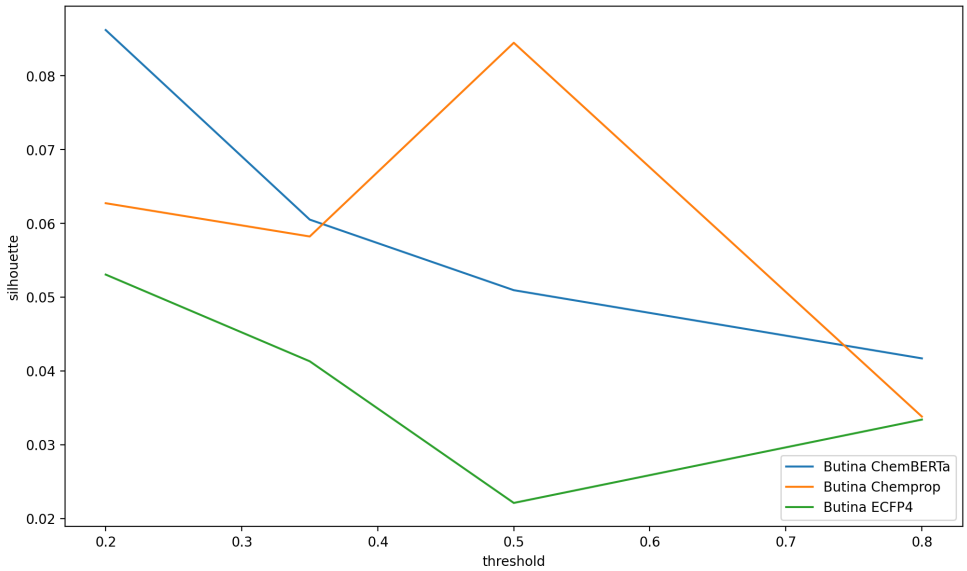

Figure .8: Silhouette scores of Butina clustering for 3 descriptors with different threshold on the Chembl_27 Renin Dataset

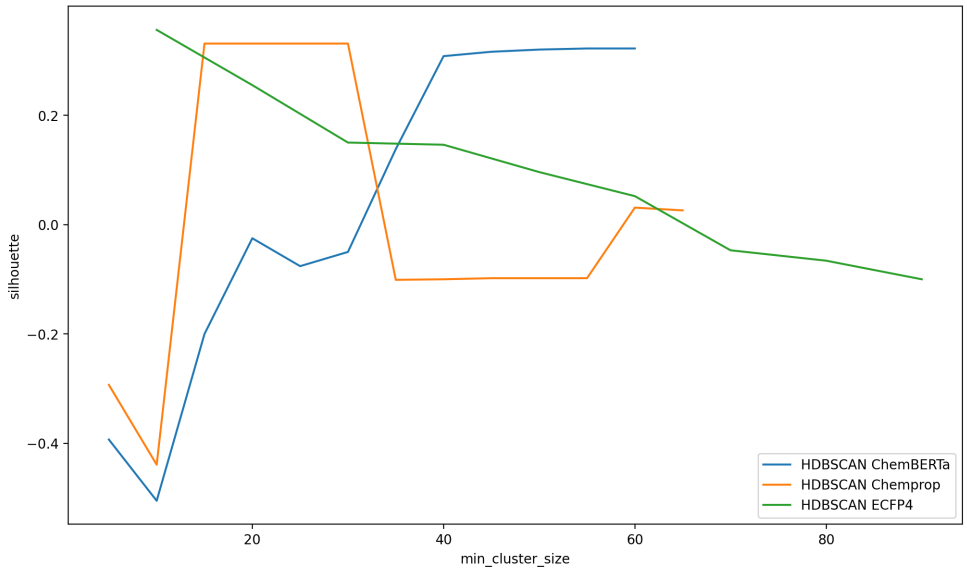

Figure .9: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the Chembl_27 Renin Dataset

B.2.3 Thb

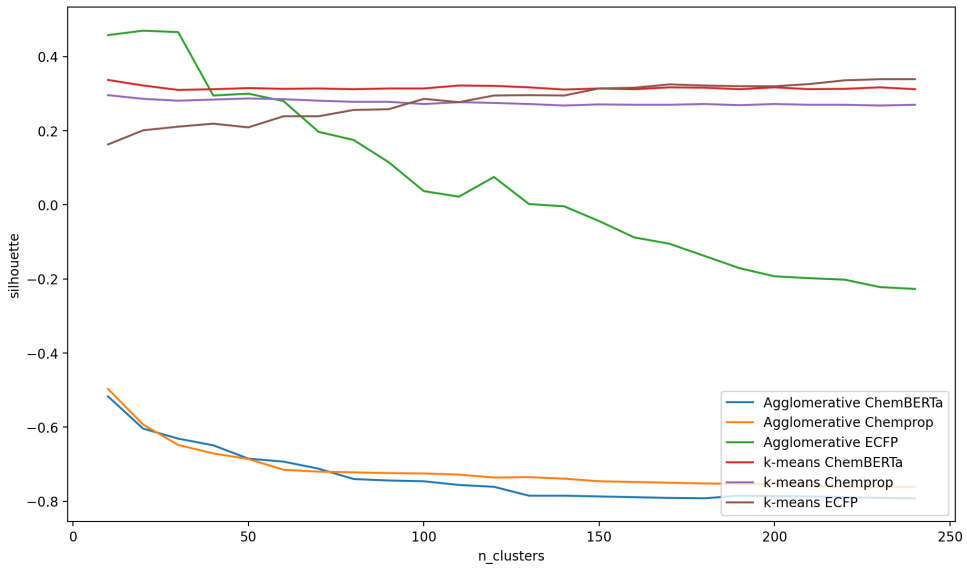

Figure .10: Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the Chembl_27 THB Dataset

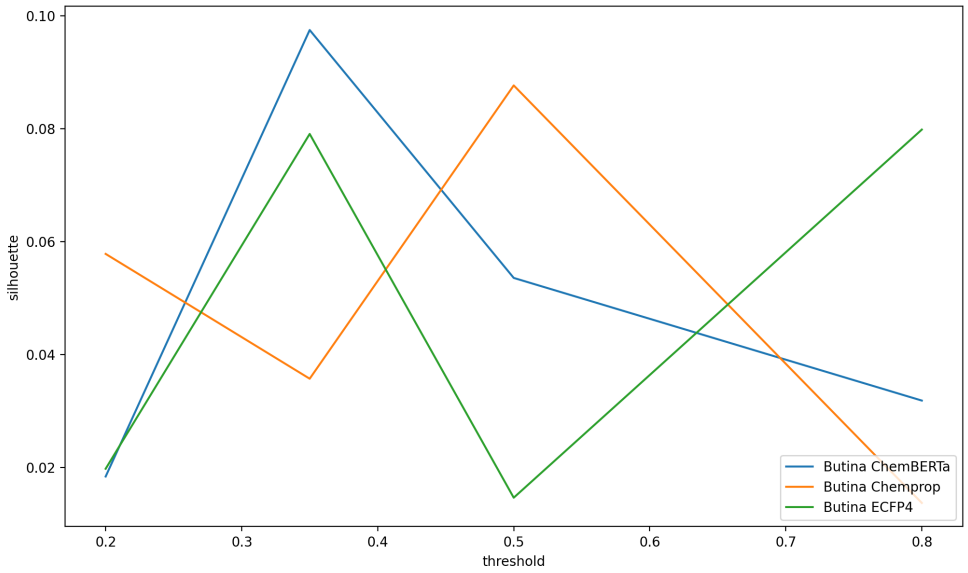

Figure .11: Silhouette scores of Butina clustering for 3 descriptors with different threshold on the Chembl_27 THB Dataset

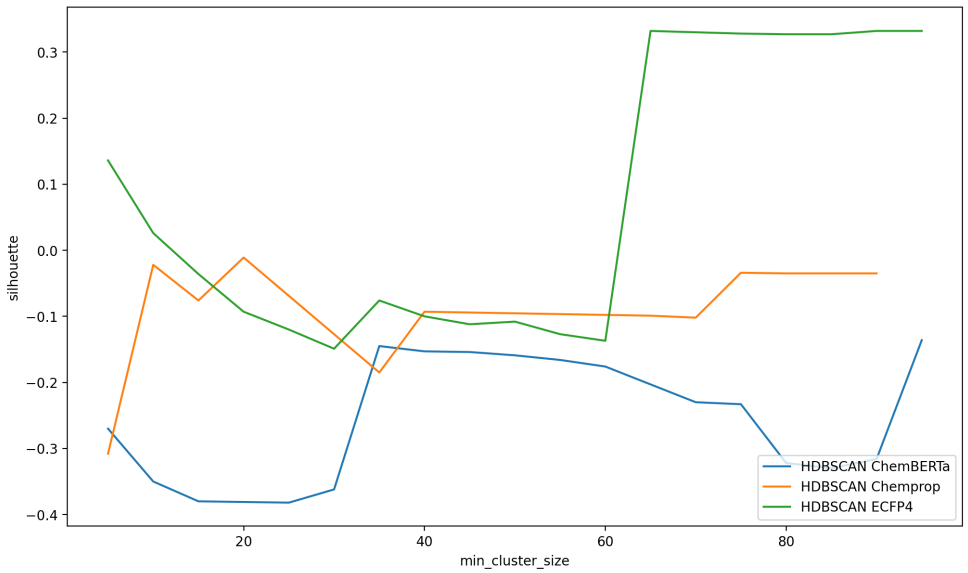

Figure .12: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the Chembl_27 THB Dataset

C.3 Figures of Clustering Results on DUD-E Dataset

C.3.1 Abl1

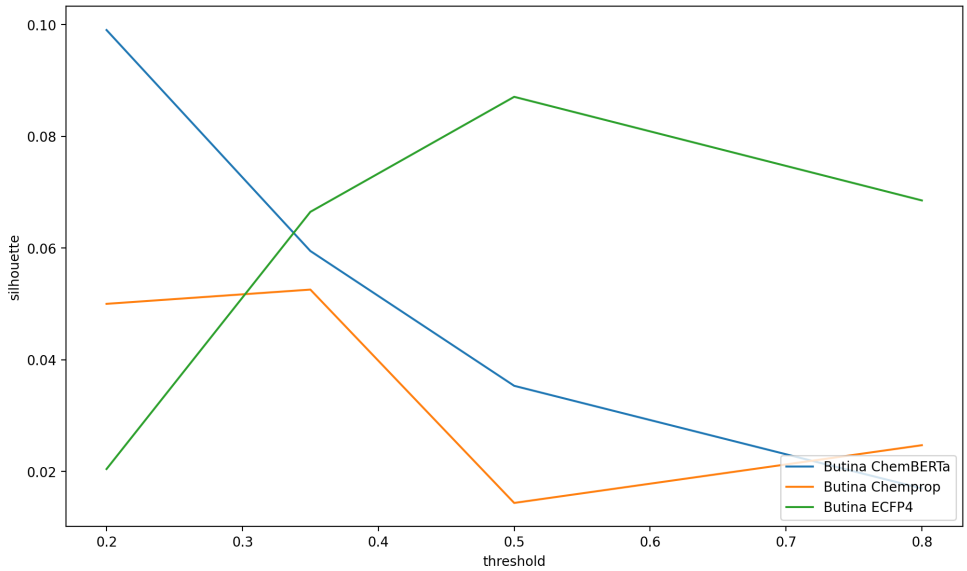

Figure .13: Silhouette scores of Butina clustering for 3 descriptors with different threshold on the DUD-E ABL1 Dataset

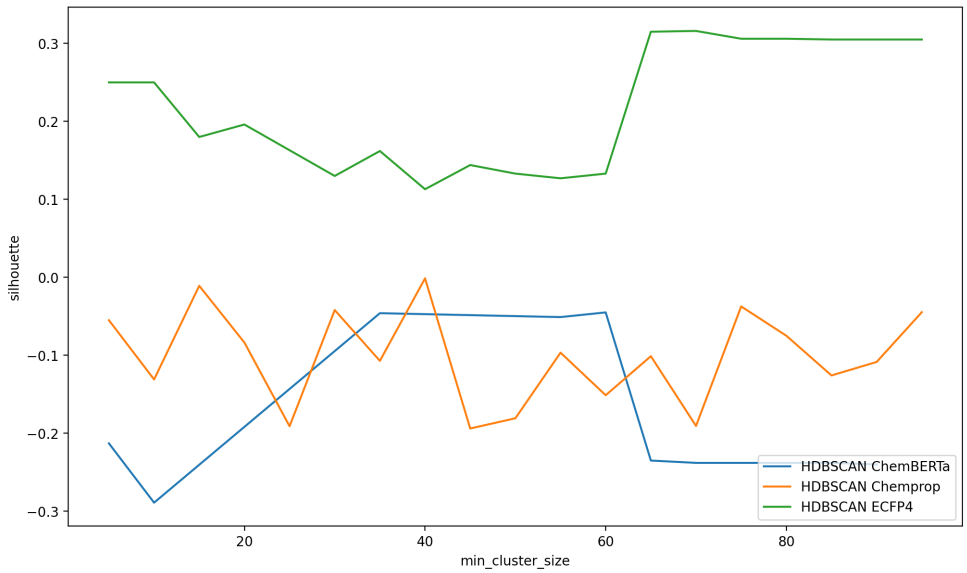

Figure .14: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the DUD-E ABL1 Dataset

C.3.2 Renin

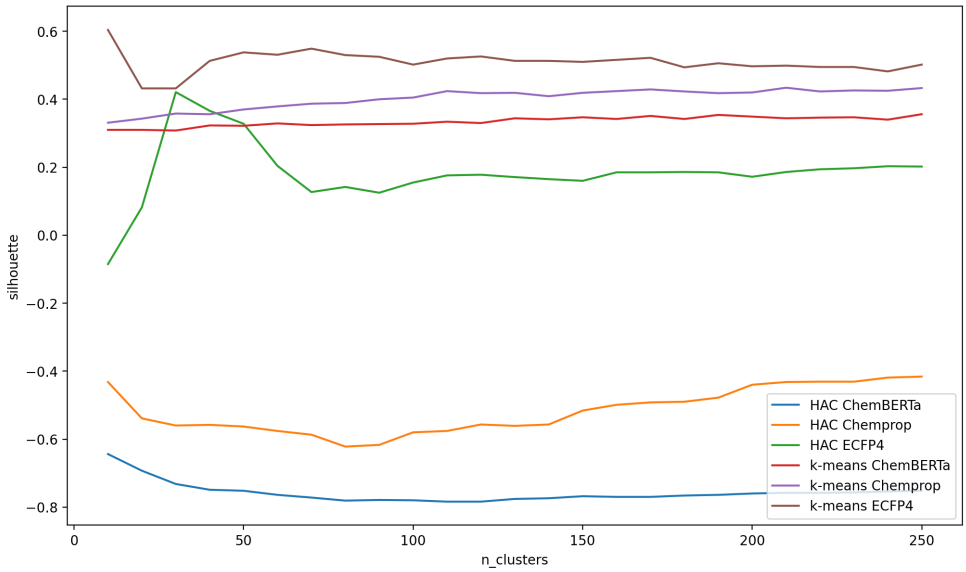

Figure .15: Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the DUD-E Renin Dataset

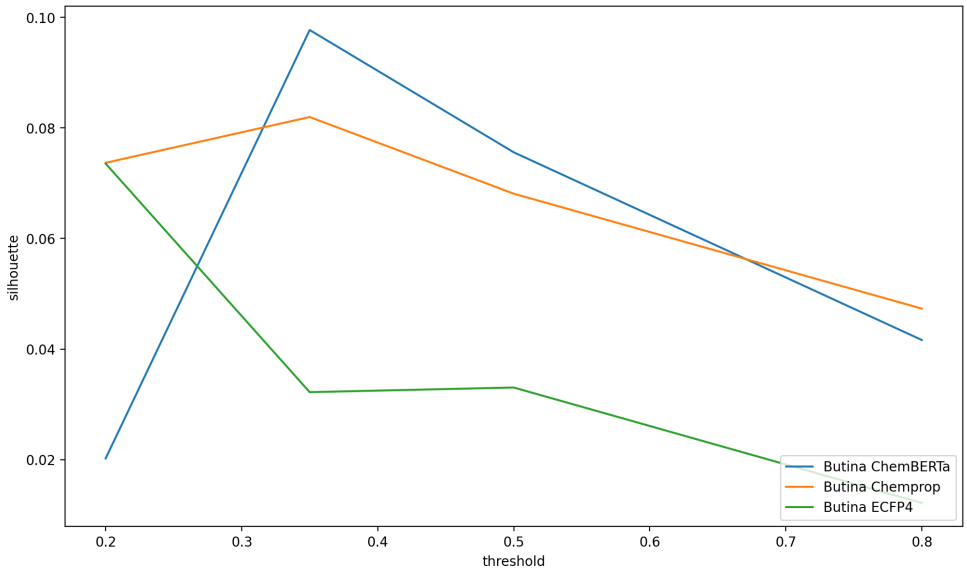

Figure .16: Silhouette scores of Butina clustering for 3 descriptors with different threshold on the DUD-E Renin Dataset

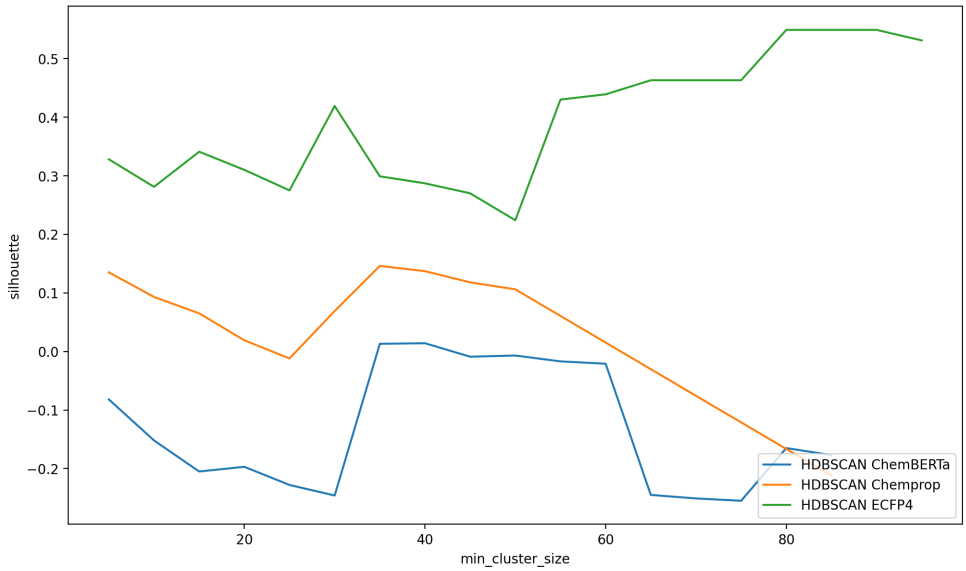

Figure .17: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the DUD-E Renin Dataset

C.3.3 Thb

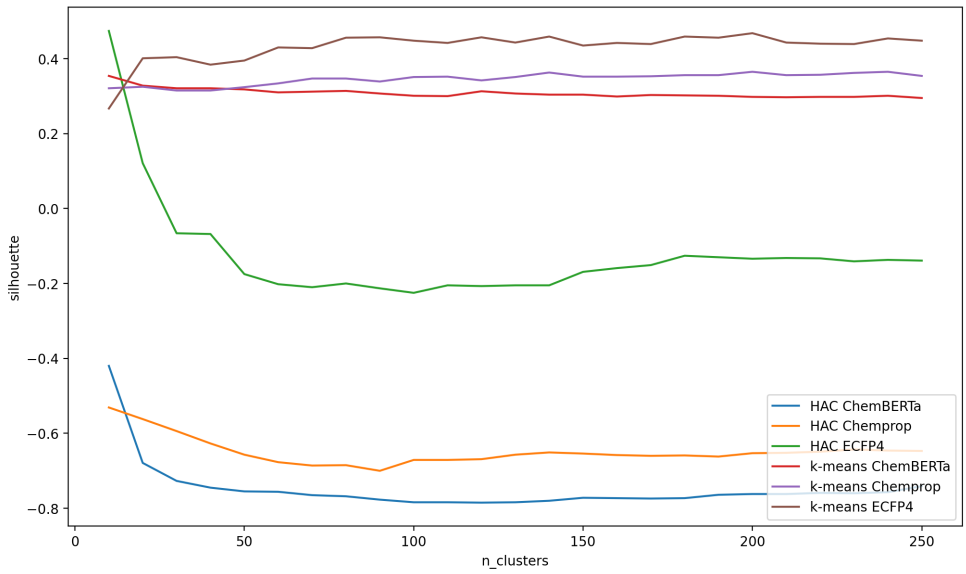

Figure .18: Silhouette scores of k-means and HAC for 3 descriptors with different n_clusters on the DUD-E THB Dataset

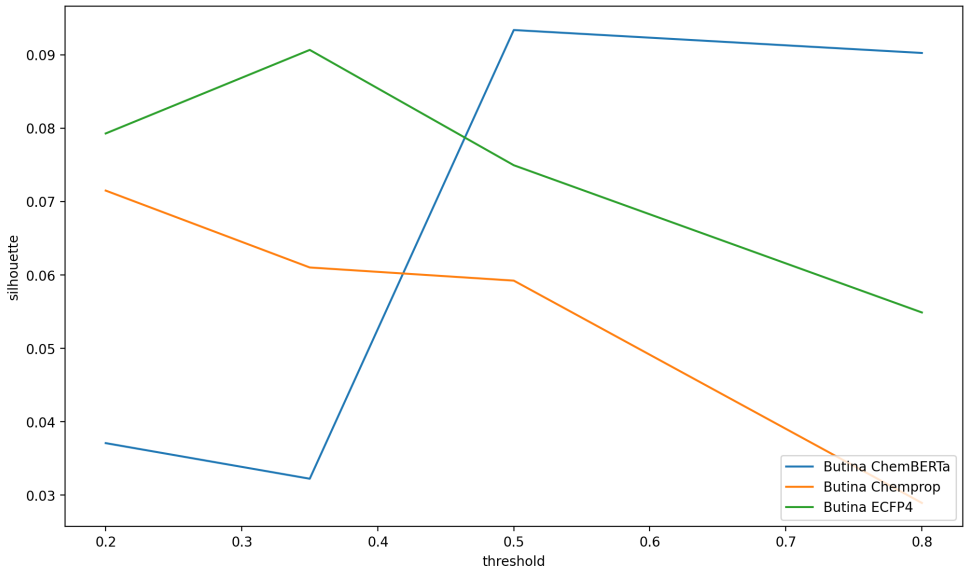

Figure .19: Silhouette scores of Butina clustering for 3 descriptors with different threshold on the DUD-E THB Dataset

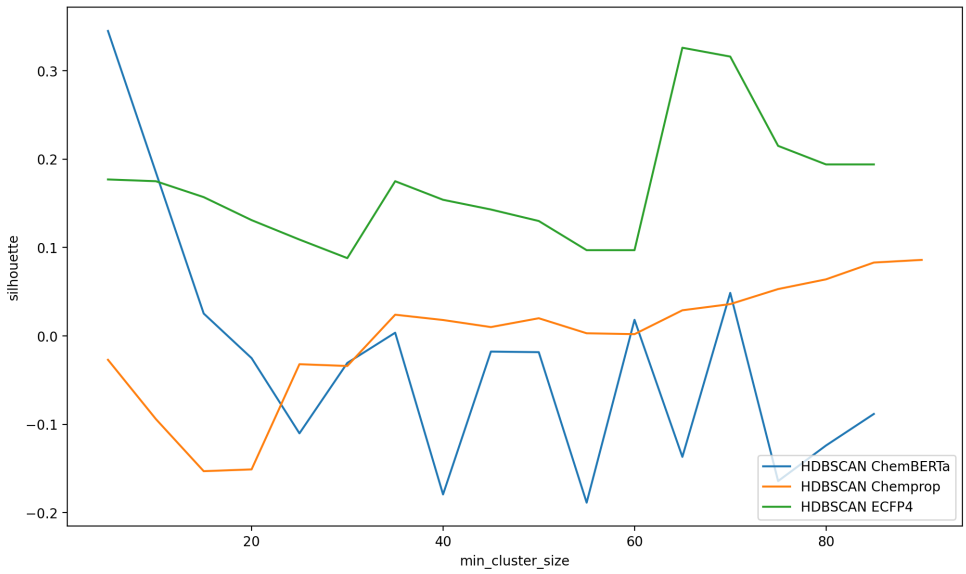

Figure .20: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the DUD-E THB Dataset

D.4 Figures of Clustering Results on ZINC Dataset

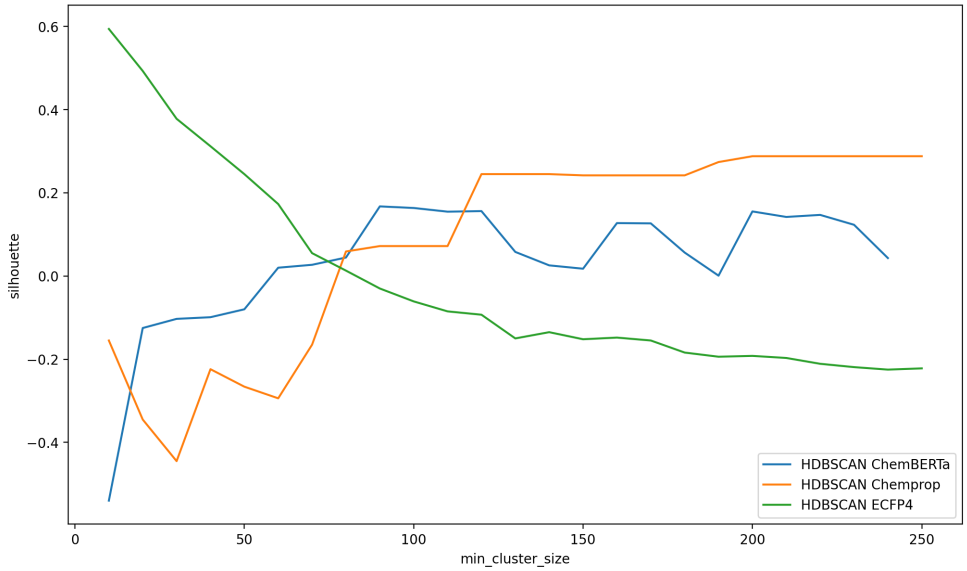

Figure .21: Silhouette scores of HDBSCAN for 3 descriptors with different min_cluster_size on the ZINC15 Database with 900,000 compounds.