



République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique



Université des Sciences et de la Technologie Houari Boumediene
Faculté d'informatique
Département : Systèmes Informatiques
Mémoire de Master
Spécialité :
Calcul haute performance
HPC

Thème :
Enhancing Virtual Screening for the Discovery
of New Drugs through Deep Learning

Sujet Proposé et encadré par :

Mr. SAADI Hocine
Me. MAHDI SILHADI Malika

Mr. BABA-ALI Riadh

Membre du Jury :

Me. DAHMANE AFIFA
Mr. LAHRECHE ABDELMADJID

Présenté par :
BENADJAL Nesrine

Projet : HPC-010

REMERCIEMENT

La plus grande reconnaissance revient à Dieu, qui m'a accordé la santé et la volonté nécessaires pour accomplir ce mémoire de fin d'études. Alhamdoulilah.

Tout d'abord, ce travail n'aurait pas pu voir le jour sans l'aide et l'encadrement de Monsieur SAADI Hocine et Monsieur BABA ALI Raidh. Je les remercie sincèrement pour la qualité exceptionnelle de leur encadrement, pour leur patience, leur rigueur et leur disponibilité.

Je tiens également à exprimer ma gratitude à Madame MAHDI SI-LHADI Malika, ma promotrice, pour son soutien précieux, ses conseils éclairés et sa confiance en mon travail.

Mes remerciements vont aussi à tous les membres respectueux du jury, Madame DAH-MANE AFIFA et Monsieur LAHRECHE ABDELMADJID, pour avoir consacré leur temps à évaluer mon travail.

Mes enseignants du master 1 et du master 2 HPC méritent également mes remerciements sincères. Leurs efforts constants nous ont permis de réaliser ce projet au cours des deux dernières années.

Je souhaite aussi exprimer ma reconnaissance envers mes amis, dont le soutien et l'amitié ont été inestimables tout au long de ce parcours. Votre encouragement et votre présence ont rendu cette aventure d'autant plus enrichissante et mémorable.

Enfin, un grand merci à ma famille, qui m'a toujours encouragé dans mes études et qui a toujours été présente pour moi.

Abstract

The quest for discovering new drugs plays a pivotal role in improving existing treatments and addressing emerging medical challenges. Virtual screening, a computational method for predicting the activity of molecules towards biological targets, has become indispensable in this process. However, persistent challenges in accuracy and efficiency necessitate the integration of cutting-edge technologies. This thesis aims to explore and implement deep learning techniques to elevate virtual screening. The primary objective is to harness the capabilities of deep learning, specifically its prowess in identifying intricate patterns within extensive datasets, to optimize the prediction of pharmacological activity for candidate compounds.

Keywords: Virtual screening, deep learning, CNN (Convolutional Neural Network), GPU (Graphics Processing Unit)

Résumé

La quête de la découverte de nouveaux médicaments joue un rôle essentiel dans l'amélioration des traitements existants et la résolution des défis médicaux émergents. Le virtual screening, une méthode informatique pour prédire l'activité des molécules envers des cibles biologiques, est devenu indispensable dans ce processus. Cependant, les défis persistants en matière de précision et d'efficacité nécessitent l'intégration des technologies de pointe. Cette thèse vise à explorer et à mettre en œuvre des techniques d'apprentissage profond pour améliorer le virtual screening. L'objectif principal est de tirer parti des capacités de l'apprentissage profond, en particulier de son pouvoir à identifier des motifs complexes au sein de vastes ensembles de données, pour optimiser la prédition de l'activité pharmaco-logique des composés candidats.

Mots-clés : Criblage virtuel, apprentissage profond, CNN (Réseau de Neurones Convolutifs), GPU (Unité de Traitement Graphique)

TABLE DES MATIÈRES

Introduction	1
0.1 Introduction générale	1
0.1.1 Problématique	2
0.1.2 Organisation du mémoire	2
1 virtual screening	4
1.1 Introduction	4
1.2 Le processus de découverte des médicaments	4
1.3 Virtual screening	6
1.3.1 Définition	6
1.4 Virtual screening basé sur les ligands (LBVS)	7
1.4.1 Définition	7
1.4.2 Les types du LBVS	8
1.4.3 Les étapes du LBVS	8
1.4.4 Collection, intégration et filtrage des données d'activité biologique .	9
1.4.5 Encodage des ligands	9
1.4.6 La similarité moléculaire	13
1.4.7 Classement	15
1.4.8 Sélection de hits	16
1.5 Virtual screening basé sur la structure (SBVS)	17
1.5.1 Définition	17
1.5.2 Les étapes du SBVS	17
1.6 Molecular docking	18
1.6.1 Définition	18
1.6.2 Relation entre moléculaire docking et virtual screening	19
1.7 Virtual screening à l'ère du Big-Data et le rôle du HPC	19
1.8 Analyse les méthodes existantes pour le virtual screening	20
1.8.1 Comparaison entre le LBVS et le SBVS	20
1.8.2 Principaux avantages du VS	21
1.8.3 Limites du VS	21
1.9 Conclusion	21

2 Apprentissage Profond(Deep learning)	22
2.1 Introduction	22
2.2 Apprentissage Profond(Deep learning)	22
2.2.1 Définition	22
2.2.2 Les réseaux neuronaux artificiels (RNA)	23
2.2.3 La configuration typique d'un RNA	23
2.2.4 perceptron :	24
2.2.5 Les fonctions d'activation	24
2.2.6 Fonctionnement de l'apprentissage profond	25
2.2.7 L'optimisation des modèles d'apprentissage profond	26
2.2.8 L'évaluation des modèles d'apprentissage profond	27
2.3 Les types de deep learning	28
2.3.1 Apprentissage profond entièrement connecté (FCNN)	28
2.3.2 Réseaux de neurones à convolution (CNN)	29
2.3.3 Réseaux de neurones récurrents (RNN)	30
2.3.4 Réseaux de neurones en graphes (GNN)	30
2.4 Sous-apprentissage et sur-apprentissage	31
2.5 Virtual screening et deep learning	33
2.5.1 Les architecteur de deep learning virtual screening	33
2.6 Conlusion	36
3 Conception	38
3.1 Introduction	38
3.2 L'organigramme de l'architecture proposée DL-VS	38
3.3 L'architecture proposée DL-VS	39
3.4 La collecte des données	42
3.4.1 Prétraitement de la base de données (molécules)	42
3.5 Codage des molécules	42
3.6 Calcul d'une valeur représentative pour chaque molécule	43
3.7 Identifier les propriétés communes entre les molécules candidates et les molécules de référence	43
3.8 Le calcule de similarité	44
3.9 La construction d'un ensemble de données d'entraînement	45
3.9.1 Description de l'ensemble de données d'entraînement	46
3.10 Le modèle d'apprentissage profond pour DL-VS	46
3.10.1 Tableau récapitulatif de notre modèle	47
3.10.2 L'algoritheme de modèle d'intelligence artificielle	49
3.11 Comparaison entre notre architecture (DL-VS) et les traveux connexes	50
3.11.1 Les méthodes employées dans notre architecture(DL-VS)	51
3.12 Conclusion	51
4 Réalisation et optimisation	52
4.1 Introduction	52
4.2 Environnement de développement	52
4.2.1 Environnement matériel	52
4.2.2 Environnement logiciel	53
4.3 Acquisition des données	55
4.3.1 Sources des données utilisé	55
4.3.2 La préparation des molécules de prédiction	56

4.4	Résultats	57
4.4.1	Le temps d'exécution :	57
4.4.2	L'évaluation du modèle :	57
4.4.3	Prédiction	58
4.5	Validation des résultats	59
4.6	Le problème de temps d'exécution	60
4.7	Optimisation	61
4.7.1	Programmation parallèle	61
4.7.2	Définition des cartes graphique	61
4.7.3	Fonctionnement des GPU	62
4.8	Programmation cuda	63
4.8.1	Définition	63
4.8.2	PYCUDA	63
4.9	Le pseudo-code de kenel GPU pour DL-VS	63
4.9.1	Définition des fonctions CUDA utilisées dans la fonction calculate_similarity_gpu	65
4.10	Comparaison du temps d'exécution entre le GPU et CPU	66
4.11	Comparaison entre Autodock MPI-Vina et DL-VS	67
4.11.1	Autodock MPI-Vina	67
4.11.2	Fonctionnement de Autodock MPI-Vina	67
4.11.3	Comparaison	68
4.12	Conclusion	69
	Conclusion Générale	70

LISTES DES FIGURES

1.1	Le processus traditionnel de découverte de médicaments	5
1.2	Le processus de virtual screening	7
1.3	Flux de travail hiérarchique du virtual screening	8
1.4	Modélisation d'une molécule sous forme de graphique [11]	10
1.5	Représentation des SMILES [12]	11
1.6	Circular-fingerprint-vector [13]	11
1.7	Les clés MACCS[15]	12
1.8	Morgan circular fingerprints [9]	13
1.9	La similarité moléculaire [19]	13
1.10	Le processus de Classement	16
1.11	La sélection de hits [23]	17
1.12	Le processus de Moléculaire docking [25]	18
1.13	le pipeline du virtual screening basé sur la structure	19
2.1	Apprentissage Profond et les réseaux neuronaux artificiels	23
2.2	Inspiration biologique des réseaux de neurones [34]	24
2.3	Les fonctions d'activation	25
2.4	Les réseaux neuronaux artificiels (RNA)	26
2.5	Matrice de confusion	28
2.6	L'opération de convolution	29
2.7	L'opération de pooling	30
2.8	Réseaux de neurones récurrents (RNN)	30
2.9	L'architacteur de GNN	31
2.10	la methode de la validation croisée	32
2.11	Architecture of Siamese RNN similarity model [22]	33
2.12	Molecular property representation [31]	34
2.13	Organigrammes de comparaison moléculaire [18]	36
3.1	Organigramme de l'architecture proposée DL-VS	39
3.2	L'architecture proposée DL-VS	41
3.3	L'ensemble de données d'entraînement	46
3.4	Résumé du modèle	48
3.5	Comparaison entre l'architecture proposée et les architectures existants	50
4.1	Données incluses dans ChEMBL.	55

4.2	La collection des molécules de référence	56
4.3	Analyse Comparative des Performances : DELL Latitude 5320 vs La Machine de CERIST	57
4.4	L'évaluation du modèle (loss=MAE)	58
4.5	Activité antivirale de C27H35N6O8P contre le COVID-19	59
4.6	Activité antivirale de C43H65N6O9P contre le COVID-19	60
4.7	Activité antivirale de C26H33N6O8P contre le COVID-19	60
4.8	Architecture générale des GPU	62
4.9	Les différents niveaux de mémoire	62
4.10	Le temps d'exécution sur GPU et sur CPU	66
4.11	Accélération du programme (CPU vs GPU)	67

LISTES DES TABLEAUX

1.1	Comparaison entre le criblage virtuel (CS) et le criblage à haut débit (HTS)	6
4.1	Les molécules identifiés par DL-VS pour covid-19	59
4.2	Comparaison entre Autodock MPI-Vina et DL-VS	68
4.3	La variation de temps d'exécution en fonction de nombre de processeurs pour Autodock MPI-Vina	68
4.4	La variation de temps d'exécution en fonction de la taille des données pour DL-VS	69

INTRODUCTION

0.1 Introduction générale

La recherche des nouveaux médicaments constitue un pilier fondamental de la médecine moderne, permettant de combattre des maladies qui affectent la vie de millions d'individus.

Cependant, ce processus s'avère extrêmement complexe, long et onéreux. En effet, il peut s'écouler plus d'une décennie avant qu'un candidat médicament ne parvienne à son approbation et atteigne les patients, tandis que les coûts de développement peuvent atteindre des milliards de dollars.

De plus, le taux d'échec est considérablement élevé. L'utilisation de techniques de biologie structurale, telles que le docking moléculaire et le virtual screening, peut considérablement accélérer et optimiser ce processus.

Le docking moléculaire prédit l'interaction entre une petite molécule et une cible protéique en simulant l'orientation de la petite molécule dans le site de liaison de la protéine[1].

Le virtual screening basé sur la structure utilise des algorithmes informatiques pour identifier des composés dans une vaste bibliothèque de molécules qui ont le potentiel de se lier à la cible protéique[2]. Bien que ces techniques soient puissantes, leur efficacité dépend fortement de la disponibilité d'une structure tridimensionnelle précise de la cible thérapeutique. Cependant, dans des nombreux cas, la structure 3D de la cible n'est pas connue ou est difficile à déterminer expérimentalement. Dans le cas de l'absence de structure 3D de la cible thérapeutique, le virtual screening basé sur la similarité (VSBS) offre une approche alternative prometteuse pour la découverte des médicaments. Ce processus consiste à identifier des composés présentant des propriétés structurales et pharmacologiques similaires à des molécules connues pour interagir efficacement avec la maladie.

Dans le cadre de notre mémoire, nous proposons d'améliorer le processus de virtual screening basé sur la similarité (VSBS) en intégrant l'utilisation des méthodes d'intelligence artificielle comme le Deep learning. Cette approche vise à affiner la mesure de similarité entre les molécules candidates et les molécules de référence, conduisant à une meilleure fiabilité et à de meilleures performances dans l'identification de composés thérapeutiques prometteurs.

0.1.1 Problématique

Le virtual screening basé sur les ligands (LBVS), également connu sous le nom de virtual screening basé sur la similarité, recommande des composés ayant les scores de similarité les plus élevés pour des tests biologiques tels que les tests *in vivo*¹ et *in vitro*². Les méthodes traditionnelles de virtual screening basées sur la similarité présentent des limitations importantes. En particulier, elles se concentrent souvent sur un seul type de similarité, ce qui peut entraîner des recommandations incohérentes lorsque le type de similarité est modifié, ce qui affecte la fiabilité de résultats. Donc, il est crucial de trouver une méthode permettant de déterminer la meilleure façon de calculer la similarité.

De plus, le calcul de la similarité peut être très long, ce qui implique un temps d'exécution énorme qui diminuer les performances.

Pour assurer la diversité des données Il est nécessaire de collecter des données provenant de plusieurs sources. Cependant, chaque source peut avoir sa propre structure relationnelle, et donc les attributs peuvent avoir des noms différents dans des ensembles de données variés. Il est donc essentiel d'avoir une bonne compréhension des ensembles de données pour pouvoir manipuler nos données et sélectionner les attributs qui nous intéressent. Ce que represent un défi à surmonter

L'approche de l'utilisation de deep learning dans le processus du virtual screening, vise à affiner la mesure de similarité entre les molécules candidates et les molécules de référence, conduisant à une meilleure fiabilité et à une amélioration des performances dans l'identification de composés thérapeutiques prometteurs. Virtual screening repose sur l'analyse de vastes bases de données (volume). Il est nécessaire de coder et de transformer les données, souvent sous forme de chaînes de caractères ou de formats spéciaux (variété), en valeurs réelles exploitables par les algorithmes de Deep Learning. Un défi crucial dans ce domaine réside dans la capacité des transformations et du codage des données à capturer fidèlement les propriétés chimiques et biologiques de chaque molécule. Cette capture est essentielle pour obtenir une valeur représentative unique pour chaque molécule et ainsi éviter les conflits de résultats et garantir la fiabilité des analyses. Ces traitements complexes engendrent un temps d'exécution considérablement élevé qui s'accroît d'autant plus avec l'augmentation rapide de la taille des données (vitesse). Cette situation met en lumière un problème de Big_Data³. Ce qui peut limiter l'efficacité et la scalabilité de virtual screening.

0.1.2 Organisation du mémoire

Ce mémoire est organisé en quatre chapitres :

- **Chapitre 1 :** Expliquer le concept de virtual screening en détail, ses objectifs et son importance dans le domaine de la recherche pharmaceutique. Présenter les deux principaux types de "virtual screening" ; basé sur la structure et basé sur les ligants.
- **Chapitre 2 :** Définir l'apprentissage profond et ses principes fondamentaux, incluant les réseaux de neurones artificiels. Présenter les différents types d'apprentissage profond comme CNN, GNN, RNN. Réaliser une revue de la littérature sur les recherches actuelles qui utilisent l'apprentissage profond pour le virtual screening .

1. *In vivo* : Tests sur un organisme vivant

2. *In vitro* : Tests en tube à essai

3. Big data : Ensemble de données volumineux (volume), complexes (variété) et à croissance rapide (vitesse), nécessitant des outils et techniques spécifiques pour leur traitement et analyse.[3]

- **Chapitre 3 :** Le chapitre 3 se concentre sur la conception de l'architecture proposée pour le virtual screening. Cette architecture s'attaque au problème de similarité en combinant plusieurs méthodes de similarité grâce à l'apprentissage profond. De plus, elle propose des méthodes robustes pour calculer les valeurs représentatives des molécules et capturer les propriétés d'interaction entre les molécules candidates et les molécules de référence. Comparer notre architecture aux architectures existantes et justifier le choix des méthodes utilisées.
- **Chapitre 4 :** Décrire la mise en œuvre de l'architecture proposée. Discuter l'impact de passage à échelle sur le temps d'exécution en comparant les performances sur deux machines distinctes avec la variation de la taille de données. Optimiser notre programme en utilisant les GPU et la programmation parallèle avec la plateforme CUDA⁴ et comparer les temps d'exécution entre le CPU et le GPU.

4. CUDA : Compute Unified Device Architecture est une technologie propriétaire développée par NVIDIA, est principalement compatible avec les cartes graphiques NVIDIA. CUDA propose un langage de programmation dérivé du langage C.[4]

CHAPITRE 1

VIRTUAL SCREENING

1.1 Introduction

Le développement de nouveaux médicaments est un processus long et coûteux. Pour optimiser ce processus, les informaticiens ont introduit une approche révolutionnaire qui est le virtual screening.

Dans ce chapitre, nous plongerons dans le monde du virtual screening, explorant les subtilités des approches basées sur le ligand et sur la structure, afin de comprendre la relation entre le virtual screening et le docking moléculaire, ainsi que leurs forces et leurs limites.

1.2 Le processus de découverte des médicaments

La découverte de médicaments est très importante dans le domaine de la biomédecine. C'est un processus complexe qui nécessite une exploration et une expérimentation approfondies. Traditionnellement, la découverte de médicaments commence par l'identification des cibles pour une maladie d'intérêt par le processus dépistage expérimental (high throughput screening HTS). Il s'agit d'une méthode de découverte scientifique utilisée pour trouver un composé principal qui peut être optimisé pour donner un candidat médicament [2], cela permet de tester un grand nombre de composés (jusqu'à 1 million), mais le HTS est coûteux.[5] La deuxième étape consiste à optimiser les composés identifiés afin d'améliorer leur efficacité et d'autres propriétés souhaitées, telles que la solubilité, la toxicité réduite et les effets hors cible. Ensuite, ils doivent passer par une série de tests cliniques pour que les candidats médicaments potentiels deviennent des médicaments approuvés.[2] La figure 1.1 illustre les étapes impliquées dans la découverte d'un nouveau médicament, de l'identification des produits naturels à la mise au jour d'un nouveau médicament.

Les étapes du processus de découverte de médicaments

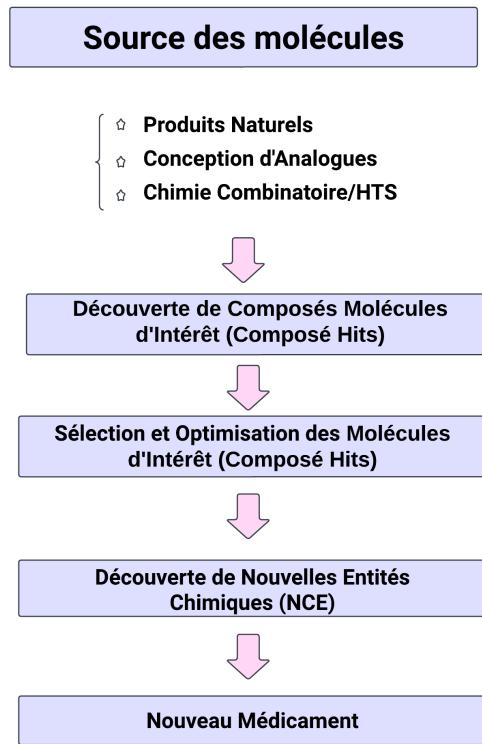


FIGURE 1.1 – Le processus traditionnel de découverte de médicaments

Ce processus présente plusieurs limitations, notamment :

- **Coût** : Les criblages à haut débit (HTS) peuvent être extrêmement coûteux en raison des frais élevés liés à l'équipement, aux réactifs et au personnel requis pour mener des campagnes de high throughput screening.[5]
- **Chronophage** : Le processus traditionnel de découverte de médicaments est chronophage[2], en particulier dans les premières étapes du développement et de l'optimisation des tests. De plus, l'analyse et l'interprétation de la grande quantité de données générées par les expériences HTS peuvent également prendre beaucoup de temps.[5]
- **Couverture limitée de l'espace chimique** : Malgré le screening de vastes datasets de molécules, le HTS peut ne pas échantillonner efficacement l'ensemble de l'espace chimique, ce qui pourrait entraîner la non-détection de nouvelles structures chimiques ou de molécules présentant des activités biologiques uniques.[2]

Pour rationaliser et accélérer le développement de médicaments, des méthodes informatiques ont été largement intégrées dans le processus de conception au cours des trois dernières décennies, dans le but de remplacer le criblage à haut débit (HTS) par une approche notable est le virtual screening (VS).[2]

Le tableau ci-dessous 1.1 compare la technique de virtual screening (VS) au high throughput screening (HTS).

TABLE 1.1 – Comparaison entre le criblage virtuel (CS) et le criblage à haut débit (HTS)

VS	HTS
Recommander $10 - 10^2$ molécule à tester	Recommander $10^8 - 10^{12}$ molécule à tester
Bon marché	Coûteux
Extensible à des dataset très volumineuses	Ne peut pas être étendu à des dataset très volumineuses

1.3 Virtual screening

1.3.1 Définition

Virtual screening (également appelé in silico screening)[6] est une technique qui utilise des ordinateurs pour explorer une dataset contenant plus de 20 à 30 millions de composés[5] et sélectionner les molécules ayant une forte probabilité de se lier à une cible d'intérêt, pour les tests biologiques.[2] Il existe deux principaux types de virtual screening : le virtual screening basé sur la structure (SBVS) qui est utilisé lorsque la structure tridimensionnelle de la protéine est connue, et le virtual screening basé sur le ligand (LBVS) qui est utilisé lorsque la structure tridimensionnelle de la protéine est inconnue.

Chaque méthode possède un processus spécifique, la figure 1.2 décrit en général le processus de virtual screening. L'étape initiale du virtual screening consiste à identifier la protéine cible impliquée dans la maladie [2], appelée récepteur. Ensuite, un grand nombre de molécules potentiels capables d'interagir avec la cible est collecté ou généré (préparation de la base de données). Les molécules peu susceptibles d'être actifs, tels que ceux qui sont toxiques, instables ou trop volumineux sont filtrés[2]. Enfin, la base de données est criblée en utilisant un logiciel pour docker les composés sur la cible et évaluer leur liaison. L'objectif du virtual screening est de réduire le nombre de molécules qui doivent être testées.[5]

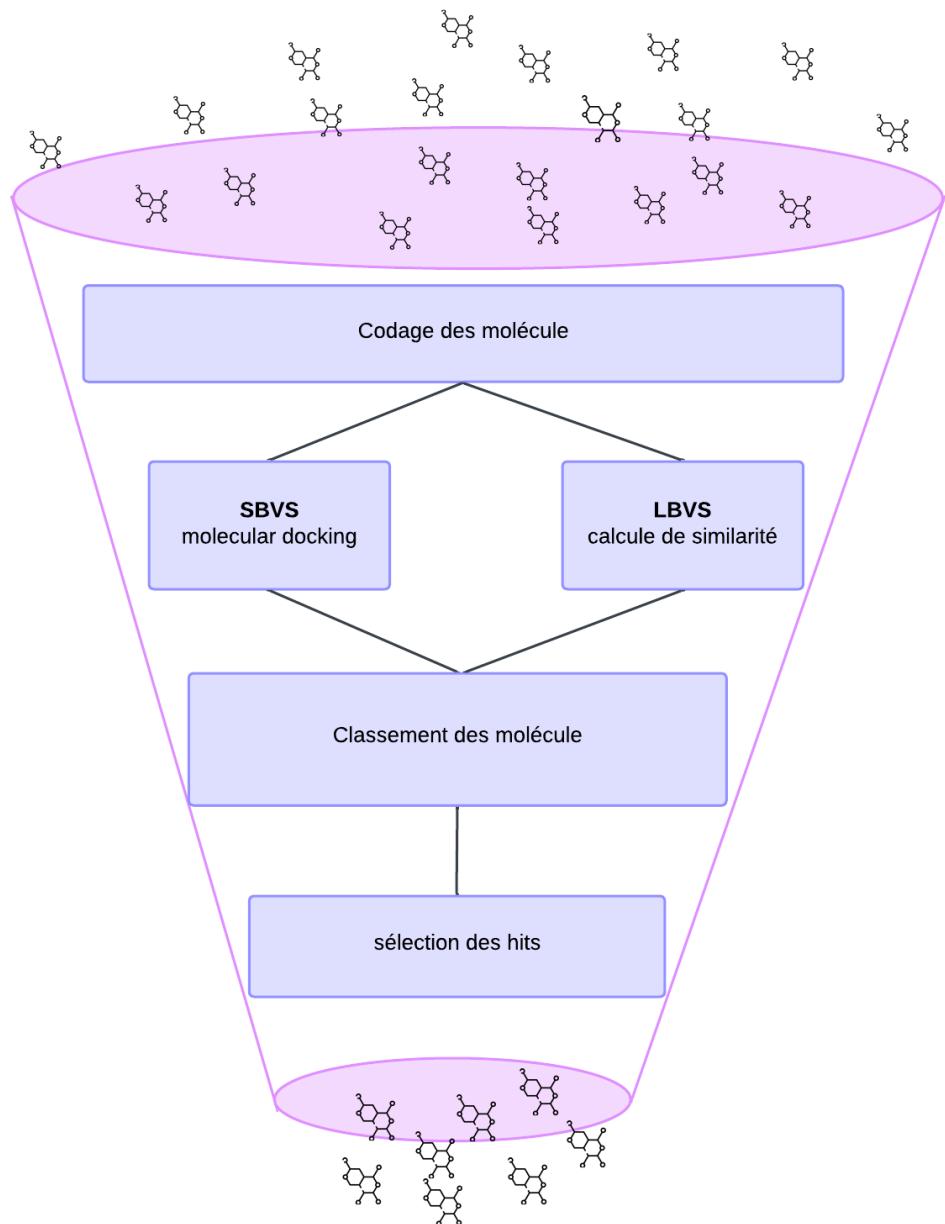


FIGURE 1.2 – Le processus de virtual screening

1.4 Virtual screening basé sur les ligands (LBVS)

1.4.1 Définition

Les méthodes basées sur les ligands constituent une technique utilisée dans le développement de médicaments pour identifier de nouveaux candidats prometteurs. Cette approche s'appuie sur la connaissance de molécules existantes, appelées ligands¹, dont on sait qu'elles interagissent avec une maladie et produisent un effet désiré [2][7]. LBVS consistent à rechercher dans de vastes collections de molécules (les molécules candidates)

1. Ligands : les molécules de référence

celles qui ressemblent le plus aux molécules de référence connues en termes de structure et de propriétés. Cela revient essentiellement à trouver des molécules similaires à celles déjà connues pour être efficaces. [7] La figure 1.3 illustre le flux de travail hiérarchique du virtual screening, en soulignant que les méthodes LBVS sont appliquées lorsque la structure 3D de la cible est inconnue.

LBVS incluent la modélisation QSAR², qui s'appuie principalement sur l'information relative aux molécules.[2] Ces méthodes établissent une relation mathématique entre la structure chimique d'une molécule et son activité biologique.

1.4.2 Les types du LBVS

Il est important de noter que les méthodes LBVS peuvent être classées en deux types principaux :

- **Recherche de similarité et cartographie des pharmacophores** : Ces techniques sont employées lorsque seuls les ligands actifs sont connus. Elles comparent les caractéristiques moléculaires des composés criblés à celles des ligands actifs connus.
- **Méthodes d'apprentissage automatique** : Lorsque des ligands actifs et inactifs sont connus, les approches d'apprentissage automatique deviennent applicables. Ces méthodes analysent de manière exhaustive les propriétés et les interactions moléculaires, offrant une prédiction plus robuste des candidats-médicaments potentiels.

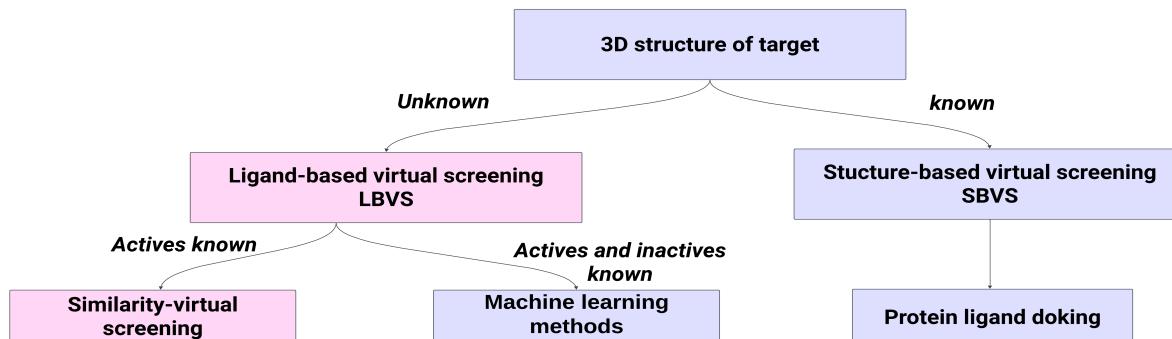


FIGURE 1.3 – Flux de travail hiérarchique du virtual screening

1.4.3 Les étapes du LBVS

1. **Collecte, intégration et filtrage des données de bioactivité** : Il s'agit du processus de collecte d'informations sur l'activité biologique des molécules. L'objectif est de créer un ensemble de données complet contenant des données expérimentales provenant de diverses sources, liées à l'activité de molécules spécifiques. Des données de bioactivité de haute qualité sont essentielles pour l'entraînement et la validation des modèles LBVS, ce qui permet de faciliter la découverte de médicaments.[8]
2. **Représentation moléculaire** : Chaque molécule de la bibliothèque est convertie en un format numérique qui capture ses caractéristiques chimiques essentielles. Cela

2. QSAR : Relation quantitative structure-activité

peut impliquer des descripteurs basés sur la forme, les propriétés physicochimiques ou les sous-structures.[9]

3. **Évaluation de la similitude :** Les méthodes LBVS calculent la similitude entre les molécules candidates et les ligands actifs connus à l'aide de mesures telles que la similitude de fingereprint ou les mesures de distance.
4. **Classement des candidats :** Les candidats sont classés en fonction de leurs scores de similitude calculés. Les molécules avec des scores les plus élevés sont plus susceptibles d'avoir une activité similaire aux ligands connus et sont prioritaires pour des investigations plus approfondies.
5. **Sélection des hits :** Un nombre défini de candidats les mieux classés, susceptibles d'être enrichis en actifs potentiels, sont sélectionnés pour des tests expérimentaux. Cela réduit considérablement le nombre de molécules nécessitant des tests physiques, ce qui permet d'économiser du temps et des ressources.

Étant donné que nous allons travailler avec LBVS nous détaillerons ci-après chaque étape de ce processus.

1.4.4 Collection, intégration et filtrage des données d'activité biologique

Dans l'étape de la découverte de médicaments, l'utilisation efficace des données d'activité biologique est primordiale. Le processus commence par la collecte de données d'activité biologique, où les informations proviennent de diverses bases de données publiques. Vient ensuite la standardisation et la normalisation des données. Ce processus rationalise la grande quantité de données d'activité biologique, la rendant plus facile à gérer et plus informative pour les efforts de développement de médicaments. [8]

1.4.5 Encodage des ligands

L'encodage de ligand consiste essentiellement à traduire la structure complexe d'une molécule en un format compréhensible par l'ordinateur afin de comparer et de prioriser les candidats médicaments potentiels. Cette représentation comprend des informations essentielles qui influencent l'interaction de liaison. Il s'agit d'une étape fondamentale du virtual screening basé sur les ligands (LBVS). Le point de départ de plusieurs encodages de ligands est le graphe moléculaire, où les noeuds et les arêtes représentent les atomes et les liaisons de la molécule.[2]

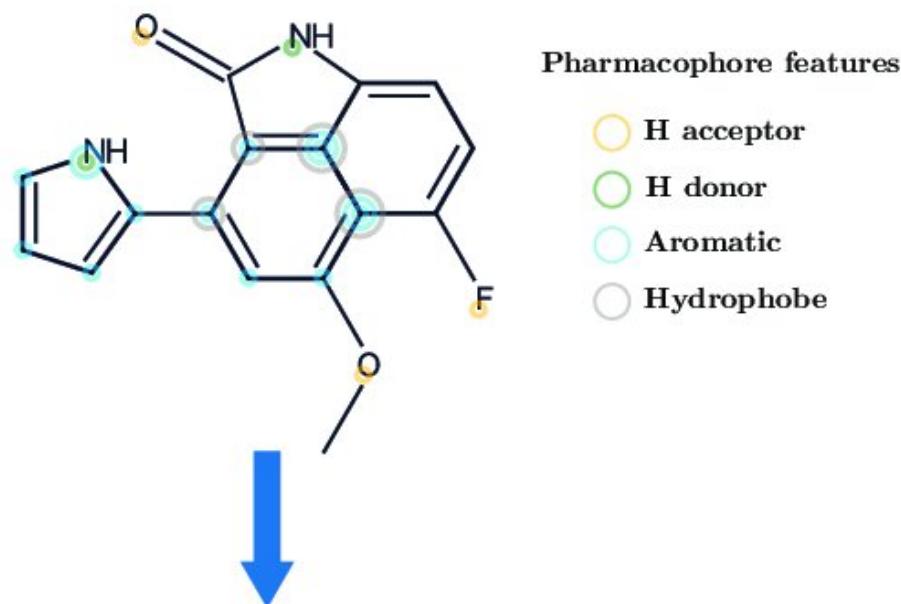
- **Graphe moléculaire :** Le graphe moléculaire est une structure mathématique utilisée pour représenter la structure d'une molécule. Il est constitué de noeuds représentant des atomes et d'arêtes représentant des liaisons chimiques entre eux (voir la figure 1.4)[10]. Le graphe moléculaire peut être encodé à l'aide de deux matrices :

1. **Matrice de caractéristiques X :** fournit une description par atome, où le type d'information stocké dans chaque noeud est défini au préalable[2]. La dimension de X est $N \times D$, où N est le nombre d'atomes et D le nombre de caractéristiques prédéfinies.

2. **Matrice de connectivité** : décrit la structure de la molécule. Son objectif est d'illustrer comment les noeuds sont connectés dans le graphe.[2] Pour stocker ces informations, elle dispose de deux formats courants :

- La matrice d'adjacence A de dimension $N \times N$, où $A_{ij} = 1$ si le noeud i est connecté au noeud j, sinon $A_{ij} = 0$.
- Les coordonnées de dimension $2 \times E$, où E est le nombre d'atomes dans le graphe. Ainsi, la première ligne représente l'index de la source et la deuxième ligne représente l'index des noeuds cibles.

Molecule



Molecular graph representation

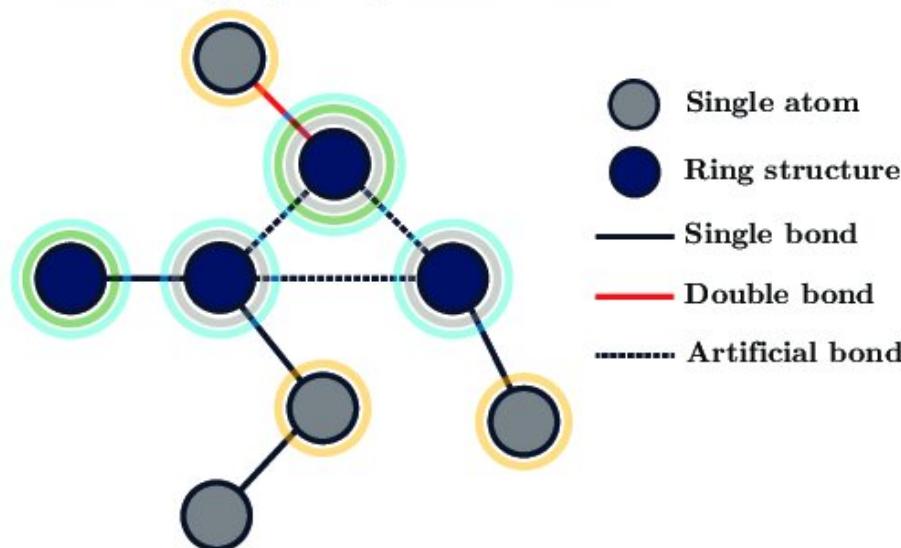


FIGURE 1.4 – Modélisation d'une molécule sous forme de graphique [11]

- **SMILES** : Simplified molecular-input line-entry system (SMILES), développé par Weininger [2], est un moyen efficace de stocker des informations à partir du graphe

moléculaire en utilisant une chaîne de caractères. Le principe de SMILES fonctionne comme un code sophistiqué qui traduit ce réseau en une simple chaîne de texte (linéarisation du graphe moléculaire), comme le montre la figure 1.5. Il procède en répertoriant les atomes un par un, ainsi que la façon dont ils sont connectés, comme en suivant un chemin à travers un labyrinthe. Il n'y a pas une seule façon de suivre ce chemin. Tout comme il existe plusieurs façons de traverser un labyrinthe, il existe souvent plusieurs chaînes SMILES valides pour une même molécule. Cela s'explique par le fait que nous pouvons commencer par différents atomes et choisir différents chemins pour visiter tous les autres. Toutefois, il peut être souhaitable de disposer d'un SMILES unique pour un composé donné, appelé SMILES canonique. La plupart des logiciels disposent de leur propre algorithme de canonisation [2].

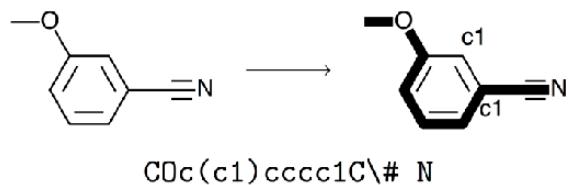


FIGURE 1.5 – Représentation des SMILES [12]

- **Empreinte digitale circulaire (Circular fingerprint)** : Les empreintes digitales constituent un type spécifique d'encodage utilisé dans le virtual screening basé sur les ligands (LBVS).[9] Il s'agit d'une vecteur binaire de longueur fixe utilisée pour représenter la présence d'une sous-structure, ou de caractéristiques au sein d'une structure moléculaire plus large, codée par 1 si présente et par 0 si absente (voir la figure 1.6). Ces sous-structures peuvent être des groupes fonctionnels ou des motifs chimiques spécifiques. Elles permettent de comparer les molécules candidates aux ligands actifs connus en fonction de leurs caractéristiques encodées.

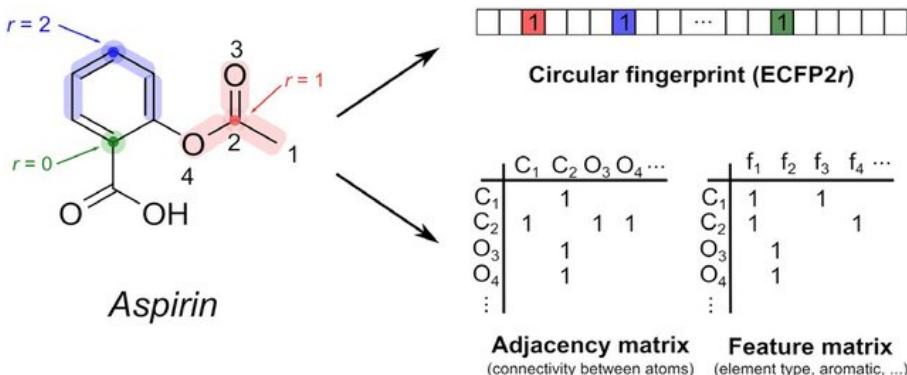


FIGURE 1.6 – Circular-fingerprint-vector [13]

Il existe plusieurs méthodes de codage. Les deux méthodes les plus utilisées sont : **MACCS fingerprint** et **Morgan fingerprint**.

- **MACCS fingerprint** : Le MACCS (Molecular ACCeptance Capability Specification) est l'une des fingerprint structurelles 2D les plus connues et les plus utilisées [14]. Il s'agit d'un sous-ensemble des fragments structurels topologiques,

également appelés clés MACCS, sont des motifs moléculaires spécifiques qui représentent des caractéristiques structurales importantes d'une molécule.[9] La figure 1.7 montre un composé contenant des caractéristiques sous-structurelles détectées par six clés MACCS [14].

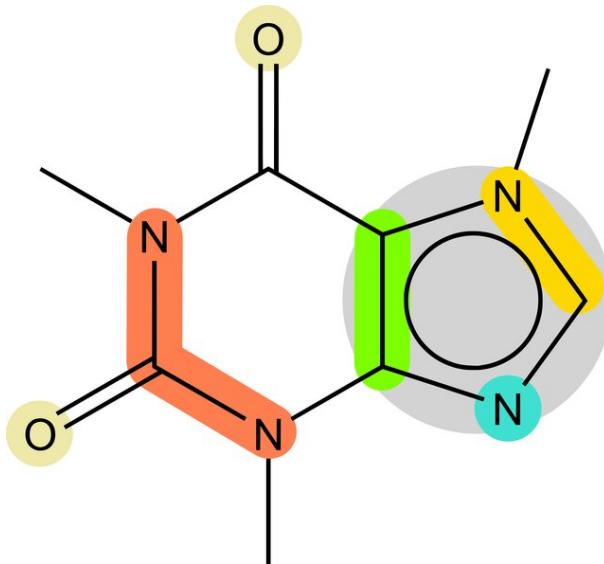


FIGURE 1.7 – Les clés MACCS[15]

- **Morgan circular fingerprints** : Morgan circular fingerprints est une méthode de représentation de la structure moléculaire qui utilise des sous-structures circulaires pour capturer les informations topologiques et chimiques d'une molécule.[16] Elle est couramment utilisée en chimioinformatique pour comparer et classer des molécules en fonction de leur similarité structurale [17].

- * **Fonctionnement** : Morgan fingerprint est une méthode efficace pour représenter la structure moléculaire et comparer des molécules. Elle fonctionne en découplant une molécule en fragments circulaires de tailles croissantes, appelés sous-structures circulaires. Chaque sous-structure représente l'environnement local autour d'un atome central et se voit attribuer une clé Morgan unique qui encode ses caractéristiques structurales [17]. Une fingerprint binaire est ensuite générée, chaque bit correspondant à la présence ou l'absence d'une clé Morgan spécifique dans la molécule (voir la figure 1.6).

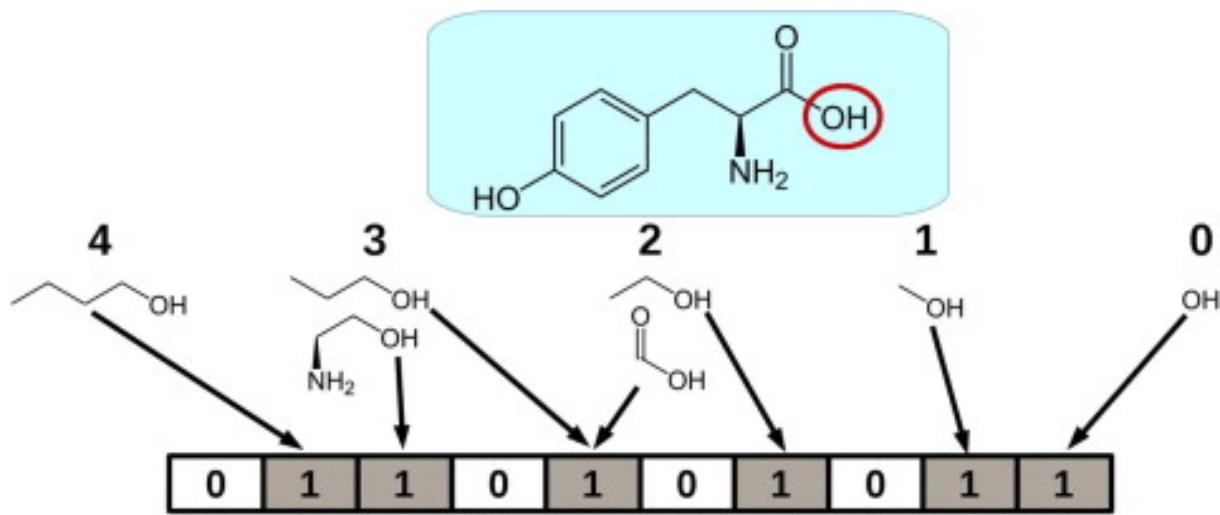


FIGURE 1.8 – Morgan circular fingerprints [9]

1.4.6 La similarité moléculaire

La similarité moléculaire joue un rôle important dans la sélection de bons candidats dans l'industrie de la découverte de médicaments, car on suppose que les molécules structurellement similaires ont des propriétés similaires.[18] Elle fait référence au degré de ressemblance entre deux molécules en fonction de leurs propriétés structurelles, chimiques ou fonctionnelles. Cette similarité peut être évaluée à l'aide de diverses méthodes et mesures (voir la figure 1.9). Les méthodes de similarité utilisées dans le LBVS sont les suivantes :

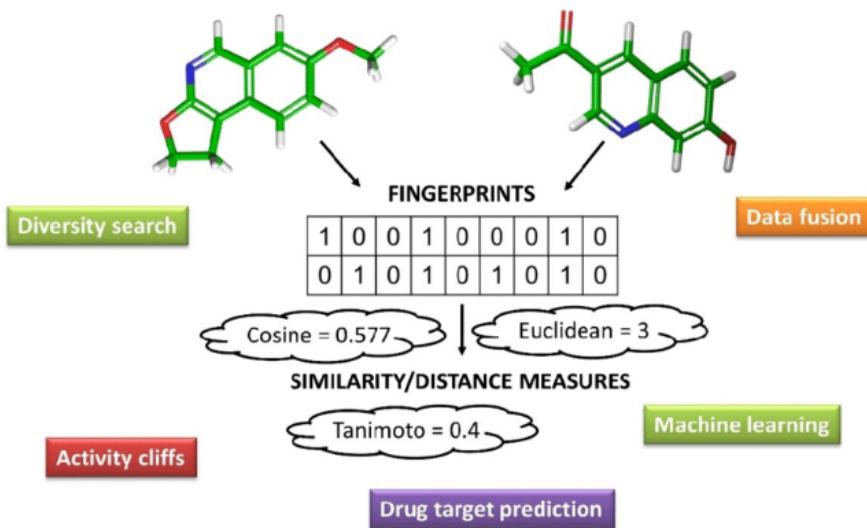


FIGURE 1.9 – La similarité moléculaire [19]

Similarité d'empreintes digitale(fingerprint) :

Ces méthodes codent la structure moléculaire en une empreinte digitale qui capture les informations pertinentes sur la structure de la molécule. La similarité entre deux molécules données A et B est calculée en comparant leurs empreintes digitales à l'aide d'une mesure

de similarité telle que l'indice de similarité de Tanimoto ou cosine similarité (voir la figure 1.9).[5][20]

- **Tanimoto similarité :** La similarité de Tanimoto, également connue sous le nom de coefficient de Tanimoto ou indice de Jaccard, est une mesure utilisée pour déterminer la similitude entre deux ensembles ou vecteurs. Elle est souvent utilisée en chimie informatique pour comparer entre les molécules.[21] La valeur obtenue varie de 0 à 1, où 1 signifie le degré de similarité le plus élevé.[9]

$$T_S(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{(|A| + |B|) - |A \cap B|} \quad (1.1)$$

où $(A \cap B)$ désigne l'intersection et $(A \cup B)$ l'union des ensembles de cases dans les empreintes digitales (A) et (B). De plus, $(|A|)$ et $(|B|)$ représentent le nombre de cases non vides dans les ensembles (A) et (B), respectivement.

- **Similarité cosinus :** Mesure le cosinus de l'angle entre deux vecteurs dans un espace multidimensionnel. Elle est couramment utilisée pour comparer des documents ou des données textuelles dans des tâches de recherche d'information et de traitement du langage naturel (voir la figure 1.9). [9]

$$\text{similarité}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{|\mathbf{A}| |\mathbf{B}|} \quad (1.2)$$

Mesures de distance :

Les mesures basées sur la distance quantifient la dissimilarité en calculant la distance entre les molécules à l'aide de la distance euclidienne, de la distance de Manhattan, de la similarité cosinus ou d'autres distances. Les mesures de distance permettent de prioriser quantitativement les composés pour une évaluation ultérieure.[5]

Méthodes :

- **Distance euclidienne :** Calcule la distance en ligne droite entre deux points dans l'espace euclidien. C'est la mesure de distance la plus courante .(voir la figure 1.9).[9] Soient $p = (p_1, p_2, \dots, p_n)$ et $q = (q_1, q_2, \dots, q_n)$ des vecteurs de dimension n . La distance euclidienne entre p et q est donnée par (1.3) :

$$d(p, q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2} \quad (1.3)$$

- **Distance de Manhattan :** (distance du taxi ou distance du bloc de ville) : Mesure la distance entre deux points en additionnant les différences absolues de leurs coordonnées. Elle est souvent utilisée dans des situations où le mouvement ne peut se faire que le long de lignes de quadrillage.[22] Soient $p = (p_1, p_2, \dots, p_n)$ et $q = (q_1, q_2, \dots, q_n)$ des vecteurs de dimension n . La distance Manhattan entre p et q est donnée par (1.4) :

$$d(p, q) = \sum_{i=1}^n |q_i - p_i| \quad (1.4)$$

- **Distance de Minkowski :** Généralise les distances euclidiennes et de Manhattan. En fonction de la valeur de (p), elle peut représenter différentes mesures de distance. Lorsque (p = 1), il s'agit de la distance de Manhattan ; lorsque (p = 2), il s'agit de la distance euclidienne. Soient $p = (p_1, p_2, \dots, p_n)$ et $q = (q_1, q_2, \dots, q_n)$ des vecteurs de dimension n . La distance minkowski entre p et q est donnée par (1.5) :

$$d(p, q) = \left(\sum_{i=1}^n |q_i - p_i|^p \right)^{1/p} \quad (1.5)$$

- **Distance de Hamming :** La distance de Hamming est une mesure utilisée pour calculer la différence entre deux chaînes de longueur égale. Elle est calculée en comptant le nombre de positions où les symboles correspondants diffèrent voir l'équation (1.6).

$$d(p, q) = \sum_{i=1}^n \delta(p_i, q_i) \quad (1.6)$$

où

$d(p, q)$: distance de Hamming entre les chaînes p et q

p_i : symbole à la position i dans la chaîne p

q_i : symbole à la position i dans la chaîne q

$\delta(p_i, q_i)$: fonction delta de Kronecker (0 si $p_i = q_i$, 1 sinon)

1.4.7 Classement

L'étape la plus cruciale du virtual screening basé sur les ligands consiste à classer les molécules candidates en fonction de leurs scores de similarité calculés, qui mesurent leur similarité aux ligands connus en termes de structure, de forme ou de caractéristiques pharmacophores. Les scores de similarité peuvent être calculés à l'aide de différentes méthodes, telles que les empreintes digitales, la forme ou des approches basées sur le champ. L'étape de classement est cruciale car elle détermine quels composés sont plus susceptibles de présenter une activité similaire aux ligands connus et sont prioritaires pour une investigation plus approfondie. La figure 1.10 illustre qu'au cours de ce processus, plus le score de similarité est élevé, plus la chance de trouver un hit est grande. Cependant, l'étape de classement présente également des défis et des limitations. Par exemple, les scores de similarité peuvent ne pas refléter l'activité biologique réelle des composés, car ils sont basés sur des représentations moléculaires simples qui ne tiennent pas compte des interactions complexes entre les ligands et la cible. [7] De plus, l'étape de classement peut manquer certains nouveaux composés qui ont des scores de similarité faibles mais une activité élevée, car ils peuvent avoir des modes de liaison ou des mécanismes d'action différents.[7] Par conséquent, l'étape de classement doit être suivie d'une étape de validation, où les composés les mieux classés sont testés expérimentalement pour confirmer leur activité et leur sélectivité.

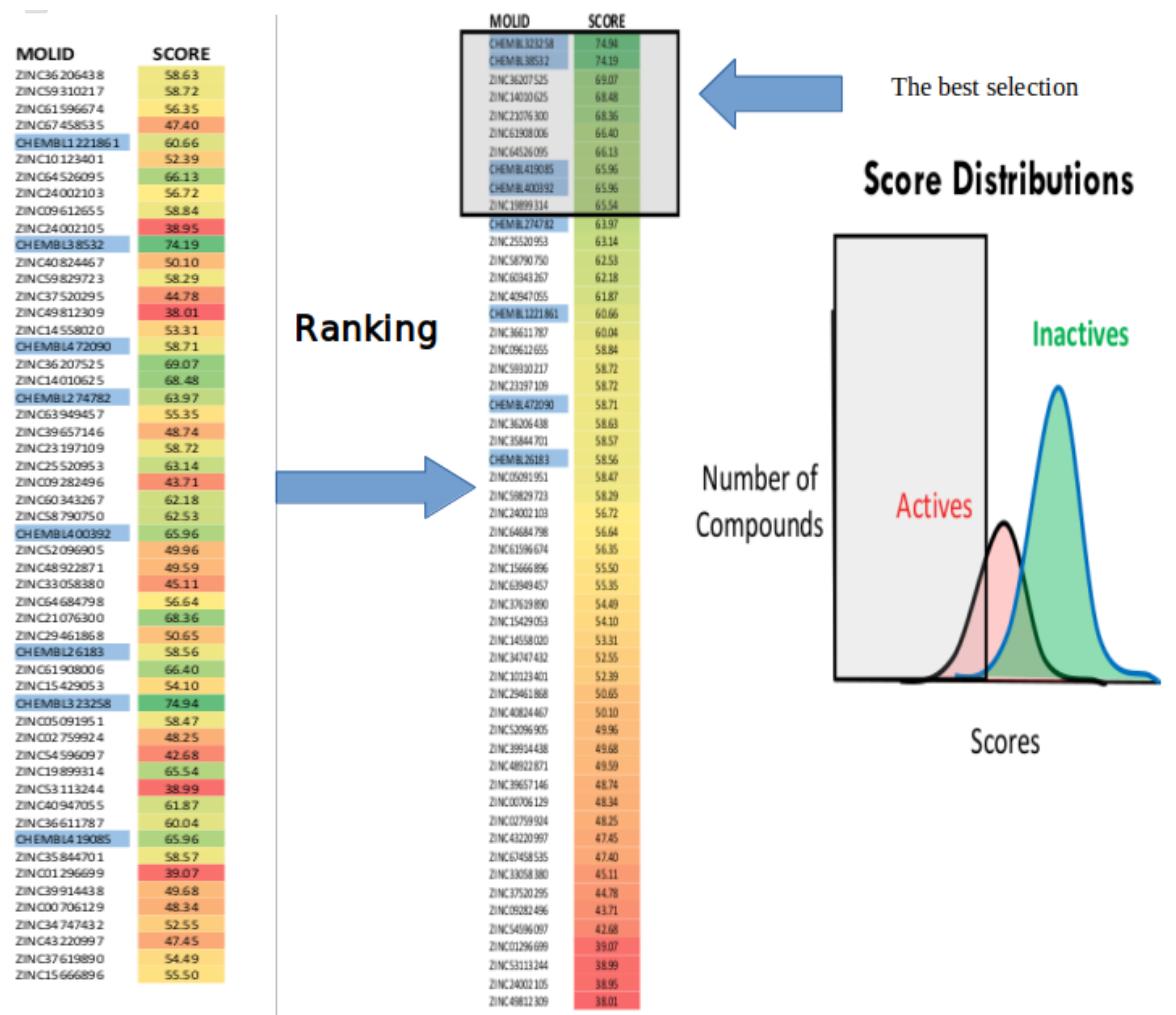


FIGURE 1.10 – Le processus de Classement

1.4.8 Sélection de hits

La sélection de hits³ est la dernière étape du virtual screening. Elle consiste à choisir, pour une investigation plus approfondie, les molécules les mieux classées, prédisées comme ayant la plus forte affinité de liaison ou l'activité biologique souhaitée.[23] Ce processus de sélection implique une évaluation minutieuse de divers facteurs, allant au-delà du simple score de classement. Les scientifiques tiennent en compte de données supplémentaires telles que les propriétés physico-chimiques, la toxicité potentielle et la facilité de synthèse afin de garantir la sélection de candidats prometteurs pour des expérimentations ultérieures *in vitro* ou *in vivo*.[23] La figure 1.11 illustre les étapes du processus de sélection de hits. En fin de compte, l'objectif de la sélection de hits est d'identifier un petit nombre de composés de haute qualité ayant le plus grand potentiel de réussite dans le pipeline de découverte de médicaments.

3. Hits : Une molécule identifiée comme ayant une potentialité prometteuse pour interagir avec une cible biologique spécifique.

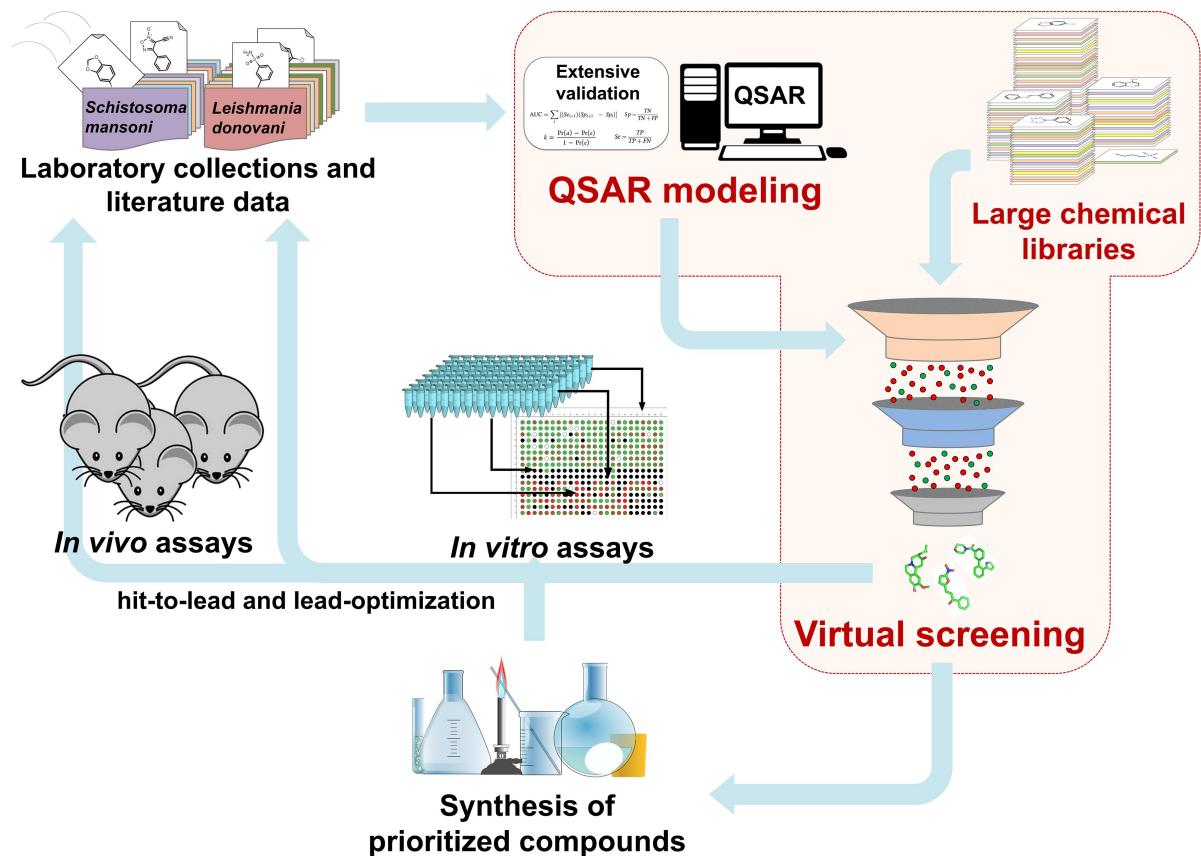


FIGURE 1.11 – La sélection de hits [23]

1.5 Virtual screening basé sur la structure (SBVS)

Dans cette section, nous allons découvrir le deuxième type de virtual screening. L’objectif est d’avoir une vue globale sur SBVS et de pouvoir comprendre la comparaison entre LBVS et SBVS.

1.5.1 Définition

Les méthodes de virtual screening basé sur la structure, également appelées virtual screening basé sur la cible (TBVS) [6], utilisent la structure tridimensionnelle de la protéine cible et certaines informations sur le site de liaison pour arrimer les molécules ligands au récepteur et évaluer leur affinité de liaison par des techniques computationnelles ou théoriques. Dans cette stratégie, les composés sont sélectionnés dans une base de données et classés en fonction de leur affinité pour le site du récepteur, cette méthode basée sur le docking moléculaire.[6] Le docking moléculaire utilise une fonction de score pour estimer l’affinité de liaison. La figure 1.3 montre que cette méthode est appliquée lorsque la structure 3D de la cible est connue.

1.5.2 Les étapes du SBVS

- Sélection du récepteur :** C’est la première étape du virtual screening, où une cible appropriée est choisie pour le processus de découverte de médicaments. Le récepteur

doit être pertinent à la maladie d'intérêt, avoir une structure connue ou prédictive, et posséder un site de liaison capable d'accueillir des ligands potentiels.[24]

2. **Préparation de la bibliothèque** : C'est la deuxième étape du virtual screening, où une collection de candidats médicaments potentiels est sélectionnée ou générée pour le docking et le scoring. La bibliothèque peut provenir de bases de données existantes de structures chimiques, ou de méthodes de chimie combinatoire qui produisent de nouveaux composés.[24]
3. **Docking et scoring** : C'est la troisième étape du virtual screening, où les poses des ligands dans le site de liaison du récepteur sont prédites et évaluées. Le docking vise à trouver l'orientation et la conformation optimales du ligand qui maximisent l'énergie d'interaction avec le récepteur.[24]
4. **Classement et filtrage** : C'est la quatrième étape du virtual screening. Après le docking et le scoring, les complexes ligand-protéine générés sont filtrés en fonction de divers critères tels que l'affinité de liaison, le modèle d'interaction, la complémentarité de forme et les caractéristiques pharmacophores. Cette étape permet de prioriser les candidats les plus prometteurs pour une analyse ultérieure, en sélectionnant les meilleurs candidats médicaments parmi les ligands arrimés et scorés.[24]
5. **Évaluations et optimisation** : C'est la dernière étape du virtual screening, où les candidats médicaments sélectionnés sont testés et affinés dans des environnements expérimentaux et computationnels. Les évaluations consistent à mesurer l'affinité de liaison, la sélectivité et l'activité des composés à l'aide d'essais biochimiques et biophysiques.[24]

1.6 Molecular docking

1.6.1 Définition

Le processus molecular docking permet de prédire comment une molécule (ligand) pourrait se lier à une autre molécule (récepteur) en trouvant la meilleure position possible (orientation) pour le ligand dans le site de liaison du récepteur(voir figure 1.12) et en minimisant l'énergie d'interaction entre le récepteur et le ligand.[1] Les résultats de ce processus sont utilisés dans le virtual screening basé sur la structure (SBVS) pour recommander les composés les plus actifs parmi une large bibliothèque.[2]

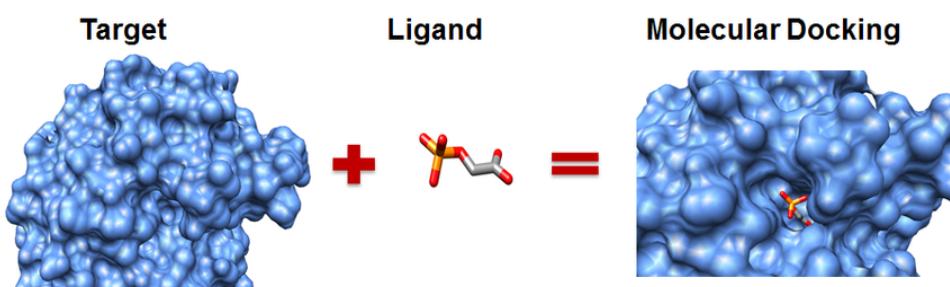


FIGURE 1.12 – Le processus de Moléculaire docking [25]

1.6.2 Relation entre moléculaire docking et virtual screening

Le processus de découverte de médicaments est l'un des plus grands défis de l'industrie pharmaceutique, car il nécessite beaucoup de temps et d'argent pour traverser toutes les phases de développement d'un nouveau médicament. Pour minimiser les coûts et les délais, des méthodes computationnelles telles que le docking moléculaire et le virtual screening sont utilisées.

Moléculaire docking est d'abord utilisé pour identifier la structure 3D du complexe final. Il prédit le meilleur site de liaison du ligand dans le composé récepteur en explorant différentes orientations et en calculant le score de cette interaction à l'aide d'une fonction de score (SF) pour former un complexe stable.[1] Les méthodes docking et de scoring varient en fonction de leurs algorithmes, de leur précision, de leur vitesse et de leur applicabilité. Glide, Autodock, DOCK et PLANTS font partie des programmes de molecular docking les plus courants.[6]

Ensuite, virtual screening basé sur la structure utilise les résultats de molecular docking pour cibler la bibliothèque de composés et sélectionner les mieux classés. La figure 1.13 illustre le pipeline du virtual screening basé sur la structure.

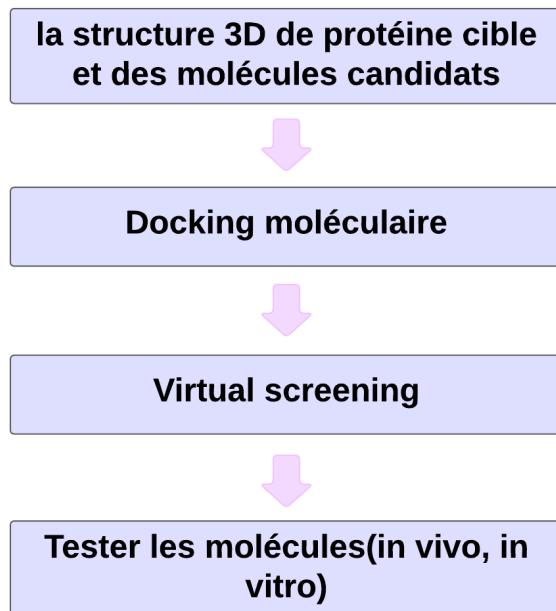


FIGURE 1.13 – le pipeline du virtual screening basé sur la structure

1.7 Virtual screening à l'ère du Big-Data et le rôle du HPC

Le virtual screening s'appuie sur des bases de données volumineuses contenant jusqu'à 166 milliards de molécules.[26] La transformation et le codage de ces grandes bases de données (molécules) impliquent des traitements complexes et des calculs mathématiques sophistiqués afin d'analyser les propriétés chimiques et biologiques des molécules. Au fil du temps, les ensembles de données moléculaires continuent de croître de manière constante, ce qui peut engendrer des temps d'exécution longs.[27] Ce défi met en évidence

un problème de Big_Data.[26] Pour résoudre ce problème, l'utilisation de l'intelligence artificielle et de calcul haute performance (HPC) s'avère cruciale. Ces technologies permettent de traiter efficacement de grandes quantités de données et d'extraire rapidement les molécules les plus prometteuses.[28] En combinant le Virtual Screening avec l'IA et le HPC, il est possible de capturer et d'analyser des dataset contenant des milliards de molécules, ce qui permet de minimiser le temps de développement de nouveaux médicaments et de lutter plus efficacement contre les maladies dangereuses.[28]

1.8 Analyse les méthodes existantes pour le virtual screening

1.8.1 Comparaison entre le LBVS et le SBVS

Le virtual screening basé sur les ligands (LBVS) et le virtual screening basé sur la structure (SBVS) sont deux méthodes computationnelles employées dans la découverte de médicaments pour identifier des candidats médicaments potentiels. Bien que les deux approches visent à prédire l'affinité de liaison des molécules à une protéine cible, elles reposent sur des principes différents et présentent des avantages et des limitations distincts.

Comparaison entre le LBVS et le SBVS :

- **Principe :**

- LBVS (Virtual screening basé sur les ligands) : Utilise les propriétés connues des ligands actifs pour le criblage.
- SBVS (Virtual screening basé sur la structure) : Repose sur la structure de la protéine cible et le docking moléculaire pour le criblage.

- **Exigences en matière de données :**

- LBVS : Nécessite des ligands actifs connus, mais pas la structure de la protéine.
- SBVS : Nécessite la structure 3D de la protéine cible.

- **Applicabilité :**

- LBVS : Convient lorsque la structure de la protéine est indisponible ou difficile à obtenir.
- SBVS : Convient aux cibles dont la structure est connue.

- **Rapidité et ressources :**

- LBVS : Relativement plus rapide, moins gourmand en calcul.
- SBVS : Plus gourmand en calcul, nécessite des ressources importantes.

- **Précision :**

- LBVS : La précision peut varier en fonction de la qualité des données sur les ligands.
- SBVS : Généralement plus précis avec des structures cibles connues.

- **Points forts :**

- LBVS : Criblage rapide même sans données sur la structure de la protéine.
- SBVS : Offre une grande précision avec des interactions protéine-ligand détaillées.

- **Limites principales :**

- LBVS : Dépend de la disponibilité et de la qualité des données sur les ligands actifs.
- SBVS : Nécessite la structure de la protéine cible. Gourmand en calcul.

1.8.2 Principaux avantages du VS

- Réduction des coûts et du temps : Le VS réduit considérablement le nombre de composés nécessitant des tests expérimentaux, ce qui permet de réaliser des économies importantes en termes de coûts et de temps par rapport au HTS.
- Efficacité accrue : Le VS permet d'explorer un espace chimique beaucoup plus large, ce qui peut conduire à l'identification de molécules candidates plus diverses et plus innovantes, présentant de meilleures propriétés pharmacologiques.

1.8.3 Limites du VS

- Précision des fonctions de score : La qualité de l'affinités de liaison prédictes dépend fortement des fonctions de score employées, qui peuvent ne pas toujours refléter parfaitement la complexité de forces impliquée dans les interactions ligand-protéine.
- Flexibilité des protéines cibles : Les structures des protéines peuvent être flexibles, et le VS peut ne pas toujours en tenir compte de manière adéquate, ce qui peut conduire à l'exclusion de ligands prometteurs qui pourraient se lier à des conformations alternatives de la cible.
- Disponibilité des structures cibles : Le SBVS nécessite une structure 3D de haute qualité de la protéine cible, qui peut ne pas toujours être facilement disponible ou précise.

1.9 Conclusion

En conclusion, le virtual screening (VS) s'est imposé comme un outil puissant et polyvalent dans l'arsenal de la découverte de médicaments, offrant des avantages significatifs par rapport aux méthodes traditionnelles de high throughput screening (HTS). Cependant, la fiabilité des molécules recommandées par les algorithmes de virtual screening reste un défi majeur. Les méthodes de calcul de similarité traditionnelles peuvent ne pas capturer toutes les propriétés pertinentes des molécules, tandis que les approches basées sur des codes ne parviennent pas toujours à saisir l'ensemble des informations chimiques et structurales. C'est pourquoi les scientifiques se dirigent vers le l'apprentissage profond pour améliorer la performance du virtual screening. Le chapitre suivant explorera en profondeur le fonctionnement du l'apprentissage profond et présentera diverses architectures utilisant cette approche pour optimiser le virtual screening.

CHAPITRE 2

APPRENTISSAGE PROFOND(DEEP LEARNING)

2.1 Introduction

Le virtual screening (VS) permet l'évaluation *in silico*¹ de vastes base de données des molécules candidates afin d'identifier celles présentant un potentiel thérapeutique. Traditionnellement, le virtual screening reposait sur des simulations de docking moléculaire et des méthodes basées sur les ligands. Au cours des dernières années, l'apprentissage profond a suscité un grand intérêt en raison de sa capacité à résoudre des problèmes complexes qui étaient auparavant considérés comme NP-hard. En exploitant la puissance des réseaux neuronaux profond, il a permis des avancées dans divers domaines tels que la vision par ordinateur, le traitement du langage naturel et la reconnaissance vocale. L'apprentissage profond a révolutionné ce domaine (VS) en offrant des capacités inégalées pour analyser des données biologiques complexes.

Ce chapitre explore les différents types d'apprentissage profond, ainsi que les diverses applications de l'apprentissage profond dans le virtual screening. À la fin de ce chapitre, vous aurez une compréhension complète de la manière dont l'apprentissage profond améliore le processus de virtual screening.

2.2 Apprentissage Profond(Deep learning)

2.2.1 Définition

L'apprentissage profond est une branche de l'apprentissage automatique qui se concentre sur les algorithmes et les modèles d'intelligence artificielle (IA) inspirés par la structure et le fonctionnement du cerveau humain.[29] Il consiste à former des réseaux neuronaux avec plusieurs couches de nœuds interconnectés pour reconnaître des motifs et effectuer des prédictions.[30] Il existe plusieurs type de l'apprentissage profond comme, apprentissage profond entièrement connecté (FCNN), réseaux de neurones à convolution (CNN), réseaux de neurones récurrents (RNN), réseaux de neurones en graphes (GNN).

La figure 2.1 illustre que, contrairement aux réseaux de neurones simples qui possèdent une architecture moins complexe avec un nombre réduit de couches cachées, les réseaux

1. Silico : basée sur l'ordinateur

de neurones profonds disposent de multiples couches cachées.[30] Ces couches supplémentaires permettent au réseau d'apprendre des caractéristiques plus complexes et abstraites, ce qui est essentiel pour accomplir des tâches de traitement de données plus sophistiquées telles que la reconnaissance d'images complexes, la traduction automatique de langues et la génération de texte. En d'autres termes, les couches supplémentaires agissent comme des filtres successifs, analysant les données à différents niveaux de granularité. Ce processus permet au réseau d'apprendre des représentations plus riches et plus profondes des données, lui offrant ainsi une meilleure compréhension du contexte et des relations entre les différents éléments.[29]

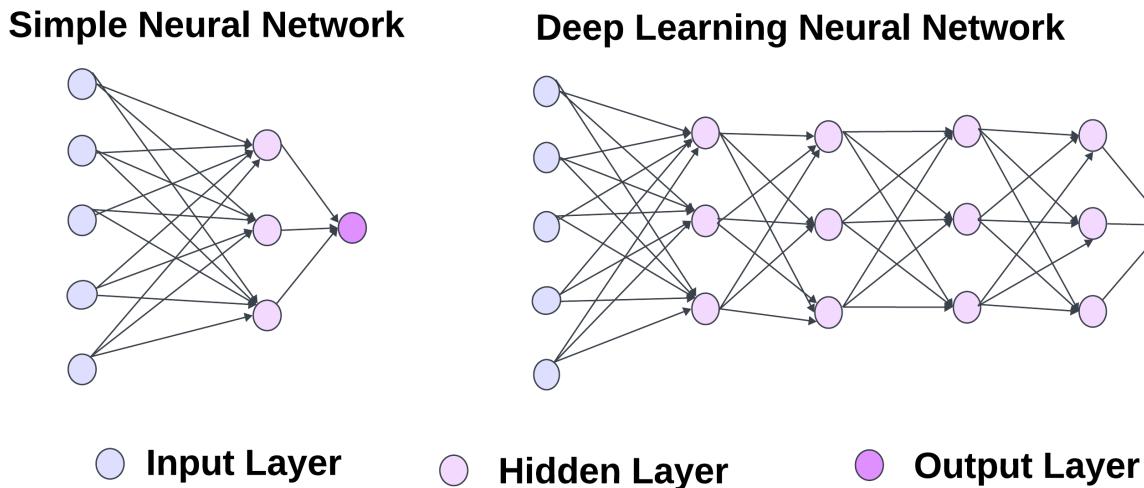


FIGURE 2.1 – Apprentissage Profond et les réseaux neuronaux artificiels

2.2.2 Les réseaux neuronaux artificiels (RNA)

Les réseaux neuronaux artificiels (RNA) sont des modèles informatiques inspirés de la structure et du fonctionnement du cerveau humain.[31] Ils sont constitués de nœuds interconnectés, également appelés neurones artificiels, organisés en couches.[32] Les éléments clés d'un réseau de neurones :

- **paramètres** : Les paramètres sont des valeurs apprises par le modèle en fonction des données (w_1, w_2, \dots, w_n) et (b_1, b_2, \dots, b_n).[33]
- **Hyperparamètres** : Les hyperparamètres sont des valeurs fournies pour ajuster le modèle et ne peuvent pas être appris à partir des données (x_1, x_2, \dots, x_n).[33]

2.2.3 La configuration typique d'un RNA

- **La couche d'entrée(input layer)** : Elle reçoit des données provenant de sources externes et les transmet aux couches cachées pour traitement. Cette couche effectue des calculs initiaux sur les données d'entrée.[33]
- **Les couches cachées(hidden layer)** : Ces couches intermédiaires sont cruciales dans le processus. Elles effectuent des calculs plus complexes en utilisant une combinaison de poids associés à chaque noeud. Cependant le rôle clé des couches cachées

résidé dans l'application des fonctions d'activation. Chaque neurone dans ces couches applique une fonction d'activation à la sortie des calculs.[32]

- **La couche de sortie(output layer)** : Elle génère le résultat final ou la prédiction en fonction des informations traitées dans les couches cachées. Encore une fois, la fonction d'activation appliquée à la sortie de cette couche est essentielle pour déterminer la réponse finale.[32]

2.2.4 perceptron :

Un perceptron ou un neurone artificiel, est une unité d'un réseau de neurones qui effectue des calculs (sumation et activation) afin de détecter des caractéristiques ou des tendances dans les données d'entrée(figure 2.2).[33]

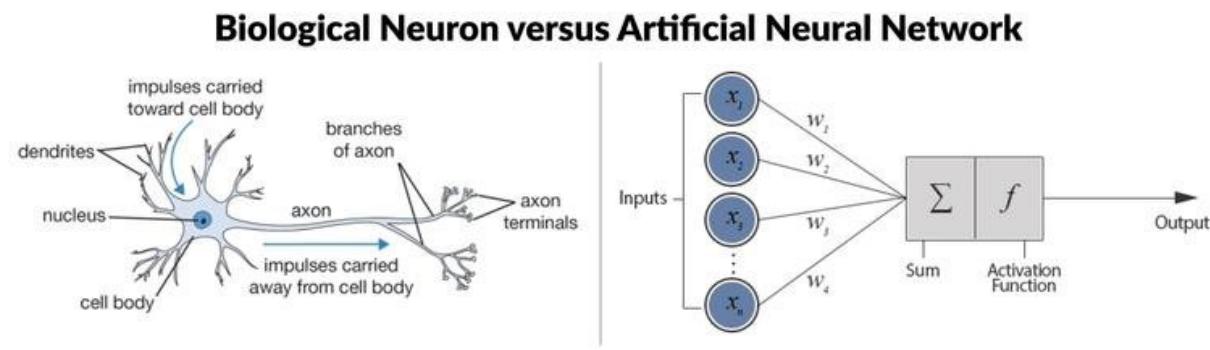


FIGURE 2.2 – Inspiration biologique des réseaux de neurones [34]

2.2.5 Les fonctions d'activation

Les fonctions d'activation sont des opérations permettant à l'information de se propager aux couches futures. Elles déterminent si un neurone doit être activé ou non, en fonction du potentiel d'activation atteint [33]. Il existe plusieurs types de fonctions d'activation (voir la figure 2.3), dont celles citées :

- ReLU (Rectified Linear Unit) :

$$f(x) = \max(0, x) \quad (2.1)$$

- Sigmoid (logistic)

$$f(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

- TanH

$$f(x) = \tanh(x) \quad (2.3)$$

- Softmax

$$\sigma(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^K \exp(x_j)} \quad (2.4)$$

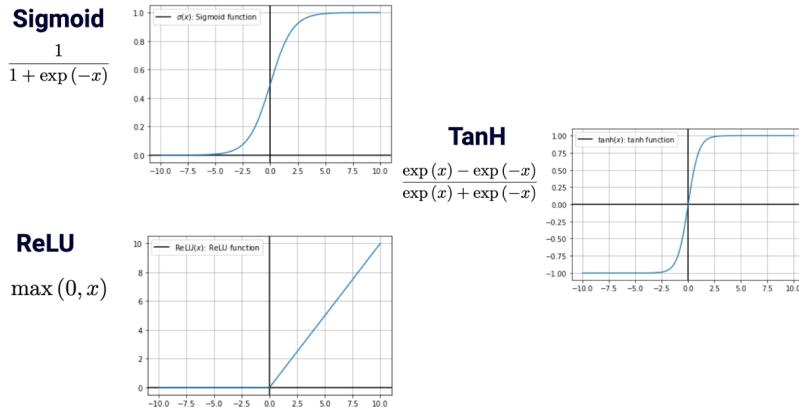


FIGURE 2.3 – Les fonctions d’activation

2.2.6 Fonctionnement de l’apprentissage profond

Le fonctionnement d’un réseau de neurones artificiels (RNA) peut être assimilé à un processus d’apprentissage et de décision basé sur des données.[33] Chaque neurone dans le réseau travaille en parallèle pour apprendre un aspect spécifique des données d’entrée.

- **Calcul de la sortie d’un neurone :** Chaque neurone prend en entrée un ensemble de valeurs x_i , les multiplie par des poids w_i , les somme et ajoute un biais b pour obtenir une valeur de sortie brute. Cette valeur est ensuite transformée par une fonction d’activation f pour déterminer la sortie du neurone. Mathématiquement, cela peut être représenté par la formule (2.5) :

$$\text{Sortie} = f \left(\sum_{i=0}^n x_i \cdot w_i + b \right) \quad (2.5)$$

- **Propagation des informations :** La sortie de chaque neurone devient l’entrée du suivant, créant ainsi un flux d’informations à travers les couches du réseau.
- **Apprentissage par rétropropagation :** Lorsque les données atteignent la couche de sortie, le réseau évalue les résultats et calcule l’erreur en utilisant une fonction de coût. Cette erreur est ensuite propagée en arrière à travers le réseau (rétropropagation) pour ajuster les poids w_i et les biais b , dans le but d’améliorer la performance du modèle.[32]
- **Amélioration progressive :** Ce processus de rétropropagation et de propagation avant constitue le cœur de l’apprentissage d’un réseau de neurones, lui permettant de s’adapter et d’optimiser sa capacité à résoudre des tâches complexes.

La figure suivante 2.4 montre un réseau de neurones avec des entrées x_1, x_2, \dots, x_n et leurs poids associés w_1, w_2, \dots, w_n , et un biais b . La somme pondérée dans le neurone est calculée comme suit :

$$Z = \sum_{i=1}^n w_i \cdot x_i + b$$

La sortie du neurone est ensuite obtenue en appliquant une fonction d'activation f à la somme pondérée :

$$f(Z) = \frac{1}{1 + e^{-Z}}$$

où

$f(Z)$ est la sortie du neurone après application de la fonction d'activation sigmoid.

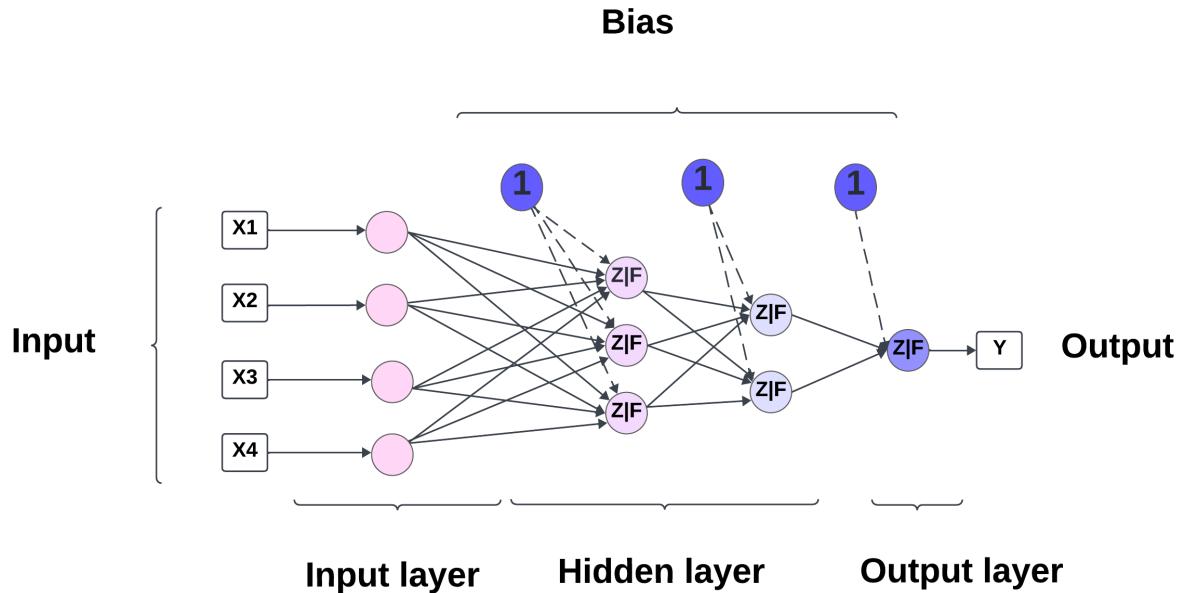


FIGURE 2.4 – Les réseaux neuronaux artificiels (RNA)

2.2.7 L'optimisation des modèles d'apprentissage profond

L'optimisation est au cœur de l'entraînement des réseaux de neurones profonds. Son objectif est de trouver les valeurs optimales des paramètres du réseau pour minimiser la fonction de coût et maximiser la précision du modèle.[33] Le processus d'optimisation débute par l'initialisation des poids selon différentes stratégies. Ensuite, à chaque itération (appelée époque), les poids sont mis à jour en fonction d'une équation spécifique.[33] Parmi les fonctions d'optimisations les plus courantes, on peut citer

- **Descente du gradient** : Cet algorithme calcule le gradient de la fonction de perte par rapport aux paramètres du modèle et les ajuste dans la direction opposée au gradient.[33]

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta_t} L(\theta_t) \quad (2.6)$$

- θ_t : Les paramètres du modèle à l'itération t .
- α : Le taux d'apprentissage(learning rate), qui contrôle la taille des pas que nous prenons lors de la mise à jour des paramètres.
- $\nabla_{\theta_t} L(\theta_t)$: Le gradient de la fonction de perte $L(\theta_t)$ par rapport aux paramètres θ_t . Il indique la direction dans laquelle la fonction de perte augmente le plus rapidement.

- **Adam** : Cet algorithme est une variante de la descente du gradient qui utilise des estimations adaptatives du taux d'apprentissage pour chaque paramètre.[33]

$$\min_x f(x) + g(x) \quad (2.7)$$

- θ_t : Les paramètres du modèle à l'itération t .
- α : Le taux d'apprentissage.
- $f(x)$: La fonction que nous cherchons à minimiser.
- $g(x)$: Le gradient de la fonction $f(x)$ par rapport aux paramètres.
- $\min_x f(x) + g(x)$: L'objectif d'optimisation, où nous cherchons à minimiser la somme de la fonction et de son gradient.
- **RMSProp** : Cet algorithme est similaire à Adam mais utilise une moyenne mobile des gradients pour améliorer la stabilité.[33]

$$\begin{aligned} v_t &= \beta v_{t-1} + (1 - \beta) g_t^2 \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{v_t + \epsilon}} g_t \end{aligned} \quad (2.8)$$

- θ_t : Les paramètres du modèle à l'itération t .
- η : Le taux d'apprentissage.
- v_t : La moyenne mobile des carrés des gradients précédents.
- β : Le facteur d'oubli (entre 0 et 1) pour la moyenne mobile.
- g_t : Le gradient de la fonction de perte par rapport aux paramètres.
- ϵ : Un petit terme d'ajustement pour la stabilité numérique.

2.2.8 L'évaluation des modèles d'apprentissage profond

L'évaluation des modèles d'apprentissage profond est essentielle pour comprendre leur performance et leur efficacité. Elle consiste à mesurer la capacité du modèle à accomplir la tâche pour laquelle il a été conçu, en utilisant des données et des métriques spécifiques [2].

Les métriques d'évaluation des modèles d'apprentissage profond

- **Matrice de confusion** : La matrice de confusion est un outil qui permet de visualiser les performances d'un modèle de classification. Elle compare les prédictions du modèle avec les vraies étiquettes (ou classes) des données.

		Actual values	
		True	False
Predicted values	True	True_Positive	False_Positive
	False	False_Negative	True_Negative

FIGURE 2.5 – Matrice de confusion

La figure 2.5 illustre la matrice de confusion et ces quatre éléments :

Vrai négatif : Indique combien de valeurs négatives sont prédites comme négatives uniquement par le modèle.[2]

Faux positif : Indique combien de valeurs négatives sont prédites comme valeurs positives par le modèle.

Faux négatif : Indique combien de valeurs positives sont prédites comme valeurs négatives par le modèle.[2]

Vrai positif : Indique combien de valeurs positives sont prédites comme positives uniquement par le modèle.[2]

- **Précision** : La précision mesure la proportion d'échantillons correctement classés comme positifs parmi tous les échantillons prédis comme positifs.[2][35]

$$\text{Précision} = \frac{TP}{TP + FP} \quad (2.9)$$

- **Rappel (Recall)** : Le rappel mesure la proportion d'échantillons correctement classés comme positifs parmi tous les échantillons réellement positifs.[35]

$$\text{Rappel} = \frac{TP}{TP + FN} \quad (2.10)$$

- **Exactitude (Accuracy)** : L'exactitude mesure la proportion d'échantillons correctement classés (positifs et négatifs) parmi tous les échantillons.[2][35]

$$\text{Exactitude} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.11)$$

2.3 Les types de deep learning

2.3.1 Apprentissage profond entièrement connecté (FCNN)

Un réseau de neurones profonds entièrement connectés (FCNN), aussi appelé réseau neuronal perceptron multicouche (MLP), est un type de réseau neuronal profond où chaque neurone d'une couche est connecté à tous les neurones de la couche suivante.

Cette connectivité complète permet au réseau d'apprendre des relations complexes entre les caractéristiques des données.[21] Les FCNNs apprennent en ajustant les poids des connexions entre les neurones. Cet ajustement est effectué à l'aide d'un algorithme d'apprentissage automatique, tel que la rétropropagation.

2.3.2 Réseaux de neurones à convolution (CNN)

Un réseau de neurones convolutifs (CNN) est un type spécifique de réseau de neurones artificiels conçu pour traiter des données en grille, en particulier des images.[29] Il utilise une opération mathématique appelée convolution pour extraire des caractéristiques pertinentes des images.[33] Grâce à leur capacité à extraire des caractéristiques pertinentes des images, les CNN sont largement utilisés en vision par ordinateur pour des tâches telles que la reconnaissance d'objets, la classification d'images, la détection de visages et la segmentation d'images.[33][31]

les points clés du fonctionnement d'un CNN

- **Noyau de convolution (Kernel)** : C'est une petite matrice qui glisse sur l'image. Elle permet d'extraire des caractéristiques spécifiques de l'image, comme les contours, les textures ou les couleurs.[33]
- **L'opération de convolution** : Elle calcule le produit scalaire élément par élément entre le filtre et une zone de l'image. La somme de ces produits est affectée à un pixel de l'image de sortie la figure 2.6 illustre comment cette opération est effectuée. En répétant ce processus pour toutes les zones possibles de l'image d'entrée, on obtient l'image de sortie.[33] [29]

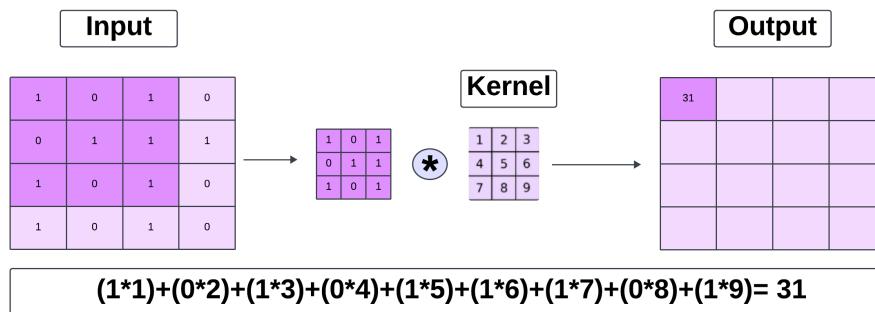


FIGURE 2.6 – L'opération de convolution

L'opération de pooling

L'opération de pooling permet de réduire la taille de kernel tout en conservant les informations les plus importantes. Cela se fait en appliquant une fonction d'agrégation (comme la moyenne, le maximum) sur des zones de la grille.[33] [29]

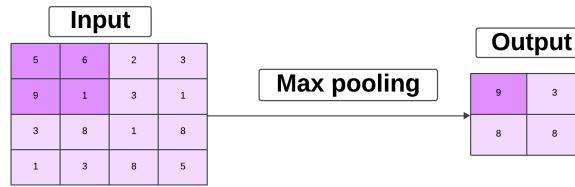


FIGURE 2.7 – L'opération de pooling

L'opération de flatting

L'opération de flattening, est une technique utilisée pour transformer les sorties de CNN en vecteur linéaires.

2.3.3 Réseaux de neurones récurrents (RNN)

Les réseaux neuronaux récurrents (RNN) constituent une catégorie de réseaux neuronaux artificiels qui se distinguent par leur capacité à gérer des séquences de données en série, à l'opposé des réseaux neuronaux traditionnels qui analysent des éléments de données isolés les uns des autres.[30]

La figure 2.8 représente un modèle de Réseau Neuronal Récurrent (RNN) possède une mémoire interne (A) qui lui permet de stocker des informations sur les éléments précédents de la séquence lors du traitement de l'élément actuel. Cette capacité est cruciale pour comprendre le contexte et les relations entre les éléments d'une séquence.[30]

Les connexions au sein d'un RNN forment des boucles, ce qui permet à l'information d'être stockée et propagée à travers le réseau. Les RNN sont utilisés dans de nombreuses applications impliquant des séquences de données, telles que :

Traitement du langage naturel (NLP) : reconnaissance vocale, traduction automatique, génération de texte.

Analyse de séries temporelles : prévision de la demande, analyse des marchés financiers.

Bio-informatique : analyse de séquences ADN et protéines.

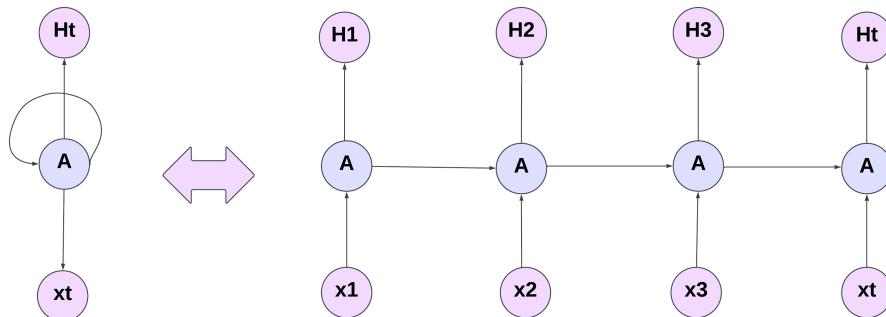


FIGURE 2.8 – Réseaux de neurones récurrents (RNN)

2.3.4 Réseaux de neurones en graphes (GNN)

Les réseaux de neurones graphiques (GNN, Graph Neural Networks) sont un type de réseau neuronal profond conçu pour traiter des données structurées sous forme de graphes, la figure 2.9 représente l'architacteur d'un réseaux de neurones sur des garaphes. Les GNN

exploitent les relations entre les nœuds et les caractéristiques des nœuds pour apprendre des représentations précises des données graphiques.[36]

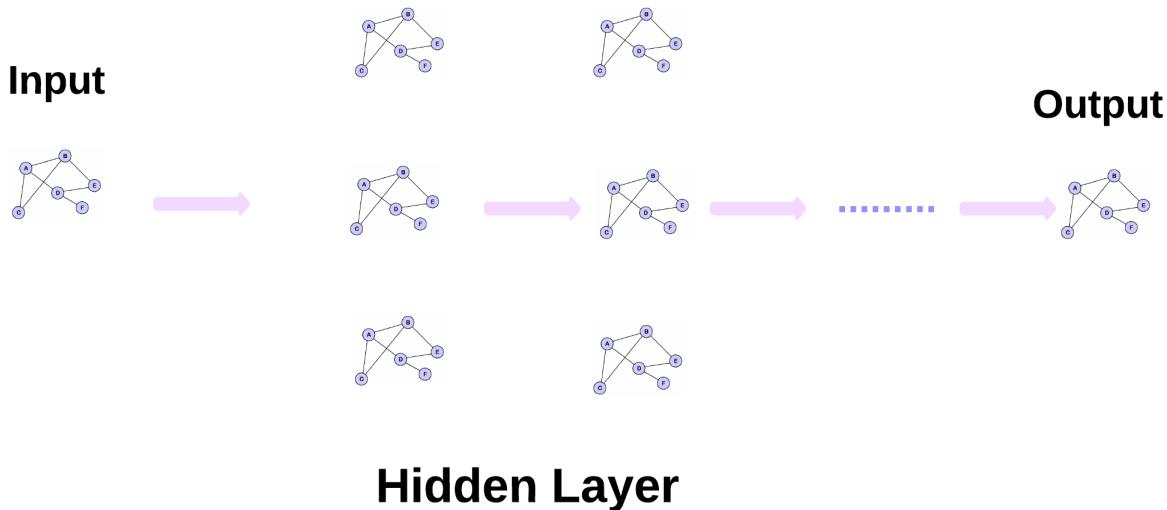


FIGURE 2.9 – L’architacteur de GNN

Fonctionnement des Réseaux de Neurones sur Graphes (GNN)

Les GNN exploitent des vecteurs caractéristiques, aussi appelés embeddings, pour encoder les informations d’un graphe, incluant sa structure et ses relations. Deux types d’embeddings interviennent :

- **Embeddings de nœuds** : Ils capturent les propriétés individuelles de chaque nœud du graphe.[37]
- **Embeddings d’arêtes** : Ils modélisent les caractéristiques des relations reliant les nœuds entre eux.[37]

En combinant ces embeddings, les GNN construisent une représentation numérique du graphe exploitable par des algorithmes de Machine Learning. Cette approche permet d’analyser et d’exploiter les structures complexes des graphes dans divers domaines, comme la chimie, la biologie, les réseaux sociaux et bien d’autres.[38] Les GNN s’attaquent à cette complexité grâce à la transmission de messages. Cette technique permet aux algorithmes d’apprentissage automatique de naviguer et d’explorer les structures des graphes. En effet, chaque noeud reçoit les informations de ses voisins, les intègre et les utilise pour générer des prédictions précises.[38]

2.4 Sous-apprentissage et sur-apprentissage

Obtenir un modèle performant dans le domaine du deep learning n’est pas trivial. Même un modèle présentant une grande précision sur un jeu de données spécifique ne garantit pas des performances optimales dans des situations futures, en raison du risque potentiel de surapprentissage ou de sous-apprentissage du modèle.[39]

Le bon apprentissage

Le bon ajustement d'un modèle fait référence à sa capacité à généraliser correctement à partir des données d'entraînement à de nouvelles données, capturer les caractéristiques pertinents des données d'entraînement, ce qui le rend capable de faire des prédictions précises sur de nouvelles données.[40] [30]

Sous apprentissage

Le sous apprentissage est un facteur clé limitant la capacité des modèles d'apprentissage automatique à produire des résultats précis. Il survient lorsque le modèle d'apprentissage automatique n'est pas suffisamment entraîné sur des données d'apprentissage. En conséquence , le modèle ne parvient pas à saisir correctement la relation entre les données d'entrée et les données de sortie. Cela se traduit par des prédictions imprécises et un modèle qui ne peut pas généraliser efficacement à de nouvelles données.[39] Pour éviter le sous-apprentissage, il est souvent nécessaire d'augmenter la complexité du modèle ou d'améliorer la qualité et la quantité des données d'entraînement.

Sur apprentissage

Le sur apprentissage l'un des principaux facteurs limitant la performance d'un modèle d'apprentissage automatique. Il se produit lorsque le modèle apprend les données d'entraînement avec trop de précision, au point de ne plus être capable de généraliser à de nouvelles données. En d'autres termes, le modèle devient incapable de produire des résultats raisonnables sur des nouvelles données. Un modèle sur-apprentissage est un modèle complexe.[41]

Comment éviter le surapprentissage et le sous-apprentissage ?

L'une des techniques couramment employée pour avoir un modèle avec bon apprentissage est la validation croisée.[41] Cette approche consiste à diviser les données d'entraînement en plusieurs sous-groupes distincts. Le modèle est ensuite entraîné et évalué à l'aide de chaque sous-groupe, permettant ainsi de calculer une performance moyenne estimée sur l'ensemble des données d'entraînement la figure 2.10 schématise la méthode de la validation croisée.[42]



FIGURE 2.10 – la méthode de la validation croisée

2.5 Virtual screening et deep learning

La méthode de virtual screening traite des données massives avec plusieurs opérations de calculs ce qui implique un temps d'exécution très élevé. Les calculs sont utilisés pour évaluer les interactions ou pour mesurer la similarité entre les molécules. Il existe plusieurs métriques de calcul, le choix d'une métrique est un autre défi. L'utilisation du deep learning accélère le processus de virtual screening et , il permet aussi de combiner plusieurs métriques pour le calcul de similarité, et par conséquent plus d'efficacité, de fiabilité et de meilleure performance.[43]

2.5.1 Les architecteur de deep learning virtual screening

Les chercheurs ont proposé plusieurs architectures pour améliorer le virtual screening par l'utilisation des modèles de deep learning pour améliorer les performances. Parmi les architectures les plus utilisées on peut citer :

Architecture of Siamese RNN similarity model[22]

Mohammed Khaldoon Altalib et al, ont proposé une architecture basée sur les RNN (voir figure 2.5.1) qui permet de déduire une valeur qui représente les propriétés biologiques des molécules avec les couches. Ensuite, LSTMIL calcule deux distances : la distance de Manhattan et la distance exponentielle de Manhattan et fusionne les deux distances avec une couche de fusion suivie de deux couches denses avec (512, 256) neurones respectivement et une couche de sortie.

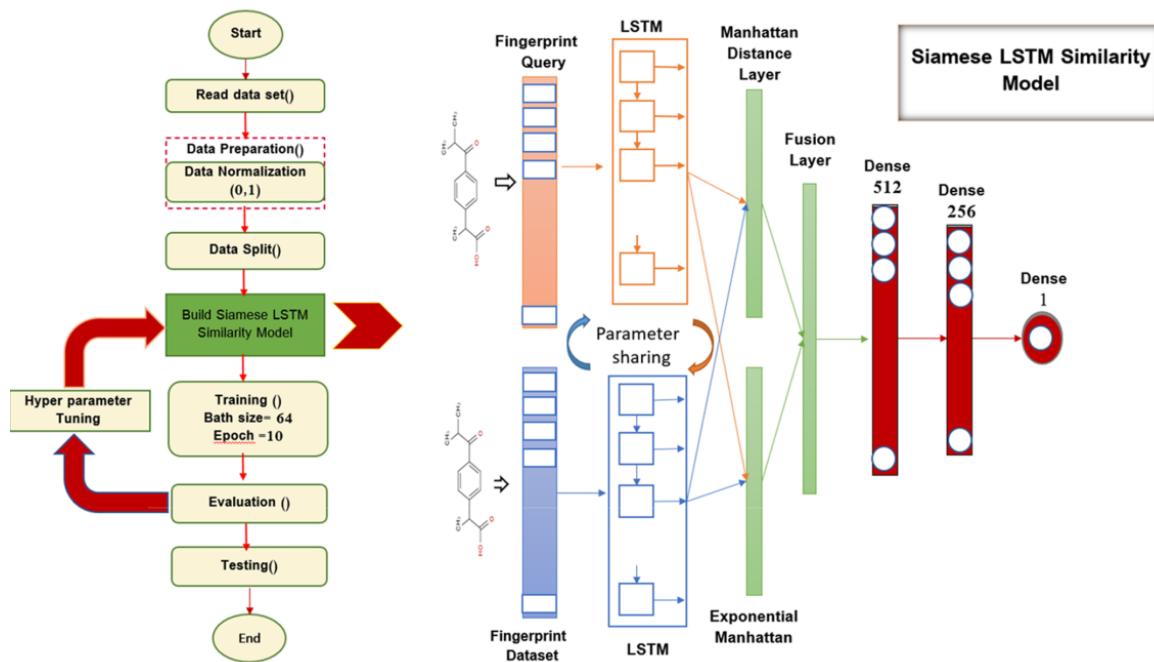


FIGURE 2.11 – Architecture of Siamese RNN similarity model [22]

Les critiques

- Complexité de mise en œuvre :** Par rapport à des modèles plus simples comme la définition d'une fonction qui calcule une valeur représentative pour les smiles, la

mise en place d'un réseau Siamese LSTM peut s'avérer plus complexe. En effet, elle requiert une compréhension approfondie des hyperparamètres et des techniques de régularisation.

- **Temps d'entraînement plus long :** En raison de sa nature d'apprentissage profond, l'entraînement d'un Siamese LSTM peut prendre plus de temps, surtout si le modèle est profond et que le jeu de données est volumineux. Il est important de souligner que cette étape de calcul d'une valeur représentative des smiles n'est qu'une étape parmi d'autres dans le processus global de virtual screening.
- **Risque de surapprentissage :** Si le modèle n'est pas correctement régularisé, il peut surapprendre les caractéristiques spécifiques du jeu de données d'entraînement et ne pas généraliser correctement aux nouvelles données. Cette valeur détermine les propriétés chimiques des molécules. Or, une erreur dans cette valeur peut entraîner des résultats non fiables. De plus, cette méthode peut aboutir à l'identification de deux molécules différentes avec la même valeur représentative, alors qu'elles présentent des structures et des interactions avec la cible distinctes.

Architactur des CNN pour deduire une valeur descriptive de molécule [31]

Luiz Anastacio Alves et al proposent une architectur avec l'utilisation de CNN pour déduire une valeur descriptive de molécule par la décomposition d'une molécule en matrices de caractéristiques et d'adjacence, qui sont ensuite traitées par convolution pour réduire la dimensionnalité. Ensuite calculer la similarité utilisent les sortes de deep learning suivis par classement des similarités finalement recommandant les meilleurs compounds(voir la figure 2.12).

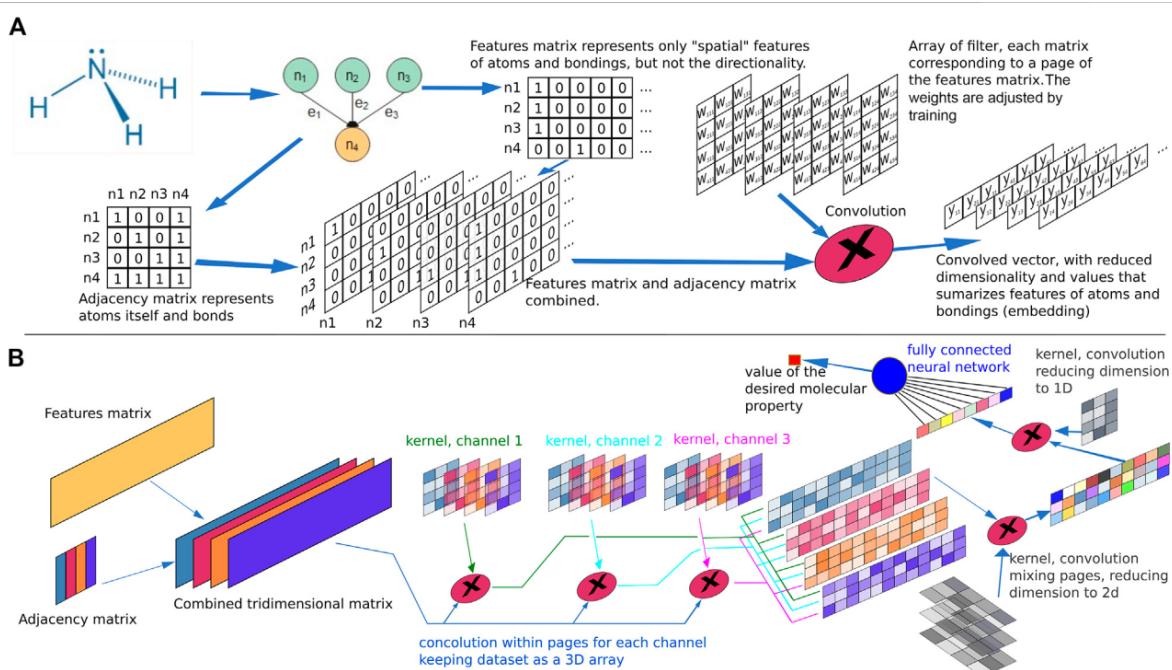


FIGURE 2.12 – Molecular property representation [31]

Les critiques

- **Complexité de mise en œuvre :** L'utilisation de réseaux de neurones convolutifs (CNN) pour déduire une valeur représentative d'une molécule s'avère un processus complexe à plusieurs niveaux. Premièrement, la représentation de la structure moléculaire en tant que données d'entrée pour le CNN pose un défi. Ensuite, l'architecture du CNN doit être capable de capturer les caractéristiques pertinentes de la structure moléculaire et les traduire en une représentation numérique utile. L'entraînement d'un modèle CNN requiert un ensemble de données de haute qualité **étiqueté** avec des informations sur les propriétés des molécules.
- **Temps d'entraînement plus long :** Les Convolutional Neural Networks (CNN) peuvent être coûteux en termes de temps d'exécution, surtout lorsqu'ils sont utilisés pour des tâches complexes telles que la représentation de molécules. Cependant, il existe des alternatives plus rapides et fiables pour calculer des valeurs représentatives de molécule comme l'utilisation des fonction qui calculer une valeur unique représentant les propriété de molécule en utilisent les fingerprints. le choix de la méthode dépend du compromis entre la précision, la vitesse d'exécution et les ressources disponibles. Le temps d'exécution est un facteur critique, explorer des alternatives telles que les empreintes digitales ou les modèles pré-entraînés peut être plus judicieux.
- **Fiabilité de la sélection :** L'évaluation de la similarité moléculaire par l'apprentissage automatique utilisant des réseaux de neurones convolutifs (CNN) peut présenter des défis en termes de fiabilité. Si les valeur représentative des molécules ne sont pas fiables, cela risque d'entraîner la sélection de molécules peu susceptibles de réussir les tests, ralentissant ainsi le processus de découverte de nouveaux médicaments.

L'architacture de graph edit distance[18]

Carlos Garcia-Hernandez et al utilisent la distance d'édition des graphes pour mesurer leur similarité(figure 2.13). Les composés chimiques sont représentés par des structures moléculaires avec des atomes sous forme des graphes. En suit graph edit distance similarity est utilisée pour comparer les deux graphes réduits. Cette méthode permet d'évaluer la similarité et quantifie le nombre minimum d'opérations d'édition (insertion, suppression) nécessaires pour transformer un graphe en un autre.[18]

Les opérations d'édition autorisées incluent :

- Insertion/suppression d'un sommet
- Insertion/suppression d'un arête

Les critiques

- **Complexité de mise en œuvre :** L'utilisation de la distance d'édition pour mesurer la similarité entre les graphes moléculaires ajoute une couche de complexité supplémentaire. Le calcul de la distance d'édition est un problème NP-complet, ce qui signifie que sa résolution exacte pour des graphes de taille importante devient rapidement impraticable.[44]
- **Temps d'entraînement plus long :** Stocker et manipuler ces graphes durant le processus de calcul de la distance d'édition peut s'avérer gourmand en mémoire, surtout lorsqu'il s'agit de comparer un grand nombre de molécules. La comparaison d'un grand nombre de molécules implique un nombre accru de calculs de distance d'édition, augmentant ainsi le temps d'exécution global.

- Fiabilité de la sélection :** L'utilisation de l'apprentissage automatique pour mesurer la similarité entre les molécules à l'aide de graphes peut présenter des limites en termes de fiabilité. En effet, les performances des modèles d'apprentissage automatique peuvent être affectées par des problèmes de sur-apprentissage et de sous-apprentissage, ce qui peut conduire à des scores de similarité non fiables. La sélection de molécules pour les tests biologiques basée uniquement sur des scores de similarité issus de modèles d'apprentissage automatique peut s'avérer problématique. Si les scores de similarité ne sont pas fiables, cela peut conduire à la sélection de molécules ayant une faible probabilité de succès dans les tests, ralentissant ainsi le processus de découverte de nouveaux médicaments.

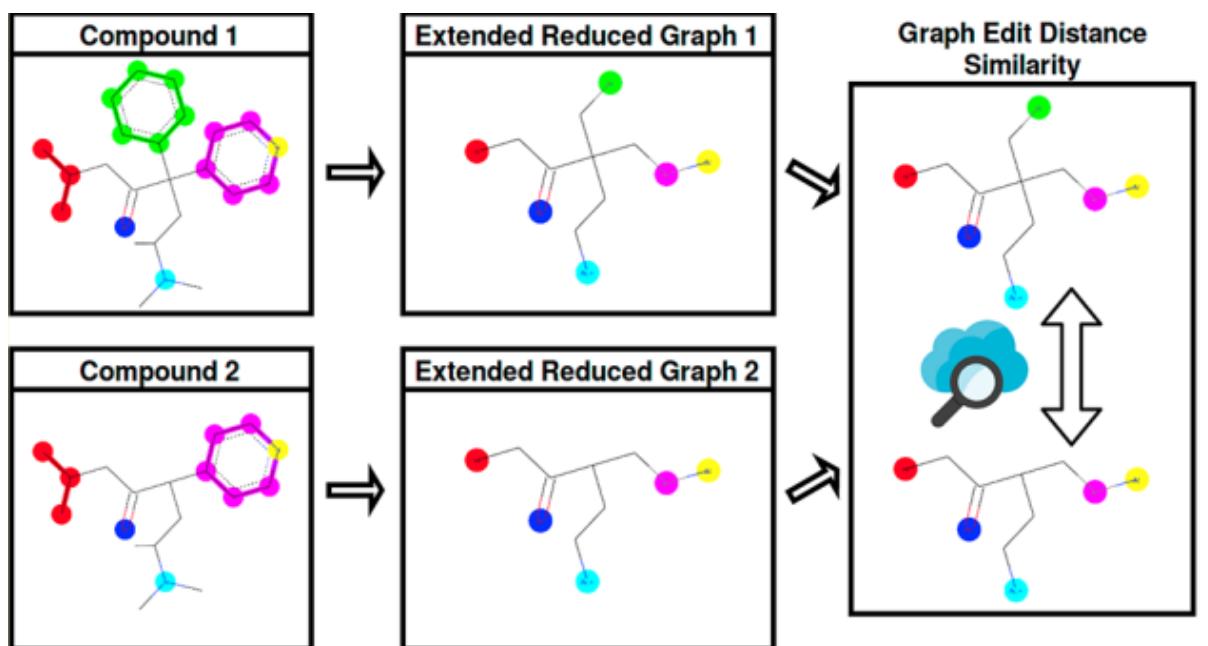


FIGURE 2.13 – Organigrammes de comparaison moléculaire [18]

2.6 Conclusion

L'apprentissage profond, offre de nombreuses opportunités pour accélérer différentes étapes du processus de découverte de nouveaux médicaments. Il s'appuie sur des techniques telles que le codage des molécules, l'évaluation des interactions entre les molécules, l'extraction de caractéristiques moléculaires et la prédiction de similarité entre les molécules. L'un des principaux avantages de l'apprentissage profond réside dans sa capacité à apprendre automatiquement à partir de grands ensembles de données. Cette caractéristique le rend particulièrement utile pour traiter des entrées hautement dimensionnelles, telles que des matrices, des molécules, des graphes moléculaires et des séquences. Dans le processus du virtual screening, l'utilisation du l'apprentissage profond permet d'améliorer la qualité des résultats, rendant ainsi les programmes plus performants et fiables. Cependant, il existe des défis à relever. L'entraînement des réseaux neuronaux profonds nécessite une grande quantité de données étiquetées et des ressources informatiques importantes. De plus, l'interprétabilité des modèles d'apprentissage profond est souvent limitée, ce qui rend difficile la compréhension de leurs décisions internes. Dans le chapitre suivant, nous proposons une architecture basée sur le Deep Learning pour améliorer la fiabilité

des résultats de virtual screening concernant les molécules recommandées pour les tests biologiques, comparer l'architecture proposée aux autres approches existantes discutées dans ce chapitre, discuter de chaque étape et justifier le choix des méthodes utilisées dans l'architecture proposée.

CHAPITRE 3

CONCEPTION

3.1 Introduction

Pour aborder le défi du virtual screening, il est essentiel de concevoir des architectures efficaces. La mesure de similarité revêt une importance cruciale. Il existe une multitude de méthodes et de métriques pour calculer cette similarité, ce qui rend le choix de la métrique la plus adaptée pour évaluer la similitude entre les molécules un véritable défi.

Pour résoudre ce problème, nous proposons d'utiliser le deep learning. Cette approche permet de calculer un score de similarité basé sur plusieurs métriques de similarité pour plus de fiabilité et de performance.

Notre architecture prendra en charge l'optimisation du processus de virtual screening, en visant à améliorer la mesure de similarité.

Dans ce chapitre, nous allons d'abord définir quelques fondements théoriques. Ensuite, nous expliquerons notre conception en détaillant les solutions que nous avons proposées.

3.2 L'organigramme de l'architecture proposée DL-VS

Notre architecture pour le Virtual Screening (VS) basé sur la similarité s'appuie sur un processus structuré et efficace pour identifier les molécules prometteuses parmi un vaste ensemble de données. L'organigramme (3.1) illustre les étapes clés de ce processus.

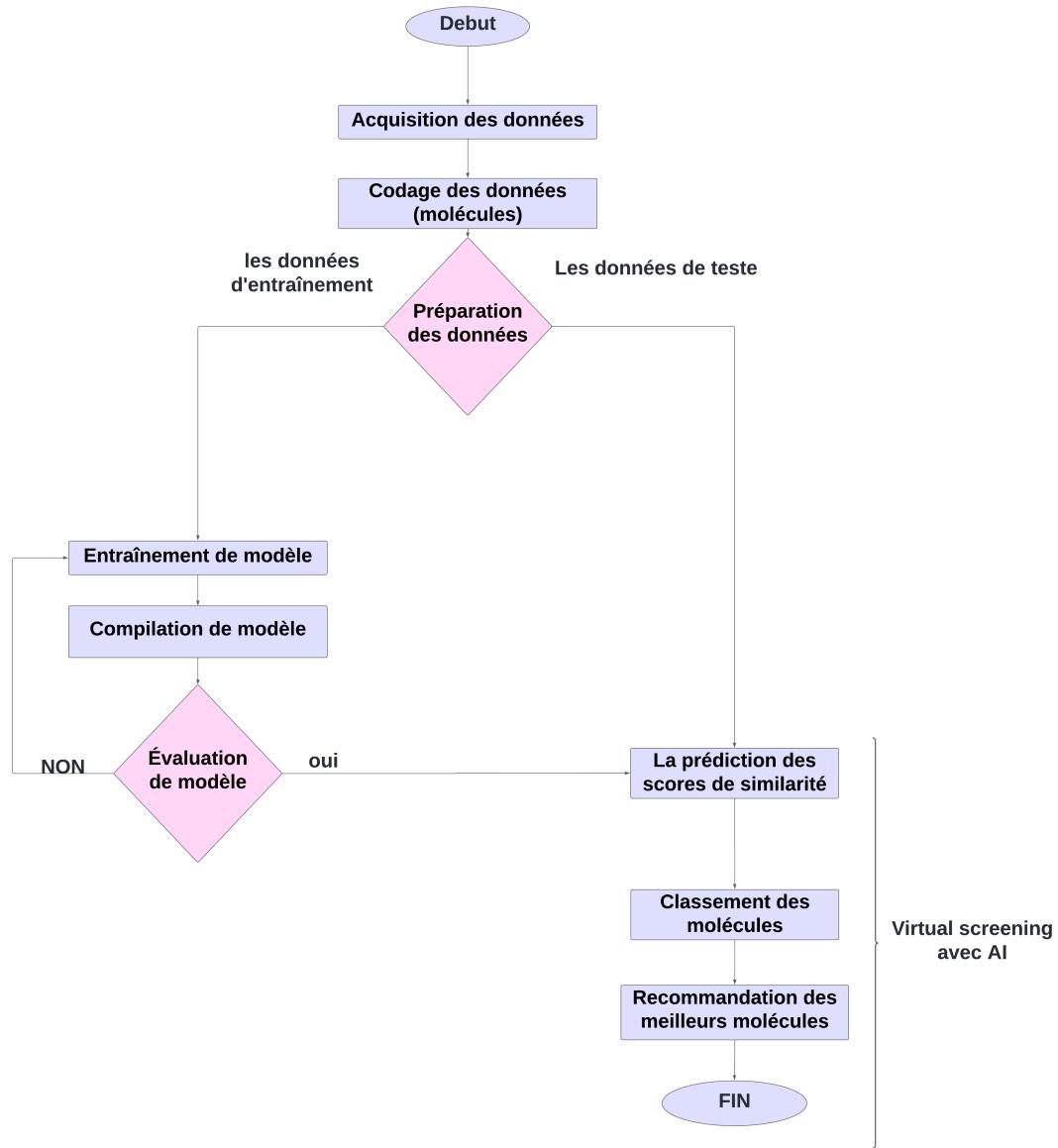


FIGURE 3.1 – Organigramme de l'architecture proposée DL-VS

3.3 L'architecture proposée DL-VS

L'intégration du deep learning et des réseaux de neurones convolutionnels au virtual screening permet d'améliorer considérablement la fiabilité des résultats. Notre architecture consiste à combiner plusieurs types de similarités pour déduire un score de similarité global. Cela permet de prendre en compte différentes dimensions de la similitude entre les molécules, augmentant ainsi la précision de notre recommandation. La figure 3.2 présente l'architecture que nous avons proposée.

La première étape de notre modèle implique la collecte de données provenant de diverses sources (dataset) associées au virtual screening. Ensuite, nous procédons à la sélection des attributs pertinents pour notre modèle. Cela implique une compréhension

approfondie du schéma relationnel de chaque source de données et de la signification de chaque attribut. La complexité réside dans le regroupement (aggregation) des attributs et la création d'un ensemble de données homogène. Une fois que les données ont été regroupées et qu'un ensemble de données homogène a été construit, la prochaine étape consiste à encoder les molécules.

Une fois le codage terminé, l'étape suivante consiste à calculer les attributs, c'est-à-dire à construire un jeu de données basé sur :

- Le calcul de la similarité entre l'ensemble des molécules candidates que nous avons collectées, et les molécules de référence qui ont démontré une interaction efficace avec la maladie que nous cherchons à traiter.
- Le calcul d'une valeur représentative pour chaque molécule candidate.
- Le calcul d'une valeur qui capture les propriétés communes entre les molécules de référence et les molécules candidates.

Ce jeu de données sera utilisé pour l'apprentissage automatique de notre modèle, qui attribuera un score à chaque molécule. Une fois que notre modèle d'IA est entraîné et validé, il sera capable de générer des scores de similarité pour chaque molécule utilisée dans la recherche d'un médicament pour une maladie donnée. En se basant sur ces scores, DL-VS classe toutes les molécules dans un ordre décroissant.

Dans l'étape finale, DL-VS sélectionne les meilleures molécules pour les recommander aux chercheurs en vue de tests biologiques. Ces molécules, ayant les scores de similarité les plus élevés, présentent ainsi le plus grand potentiel d'efficacité thérapeutique pour la maladie que nous voulons traiter.

Pour une meilleure compréhension de notre architecture, nous allons détailler chaque partie dans une section spécifique.

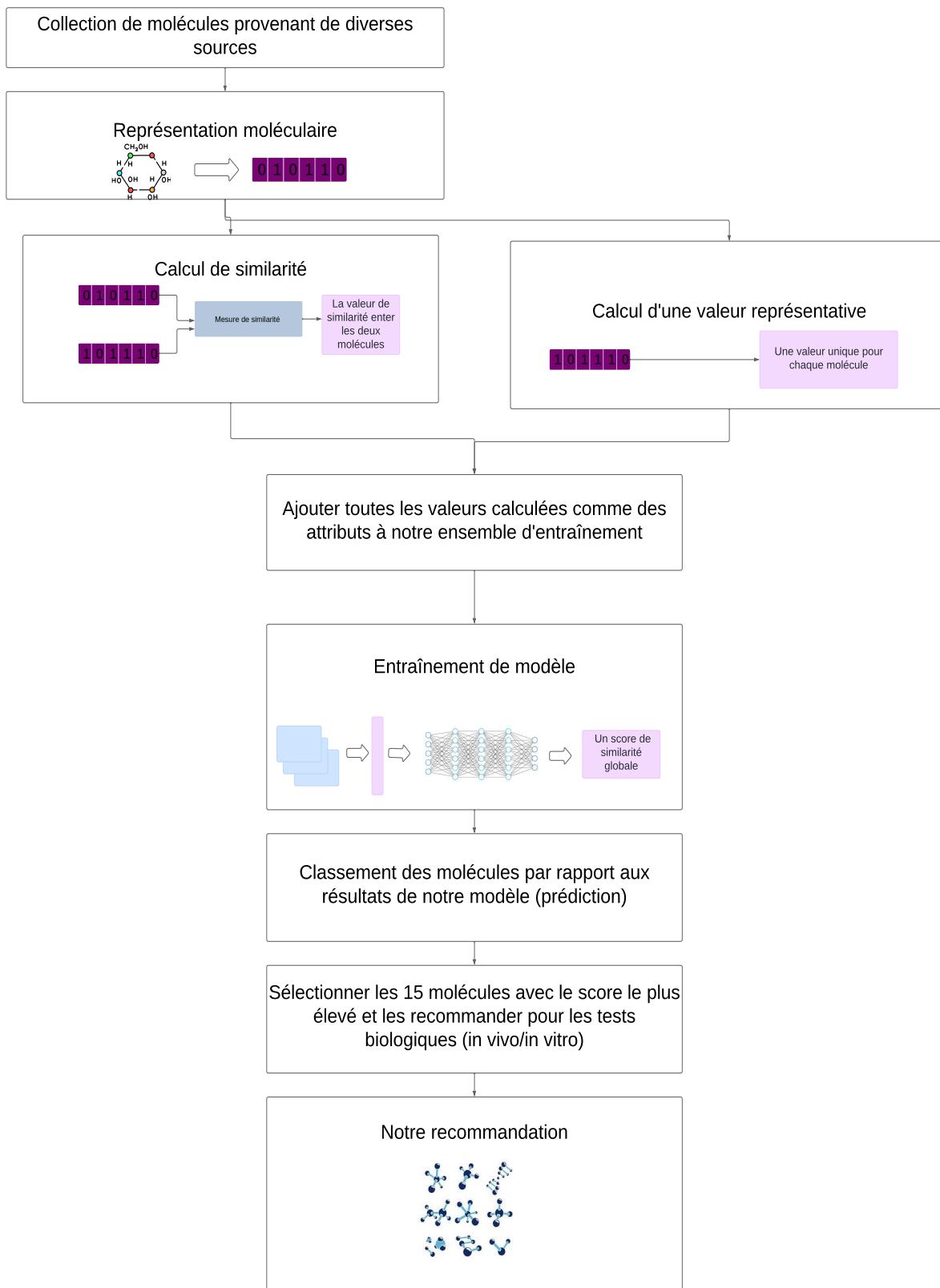


FIGURE 3.2 – L’architecture proposée DL-VS
41

3.4 La collecte des données

Concevoir une architecture efficace pour la recommandation des meilleures molécules nécessite un ensemble de données complet et de haute qualité. La collecte des données est une étape cruciale dans le processus de découverte de médicaments. Travailler avec plusieurs ensembles de données provenant de différentes sources est complexe, mais c'est essentiel pour obtenir une variété de données représentatives. Les étapes importantes à considérer lors de la collecte de données :

- **Compréhension des attributs** : Avant de collecter les données, il est crucial de comprendre le sens de chaque attribut dans les différents ensembles de données. Cela nous permettra de renommer les attributs de manière cohérente et de garantir une interprétation correcte lors de l'analyse ultérieure.
- **Schéma relationnel** : Chaque ensemble de données a son propre schéma relationnel, c'est-à-dire comment les tables ou les attributs sont liés entre elles. Comprendre ces relations est essentiel pour collecter les données de manière appropriée.
- **Homogénéité des données** : L'incohérence entre les données provenant de différentes sources doit être traitée pour garantir la cohérence des données consolidées.
- **Traitements des données** : Appliquer des techniques de nettoyage des données pour gérer les valeurs manquantes, les doublons et les valeurs aberrantes.
- **Regroupement des données** : Une fois la compréhension des différents schémas relationnels des sources de données acquise, la concaténation des différentes bases de données permettra d'aboutir à une base de données homogène.

3.4.1 Prétraitement de la base de données (molécules)

Une fois le jeu de données homogène construit, nous passons à l'étape de prétraitement. Cette étape vise à garantir la qualité des données et à les préparer pour l'analyse ultérieure. Tout d'abord, nous éliminons les doublons de molécules en supprimant les instances redondantes. Cela permet de réduire la taille du jeu de données et d'éviter les biais potentiels. Ensuite, nous nous attaquons au problème des valeurs manquantes, qui peuvent affecter négativement la précision des analyses. Nous utilisons la bibliothèque RD-Kit pour interroger des bases de données chimiques et récupérer les valeurs manquantes pour chaque molécule. RDKit envoie une requête avec l'identifiant de la molécule pour obtenir la valeur manquante correspondante. En remplissant ainsi les valeurs manquantes, nous améliorons la qualité globale du jeu de données et garantissons sa fiabilité pour les étapes du Virtual Screening [45].

Dans notre cas, nous nous concentrons sur la formule chimique, le nom et l'identifiant unique (ID), ainsi que la représentation SMILES de chaque molécule, ce qui nous permet de les différencier et d'éviter les téléchargements redondants.

3.5 Codage des molécules

Afin de formuler nos recommandations, il est nécessaire de coder les molécules. Plus précisément, le SMILES est un attribut qui représente les molécules, donc pour chaque molécule nous allons coder son SMILES canonique.

Initialement, nous avons opté pour le codage MACCS. Cependant, le codage MACCS (Molecular ACCess System) est un type d'empreinte moléculaire qui se base sur un ensemble prédéfini de 166 motifs structuraux.[9] Il est utile pour certaines applications, mais il peut ne pas capturer toutes les propriétés chimiques et biologiques d'une molécule car il est limité par son ensemble fixe de motifs.[14]

Pour pallier cette limitation, nous avons adopté la méthode des empreintes moléculaires de Morgan (Morgan Fingerprinting). L'utilisation des empreintes de Morgan nous permet de capturer avec précision non seulement les propriétés structurelles, mais aussi les propriétés chimiques et biologiques des molécules. Ce codage riche en informations est essentiel pour nos analyses, car il contribue à une meilleure caractérisation des entités moléculaires dans notre modèle.[46]

3.6 Calcul d'une valeur représentative pour chaque molécule

Suite au codage des molécules et à l'obtention de leurs empreintes digitales, nous cherchons à représenter les propriétés regroupées dans chaque empreinte par une valeur décimale unique. Cette valeur représentative, calculée pour chaque molécule, sera ensuite intégrée à notre ensemble de données d'entraînement afin de capturer l'ensemble des propriétés moléculaires lors de la prédiction de score de similarité entre les molécules par notre modèle IA.

Pour calculer cette valeur, nous procédons à la création d'un vecteur de caractéristiques. Ce vecteur est initialisé de manière à ce que chaque élément corresponde à une puissance de deux, débutant par 2^0 pour le premier élément, 2^1 pour le second, et ainsi de suite jusqu'à $2^{(n-1)}$ pour le n-ième élément, où n est la taille de l'empreinte de Morgan.

Ensuite, nous calculons une valeur représentative pour chaque molécule en multipliant le vecteur d'empreintes de Morgan par le vecteur de caractéristiques. La valeur résultante est une somme pondérée (3.1) qui intègre les propriétés chimiques et biologiques de chaque molécule, donnant ainsi une représentation numérique unique pour chaque entité moléculaire.

$$V = \sum_{i=0}^{n-1} \text{fingerprint}[i] \times 2^i \quad (3.1)$$

3.7 Identifier les propriétés communes entre les molécules candidates et les molécules de référence

Pour pouvoir capturer les propriétés communes entre les molécules de référence et les molécules candidats nous procédons par une multiplication élément par élément des empreintes de Morgan des molécules candidates avec celles des molécules de référence. Cette opération vise à isoler les caractéristiques communes, se traduisant par un nouveau vecteur qui met en évidence les similitudes structurelles. Le vecteur résultant est composé de valeurs binaires. Pour faciliter l'analyse quantitative, nous convertissons ce vecteur en valeurs numériques réelles.

Pour chaque molécule candidate i , nous ajoutons n attributs $A_{i1}, A_{i2}, \dots, A_{in}$. Avec n est le nombre des molécules de référence. Chaque attribut A_{ij} est calculé comme suit :

1. Multiplication élément par élément des empreintes pour obtenir un vecteur de similarité S_{ij} .
2. Conversion du vecteur de similarité S_{ij} en une valeur numérique V_{ij} qui représente la similarité entre C_i et R_j .
3. Assignation de la valeur numérique V_{ij} à l'attribut A_{ij} .

Algorithm 1 Intersection Fingerprint

Require: $v1, v2$

```
1: result  $\leftarrow [a \cdot b \text{ for } a, b \text{ in } \text{zip}(v1, v2)]
2: cod_fer  $\leftarrow [2^i \text{ for } i \text{ in range(len}(v1))]
3: return  $\sum(\text{result}[i] \cdot \text{cod_fer}[i] \text{ for } i \text{ in range(len(result)))})$$$ 
```

3.8 Le calcul de similarité

Dans le but d'identifier des candidats médicaments potentiels pour la maladie ciblée, nous proposons une approche basée sur l'apprentissage profond. Cette approche vise à mesurer la similarité entre les molécules de notre ensemble de données et les molécules connues pour leur interaction favorable avec la maladie. En utilisant des métriques de similarité multiples, nous construisons un modèle de deep learning capable de prédire un score de similarité pour chaque molécule. Ce modèle exploite les propriétés partagées entre les molécules actives qui ont un bon résultat avec la maladie et les molécules de notre ensemble de données d'entraînement, permettant ainsi d'identifier les candidats les plus prometteurs. En substance, notre approche consiste à rechercher, au sein de notre ensemble de données, les molécules présentant des propriétés similaires à celles des molécules actives, afin de proposer des recommandations de médicaments potentiels.

Algorithm 2 Similarity Calculation

```
1: Input : vector1, vector2, and a string metric
2: Output : float
3: if metric == "cosine" then
4:   dot_product  $\leftarrow \text{np.dot}(\text{vector1}, \text{vector2})$ 
5:   norm1  $\leftarrow \text{np.linalg.norm}(\text{vector1})$ 
6:   norm2  $\leftarrow \text{np.linalg.norm}(\text{vector2})$ 
7:   return dot_product/(norm1  $\times$  norm2)
8: else if metric == "Tanimoto" then
9:   intersection  $\leftarrow \sum_{i=1}^n \min(\text{vector1}_i, \text{vector2}_i)$ 
10:  union  $\leftarrow \sum_{i=1}^n \max(\text{vector1}_i, \text{vector2}_i)$ 
11:  return intersection/union
12: else
13:   raiseValueError("Invalidsimilaritymetric.")
14: end if
```

Ces valeurs calculées sont ensuite stockées dans des attributs spécifiques créés à cet effet. Ainsi, pour (n) molécules de référence, nous créons (n) attributs pour chacune

des deux mesures de similarité. Pour pouvoir ajouter les scores de similarité calculés au l'ensemble d'entraînement.

3.9 La construction d'un ensemble de données d'entraînement

La construction de notre ensemble de données d'entraînement requiert deux types de données distincts :

- **Un ensemble de données regroupant les molécules candidates** : Ce jeu de données servira d'instances pour notre modèle d'apprentissage.
- **Un ensemble de données regroupant les molécules de référence** : Ce jeu de données regroupe les molécules qui ont une bonne interaction avec la maladie.

L'ensemble de données d'entraînement utilisé pour notre modèle d'apprentissage automatique est composé de plusieurs types d'attributs décrivant les molécules et leur interaction avec la maladie cible.

- **Attributs de similarité** : Ces attributs représentent les scores de similarité calculés entre chaque molécule de notre ensemble de données et un ensemble de molécules de référence connues pour leur activité thérapeutique contre la maladie cible. Plusieurs mesures de similarité sont employées pour capturer différentes perspectives de la similarité moléculaire.
- **Attributs de propriétés communes** : Ces attributs capturent les propriétés communes partagées entre chaque molécule de notre ensemble de données et les molécules de référence. Ils décrivent les caractéristiques structurales, physico-chimiques et pharmacologiques qui sont susceptibles d'être importantes pour l'activité thérapeutique.
- **Attributs de propriétés moléculaires** : Ces attribut représentent les propriétés chimiques et biologiques de chaque molécule de notre ensemble de données.
- **Attribut de label** : Cet attribut correspond au score de similarité mesuré entre chaque molécule de notre ensemble de données et les molécules de référence. Il sert de variable cible pour le modèle d'apprentissage automatique, indiquant le niveau d'activité thérapeutique attendue pour chaque molécule.

La table 3.3 présente une représentation schématique de notre ensemble de données d'entraînement, accompagnée d'une légende décrivant la signification de chaque attribut.

FIGURE 3.3 – L’ensemble de données d’entraînement

#	valeur représentative	cosine(Ci,R0)	cosine(Ci,R1)	cosine(Ci,R119)	Tanimoto(Ci,R0)	Tanimoto(Ci,R1)	Tanimoto(Ci,R119)	intersection(Ci,R0)	intersection(Ci,R1)	intersection(Ci,R119)	Score_similitude i
C0	0,7	0,4	0,6	0,4	0,6	0,2	0,8	0,6	0,7	0,6	0,4	0,3
C1	0,8	0,7	0,5	0,8	0,4	0,6	0,4	0,3	0,8	0,3	0,2	0,8
C2	0,3	0,6	0,2	0,6	0,8	0,6	0,2	0,5	0,7	0,6	0,2
C3	0,7	0,2	0,8	0,4	0,6	0,7	0,8	0,7	0,3	0,4	0,7	0,6
.
.
Cn	0,2	0,5	0,2	0,3	0,8	0,3	0,5	0,5	0,2	0,8	0,8	0,7

3.9.1 Description de l’ensemble de données d’entraînement

L’ensemble d’entraînement représente les valeurs de similarité entre les candidats (C0, C1, C2, C3, ..., Cn) et différentes références (R0, R1, ..., R119). Les colonnes sont organisées pour présenter plusieurs types de mesures de similarité :

- **Valeur représentative** : La première montre une valeur représentative pour chaque molécule.
- **Cosine(Ci, Rj)** : Ces colonnes montrent les valeurs de similarité cosinus entre chaque molécule Ci et les différentes molécules de référence R0, R1, ..., R119.
- **Tanimoto(Ci, Rj)** : Ces colonnes présentent les indices de similarité de Tanimoto entre chaque molécule candidat Ci et les différentes molécules de références R0, R1, ..., R119.
- **Intersection(Ci, Rj)** : Ces colonnes montrent les valeurs d’intersection entre chaque molécule candidat Ci et les différentes molécules de références R0, R1, ..., R119.
- **Score de similarité i** : La dernière colonne présente un score global de similarité pour chaque molécule candidat, pour initialiser le score de similarité de notre ensemble d’entraînement, on a calculé la moyenne des 120 similarités cosine et des 120 similarités tanimoto. Le modèle ne se contente pas de faire une simple moyenne des similarités, mais prend en compte des attributs supplémentaires pour trouver le score de similarité. Ces attributs représentent les propriétés communes entre les molécules de référence et les molécules candidates. Le modèle utilise une valeur représentative pour chaque molécule.

3.10 Le modèle d’apprentissage profond pour DL-VS

Avant de débuter l’entraînement d’un modèle d’IA pour la prédiction de score de similarité moléculaire, il est crucial de préparer les données de manière adéquate. Cette

étape implique la suppression des informations superflues telles que les ID de molécules ou les SMILES, en ne conservant que les valeurs pertinentes pour l'analyse. Ensuite les données sont normalisées pour garantir une échelle uniforme et faciliter l'apprentissage du modèle.

L'étape suivante consiste à choisir un modèle d'apprentissage automatique approprié et de définir son architecture.

Les réseaux neuronaux convolutifs (CNN) s'avèrent souvent efficaces pour les tâches de similarité moléculaire. L'architecture du réseau neuronal doit être soigneusement définie, en précisant le nombre de couches, le nombre de neurones par couche, les fonctions d'activation et d'autres paramètres pertinents.

L'entraînement du modèle implique l'alimentation itérative de l'ensemble d'entraînement par lots de données. Au cours de ce processus, les paramètres du modèle (poids et biais) sont ajustés à l'aide d'un algorithme d'optimisation pour minimiser une fonction de perte.

Les performances du modèle sont évaluées sur l'ensemble de validation à chaque itération d'apprentissage, et l'entraînement est poursuivi jusqu'à ce qu'un critère d'arrêt prédéfini soit atteint, tel qu'un nombre maximal d'itérations ou une amélioration minimale des performances.

Notre modèle d'apprentissage profond s'articule autour d'une architecture réseau neuronal convolutif suivie d'un réseau neuronal dense.

- **Couche convolutif :** Le modèle vise à capturer les motifs et les relations spatiales entre les molécules, permettant d'identifier les propriétés partagées entre les molécules de référence et les molécules candidates et diminuer le nombre d'attributs à traiter afin d'optimiser le calcul du similarité pour les couches entraînement connectés
- **Réseau neuronal dense :** Les données issues de la couche convulsive sont réduites en dimensionnalité, seront ensuite traitées par un réseau neuronal dense. Ce réseau, composé de plusieurs couches cachées interconnectées, permettra d'apprendre des relations non linéaires complexes entre les caractéristiques des molécules et leur interaction avec la maladie. Le modèle prédira un score de similarité pour chaque molécule de notre ensemble de données, indiquant son potentiel d'interaction positive avec la maladie.

3.10.1 Tableau récapitulatif de notre modèle

Le tableau (3.4) résume l'architecture de notre modèle. Le modèle est composé de plusieurs couches, chacune ayant une fonction spécifique. Les couches conv1d, conv1d1, conv1d2, conv1d3, conv1d4, conv1d5 et conv1d6 sont des couches convolutives 1D. Ces couches appliquent des filtres convolutifs aux données d'entrée afin d'extraire des caractéristiques. Les couches maxpooling1d, maxpooling1d1 et maxpooling1d2 sont des couches de pooling 1D. Ces couches réduisent la dimension des données en effectuant un pooling maximal sur des fenêtres temporelles. La couche flatten aplatis les données en un vecteur 1D. Les couches dense, dense1, dense2, dense3, dense4 et dense5 sont des couches denses. Ces couches appliquent une transformation linéaire aux données.

FIGURE 3.4 – Résumé du modèle

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 352, 64)	704
conv1d_1 (Conv1D)	(None, 343, 64)	41,024
conv1d_2 (Conv1D)	(None, 334, 64)	41,024
max_pooling1d (MaxPooling1D)	(None, 167, 64)	0
dropout (Dropout)	(None, 167, 64)	0
conv1d_3 (Conv1D)	(None, 158, 32)	20,512
conv1d_4 (Conv1D)	(None, 149, 32)	10,272
max_pooling1d_1 (MaxPooling1D)	(None, 74, 32)	0
dropout_1 (Dropout)	(None, 74, 32)	0
conv1d_5 (Conv1D)	(None, 65, 16)	5,136
conv1d_6 (Conv1D)	(None, 56, 16)	2,576
max_pooling1d_2 (MaxPooling1D)	(None, 28, 16)	0
dropout_2 (Dropout)	(None, 28, 16)	0
flatten (Flatten)	(None, 448)	0
dense (Dense)	(None, 448)	201,152
dense_1 (Dense)	(None, 400)	179,600
dense_2 (Dense)	(None, 300)	120,300
dense_3 (Dense)	(None, 200)	60,200
dense_4 (Dense)	(None, 180)	36,180
dense_5 (Dense)	(None, 1)	181

3.10.2 L'algorithme de modèle d'intelligence artificielle

Algorithm 3 Build and Train Model

Require: x_train , y_train , $input_shape$

```
1: model ← Sequential()
2: model.add(Conv1D(filters=64,kernel_size=10,activation='relu',input_shape=input_shape,
    kernel_regularizer=l2(0.01)))
3: model.add(Conv1D(filters=64,kernel_size=10,activation='relu',kernel_regularizer=l2(0.01))

4: model.add(Conv1D(filters=64,kernel_size=10,activation='relu',kernel_regularizer=l2(0.01)))

5: model.add(MaxPooling1D(pool_size=2))
6: model.add(Dropout(0.5))
7: model.add(Conv1D(filters=32,kernel_size=10,activation='relu',kernel_regularizer=l2(0.01))

8: model.add(Conv1D(filters=32,kernel_size=10,activation='relu',kernel_regularizer=l2(0.01)))

9: model.add(MaxPooling1D(pool_size=2))
10: model.add(Dropout(0.5))
11: model.add(Conv1D(filters=16,kernel_size=10,activation='relu',kernel_regularizer=l2(0.01))

12: model.add(Conv1D(filters=16,kernel_size=10,activation='relu',kernel_regularizer=l2(0.01))

13: model.add(MaxPooling1D(pool_size=2))
14: model.add(Dropout(0.5))
15: model.add(Flatten())
16: model.add(Dense(units=448,activation='relu',kernel_regularizer=l2(0.01)))
17: model.add(Dense(units=400,activation='relu',kernel_regularizer=l2(0.01)))
18: model.add(Dense(units=300,activation='relu',kernel_regularizer=l2(0.01)))
19: model.add(Dense(units=200,activation='relu',kernel_regularizer=l2(0.01)))
20: model.add(Dense(units=180,activation='relu',kernel_regularizer=l2(0.01)))
21: model.add(Dense(units=1,activation='linear'))
22: model.compile(loss='mae',optimizer='adam')
23: model.summary()
24: early_stopping ← EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
25: historique ← model.fit(x=x_train, y=y_train, epochs=100, batch_size=128, validation_split=0.2,
    callbacks=[early_stopping])
26: return model, historique
```

Dans notre modèle, nous avons utilisé plusieurs fonctions et techniques d'optimisation pour améliorer les performances et pour généraliser le modèle. Voici une explication détaillée de chaque fonction d'optimisation utilisée et leur utilité :

1. **Regularisation L2** : La régularisation L2 est utilisée pour éviter le surapprentissage (overfitting) en ajoutant une pénalité pour les poids de grande amplitude.[47] Dans notre modèle, la régularisation L2 est appliquée aux couches convolutives et

aux couches denses. Le modèle est encouragé à garder les poids aussi petits que possible, ce qui peut aider à réduire le surapprentissage. [40]

2. **Dropout** : Le Dropout est une technique de régularisation où un certain pourcentage des neurones est aléatoirement désactivé (mis à zéro) pendant chaque itération d'entraînement. Cela aide à améliorer la généralisation du modèle.[40]
3. **Early Stopping** : L'early stopping est une technique qui arrête l'entraînement du modèle lorsque la performance sur un ensemble de validation cesse de s'améliorer. Cela permet de prévenir le surapprentissage et de réduire le temps d'entraînement.[39] Le callback EarlyStopping surveille la perte de validation (val_loss) et arrête l'entraînement si la perte ne s'améliore pas après 10 epochs (patience=10). De plus, il restaure les poids du modèle à l'état où la perte de validation était la meilleure (restore_best_weights=True).[40]
4. **Optimizer Adam** : Adam (Adaptive Moment Estimation) est un algorithme d'optimisation qui combine les avantages de deux autres méthodes populaires AdaGrad et RMSProp. Adam calcule des taux d'apprentissage adaptatifs pour chaque paramètre, en utilisant les moments du premier ordre (moyenne) et du second ordre (variance) des gradients.[30] Adam est utilisé dans notre modèle comme optimiseur du modèle. Il est apprécié pour sa rapidité de convergence et sa capacité à bien performer avec un minimum de réglage de paramètres.[30]

Ces techniques combinées permettent de construire un modèle plus robuste et capable de mieux généraliser sur des données non vues.

3.11 Comparaison entre notre architecture (DL-VS) et les travaux connexes

Pour évaluer les performances de notre architecture et identifier ses points forts par rapport aux architectures existantes (Architecture of Siamese RNN similarity model[22], Architactur des CNN pour deduire une valeur descriptive de molécule [31], L'architacture de graph edit distance[18]), nous avons réalisé une analyse comparative. Le tableau(3.5) résume les caractéristiques clés des architectures comparées, mettant en évidence les différences et similitudes entre elles.

FIGURE 3.5 – Comparaison entre l'architecture proposée et les architectures existants

	Codage			Smilarité				Deep Learning			
	SMILES	Graphe moléculaire moléculaire	FingerPrint	Une similarité	Plusieurs similarités	Valeure descriptive	DNN	GNN	CNN	RNN	
DL-VS	*		*		*	*	*	*		*	
SIAMESE RNN SIMILARITY MODEL	*		*		*			*			*
MOLÉCULE PROPERTY REPRESENTATION		*		*		*	*			*	
GRAPH EDIT DISTANCE SIMILARITY		*		*					*		

3.11.1 Les méthodes employées dans notre architecture(DL-VS)

Codage :

Pour le codage des molécules, nous avons opté pour deux méthodes : **les SMILES** et **les fingerprints**. Nous avons privilégié les SMILES en raison de leur disponibilité et parce qu'ils permettent d'extraire de multiples informations grâce aux fonctions de la bibliothèque RDKit.[48] En outre, les SMILES s'avèrent particulièrement utiles pour traiter les valeurs manquantes dans nos données, offrant un traitement rapide et efficace. Concernant les fingerprints, la bibliothèque RDKit offre des outils pour les calculer à partir des SMILES. Notre objectif est de déterminer le score de similarité, plutôt que de coder les molécules elles-mêmes. L'utilisation de modèles de deep learning ou de graphes pour le codage des molécules serait trop chronophage et pourrait conduire à des résultats peu fiables. C'est pourquoi nous avons choisi le codage par fingerprints, car RDKit fournit des fonctions éprouvées pour les calculer rapidement et avec précision. Un autre avantage des fingerprints est qu'ils nous aident à calculer la similarité entre les SMILES, puisqu'ils sont représentés sous forme de vecteurs, ce qui nous permet de calculer différents types de similarité.

Similitude :

Dans notre approche de calcul de similarité moléculaire, nous avons initialement tenté de calculer un seul type de similarité. Cependant, nous avons constaté que le changement du type de mesure de similarité (par exemple, passer de la similarité cosinus à la similarité de Tanimoto) entraînait des recommandations différentes pour une même maladie. Pour résoudre ce problème, nous avons décidé d'intégrer des modèles de deep learning afin d'améliorer le calcul de la similarité.

Pour améliorer le calcul de similarité, nous avons décidé de calculer plusieurs types de similarité (cosinus et Tanimoto) pour augmenter la précision de nos prédictions.

3.12 Conclusion

L'architecture proposée met en œuvre un processus de Virtual Screening (VS) basé sur la similarité, permettant de sélectionner un nombre restreint de molécules prometteuses parmi un vaste ensemble de données. Cette approche vise à faciliter la recommandation de molécules pour des tests biologiques et à accélérer le processus de découverte de médicaments. L'architecture est conçue pour garantir la fiabilité, la maintenabilité, les meilleures performances, la scalabilité et l'ensemble des fonctionnalités requises. Le chapitre suivant se consacrera à la mise en œuvre de l'architecture proposée et à l'analyse de ses résultats. Nous explorerons les performances de l'architecture, sa fiabilité, son évolutivité et sa capacité à répondre aux besoins spécifiques du domaine du Virtual Screening.

CHAPITRE 4

RÉALISATION ET OPTIMISATION

4.1 Introduction

Dans le chapitre précédent, nous avons présenté la conception du modèle proposé pour résoudre le problème de virtual screening. Ce chapitre se focalise sur la mise en œuvre de notre architecture. Nous explorerons les langages de programmation, les bibliothèques et les outils de développement qui ont été mobilisés pour concrétiser notre proposition. Nous présenterons également les tests et une analyse approfondie des résultats, en examinant le temps d'exécution et son interdépendance avec l'augmentation volumétrique des données. Face au défi imposant du big data, nous discuterons des stratégies permettant de le surmonter par l'exécution sur des architectures parallèles, telles que les cartes graphiques (GPU). Enfin, nous procéderons à une comparaison critique entre les performances temporelles des CPU et des GPU.

4.2 Environnement de développement

4.2.1 Environnement matériel

Latitude DELL 5320

- La RAM DDR4 : 16GO
- Processeur Intel® Core™ i5 de 11ème génération avec 8 cœurs
- Disque : SSD de 512 Go

La machine du CERIST

- La RAM DDR4 : 16GO
- Processeur Intel Xeon R-1370 avec 16 cœurs
- Carte graphique NVIDIA Geforce RTX3060 Ti basé sur l'architecture Ampere, ce GPU est équipé de 4864 cœurs CUDA. Il offre une fréquence de base de 1.41 GHz et peut atteindre jusqu'à 1.78 GHz en fréquence boost. Le RTX 3060 Ti est doté de

8 Go de mémoire vidéo GDDR6 avec une interface mémoire de 256 bits. Il intègre la deuxième génération de coeurs de raytracing et la troisième génération de coeurs Tensor.

4.2.2 Environnement logiciel

Python

Python est un langage de programmation généraliste de haut niveau, interprété et multiparadigme, caractérisé par sa simplicité, sa lisibilité et sa puissance. Python est un langage interprété, ce qui signifie qu'il n'est pas nécessaire de le compiler avant de l'exécuter. Cela permet un développement rapide et interactif.[49] Il dispose d'un vaste écosystème de bibliothèques tierces couvrant un large éventail de domaines, tels que le développement Web, l'analyse de données, l'apprentissage automatique, la science des données et la visualisation de données.[49]

Visual Studio Code

Visual Studio Code est un éditeur de code gratuit et open-source développé par Microsoft. Il est l'un des éditeurs de code les plus populaires au monde, utilisé par les développeurs de logiciels pour écrire, déboguer et tester du code dans une variété de langages de programmation.[50] Visual Studio Code offre une large gamme de fonctionnalités pour aider les développeurs à coder plus efficacement, notamment la coloration syntaxique, la complétion de code, le refactoring de code et la prise en charge du contrôle de version [50].

Les bibliothèques

La première étape dans le développement de notre application consiste à préparer l'environnement de traitement. Cela inclut l'installation des bibliothèques nécessaires et la configuration du logiciel en fonction du langage de programmation choisi.

- **Pandas** Pandas est une bibliothèque logicielle open-source écrite en Python pour la manipulation et l'analyse de données. Elle est particulièrement adaptée aux données tabulaires et offre un large éventail de fonctionnalités pour le nettoyage, le traitement, l'agrégation et la visualisation des données.[51]
- **Numpy** NumPy (Numerical Python) est une bibliothèque logicielle open-source écrite en Python pour le calcul numérique. Elle fournit des structures de données et des fonctions optimisées pour les opérations mathématiques sur les tableaux multidimensionnels.[52] NumPy est un élément essentiel de la pile logicielle scientifique Python et est largement utilisée dans divers domaines, notamment :
 - Analyse de données
 - Apprentissage automatique et intelligence artificielle
 - Développement scientifique
 - Finance
- **Matplotlib** Matplotlib est une bibliothèque logicielle open-source écrite en Python pour la création de visualisations statiques, animées et interactives. Elle offre un

ensemble d'outils puissants et polyvalents pour représenter des données sous forme de graphiques, de diagrammes et de cartes.[53] Matplotlib est un élément essentiel de la pile logicielle scientifique Python et est largement utilisée dans divers domaines, notamment :

- Analyse de données
 - Apprentissage automatique et intelligence artificielle
 - Développement scientifique
 - Finance
- **Rdkit** RDKit (The Chemistry Development Kit) est une bibliothèque logicielle open-source écrite en Python pour la manipulation, l'analyse et la visualisation de données chimiques.[54] Elle offre un ensemble complet d'outils pour travailler avec des molécules, des structures chimiques et des réactions chimiques.[48] RDKit est largement utilisée dans divers domaines, notamment :
 - **Chimie médicinale et découverte de médicaments** : RDKit est utilisée pour la conception de molécules, la prédiction de propriétés moléculaires, l'analyse de structures chimiques et la gestion de bases de données chimiques.[54][48]
 - **Chimie théorique et computationnelle** : RDKit est utilisée pour calculer des propriétés moléculaires, simuler des réactions chimiques et modéliser des interactions moléculaires.[48][54]
 - **Matérialogie et science des matériaux** : RDKit est utilisée pour la conception de matériaux, la prédiction de propriétés de matériaux et l'analyse de structures cristallines.[54][48]
 - **Bioinformatique et biologie moléculaire** : RDKit est utilisée pour l'analyse de protéines, l'étude d'interactions protéine-ligand et la visualisation de structures biomoléculaires.[54][48]
 - **Sklearn** Sklearn est une bibliothèque logicielle open-source écrite en Python pour l'apprentissage automatique et l'analyse de données. Elle offre un large éventail d'algorithme et d'outils pour des tâches telles que la classification, la régression, le clustering, le traitement du langage naturel et la réduction de dimensionnalité.[55]
 - **Keras** Keras est une interface de programmation d'application (API) de haut niveau pour la création de réseaux de neurones artificiels (RNA) avec la bibliothèque TensorFlow. Elle offre une approche simplifiée et conviviale pour la construction et l'entraînement de modèles de deep learning, en s'appuyant sur les fonctionnalités puissantes de TensorFlow. Keras peut exécuter des modèles sur différents backends, tels que CPU, GPU et TPU, offrant une flexibilité et une évolutivité. Elle fournit une large gamme de couches prédefinies, couvrant les réseaux de neurones convolutifs (CNN), les réseaux de neurones récurrents (RNN), et les réseaux de neurones artificiels profonds (DNN). Keras s'intègre parfaitement avec TensorFlow, permettant d'accéder aux fonctionnalités avancées de TensorFlow pour des tâches de deep learning plus complexes.

4.3 Acquisition des données

Une fois l'environnement prêt, nous pouvons entamer l'acquisition des données, une phase cruciale de notre processus.

4.3.1 Sources des données utilisé

- **Pubchem** : PubChem, développée par les National Institutes of Health (NIH), est une base de données ouverte dédiée à la chimie. Son caractère ouvert permet aux utilisateurs de partager et d'accéder à des informations scientifiques sur les produits chimiques, favorisant ainsi une collaboration accrue au sein de la communauté scientifique. Depuis sa création en 2004, PubChem s'est imposée comme une référence incontournable pour les chercheurs, les étudiants et le grand public, leur offrant un accès à une vaste collection de données sur les substances chimiques.[56]
- **HMDB** : La base de données HMDB (Human Metabolome Database) est une ressource précieuse pour les chercheurs et les étudiants en sciences de la vie. Elle rassemble des informations complètes sur les métabolites humains, les enzymes impliquées dans leur métabolisme et les voies métaboliques dans lesquelles ils sont impliqués.[57]
- **CHEMBL** : (Chemical Database for Molecular Activity) est une base de données organisée manuellement et gratuite qui contient des informations sur des molécules bioactives ayant des propriétés similaires à celles des médicaments. Elle se distingue par sa couverture complète de tous les aspects de la découverte de médicaments et sa taille immense, avec des informations sur plus de 2,2 millions de composés et plus de 18 millions d'enregistrements d'effets sur les systèmes biologiques[58]. La figure 4.1 illustre les données incluses dans ChemBL.[58]

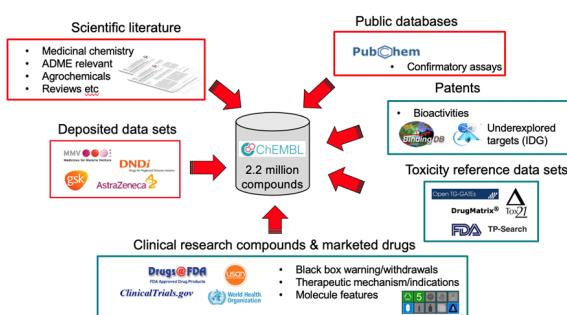


FIGURE 4.1 – Données incluses dans ChEMBL.

- **CheBI** : (Chemical Entities of Biological Interest) est un dictionnaire d'entités moléculaires gratuit qui se concentre sur les petits composés chimiques. Le terme **entité moléculaire** désigne tout atome, molécule paire, complexe, conformère, constitutionnellement ou isotopiquement, identifiables comme une entité distincte. Les entités moléculaires en question sont soit des produits naturels, soit des produits de synthèse utilisés pour intervenir dans les processus des organismes vivants.[59]

4.3.2 La préparation des molécules de prédition

Pour pouvoir utiliser le modèle DL-VS, nous allons télécharger un autre ensemble de données de prédition regroupant des molécules présentant une interaction favorable avec la maladie COVID-19.

La détection de la structure 3D de la protéine responsable de la COVID-19 représente un défi majeur. Pour cette raison, l'utilisation de méthodes de virtual screening basées sur la similarité moléculaire s'avère être une stratégie efficace. Ainsi, nous envisageons de tester notre programme de virtual screening DL-VS sur cette maladie causée par le SARS-CoV-2. Ce processus pourrait potentiellement accélérer la découverte de composés thérapeutiques sans la nécessité de connaître la structure tridimensionnelle précise du virus.

La première étape consiste à rassembler une collection des molécules de référence. Ces molécules, présentent des propriétés prometteuses pour inhiber la progression de la maladie et atténuer ses symptômes. La figure 4.2 illustre les structures chimiques de quelques molécules de référence sélectionnées.

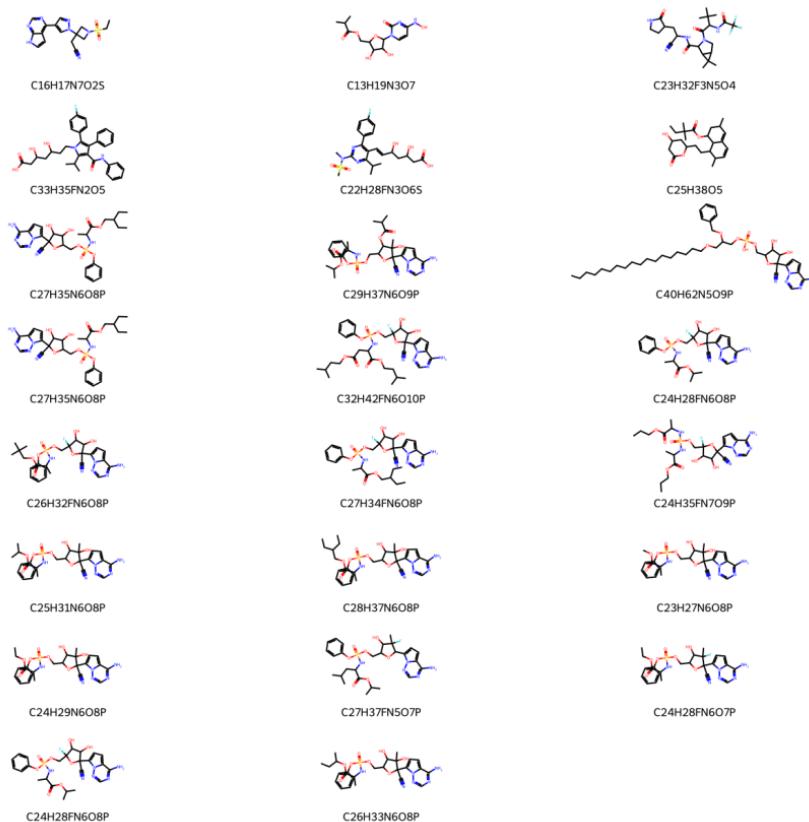


FIGURE 4.2 – La collection des molécules de référence

Après avoir rassemblé notre collection de molécules de référence, nous allons calculer les similarités (cosine/ Tanimoto) entre ces molécules et les molécules candidates, et calculer la valeur qui identifier les propriétés communs entre les molécules candidates et les molécules de référence, préparé notre jeu de données pour prédire les scores de similarité en utilisant notre modèle.

4.4 Résultats

4.4.1 Le temps d'exécution :

Pour illustrer la relation entre la taille des données et le temps d'exécution, nous avons varié la taille des données (molécules candidates) et calculée le temps d'exécutions. Nous avons exécuté notre programme sur deux machines DELL Latitude 8 cœur et sur une machine du CERIST dotée de 16 cœurs. La figure 4.3 montre la variation du temps d'exécution en fonction de la taille des données sur les deux machines.

Remarque : Même si les performances de la machine se sont améliorées, le temps d'exécution reste conséquent.

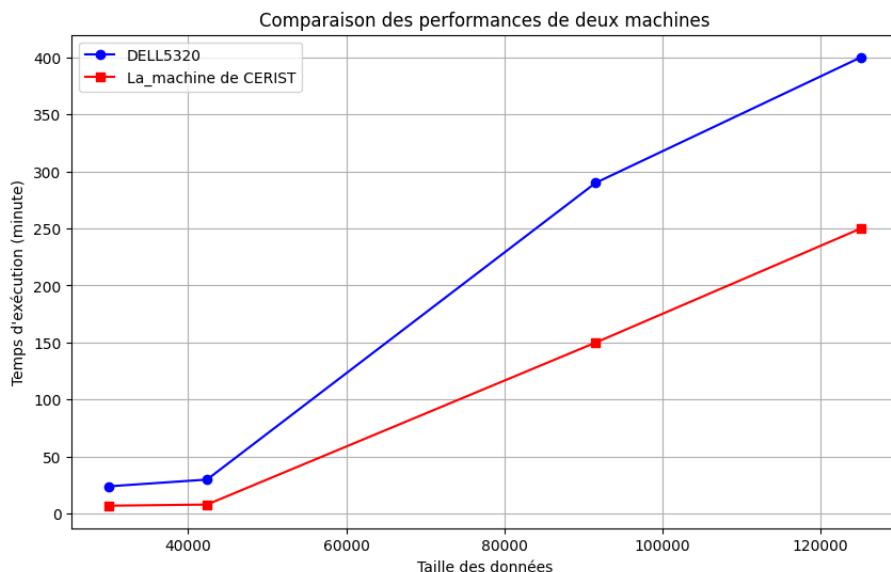


FIGURE 4.3 – Analyse Comparative des Performances : DELL Latitude 5320 vs La Machine de CERIST

4.4.2 L'évaluation du modèle :

Notre modèle est structuré pour une tache de régression. Utiliser la fonction Mean absolute error(MAE) est approprié pour les problèmes de régression, car elle mesure la différence moyenne entre les valeurs prédites et les valeurs réelles, ce qui donne une indication directe de la performance du modèle en termes d'erreur absolue. La figure 4.4 illustre les variation de l'erreur.

L'Erreur Absolue Moyenne (MAE), aussi connue sous le nom de Moyenne des Erreurs Absolues, est une mesure statistique utilisée pour évaluer la performance d'un modèle de prédiction. Elle permet de quantifier l'écart moyen entre les valeurs prédites

par le modèle et les valeurs réelles observées[60]. La formule (4.1) est la formule de calcul du MAE.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (4.1)$$

Où :

- n est le nombre de points de données.
- y_i représente la valeur cible réelle pour le point de données i.
- \hat{y}_i représente la valeur prédictive pour le point de données i.

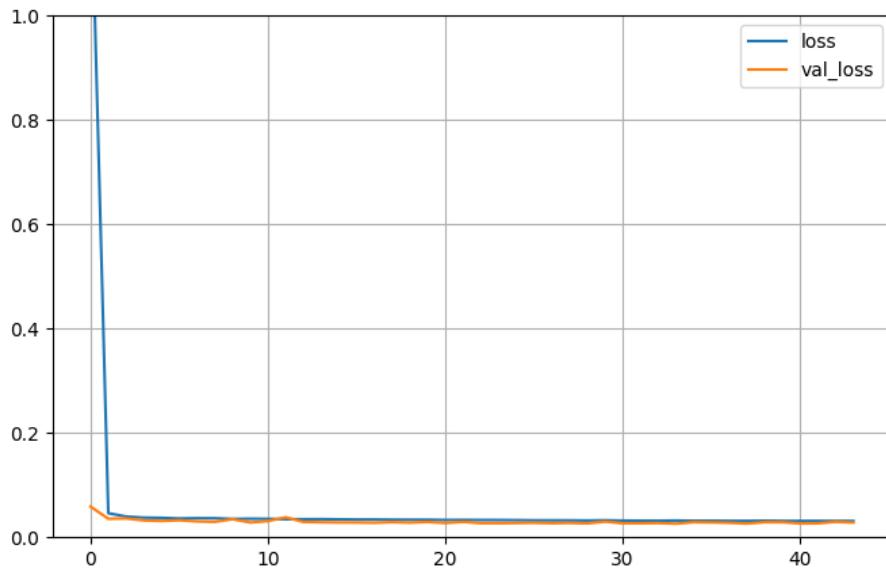


FIGURE 4.4 – L'évaluation du modèle (loss=MAE)

Discussion sur l'évaluation de l'erreur d'entraînement et l'erreur de validation

D'après le graphique 4.4, on observe que les deux courbes de perte convergent rapidement vers une valeur faible. Ceci indiquant que notre modèle apprend bien les données d'entraînement. Les courbes restent relativement stables après la convergence initiale, ce qui suggère que le modèle ne souffre ni de sur-apprentissage (overfitting) ni de sous-apprentissage (underfitting). Les valeurs de loss et de loss de validation sont également très faibles, ce qui confirme que le modèle est performant.

4.4.3 Prédition

La table 4.1 montre les molécules identifiées par notre programme de virtual screening comme des candidats prometteurs pour le traitement du COVID-19.

TABLE 4.1 – Les molécules identifiées par DL-VS pour covid-19

Formule chimique	Score de similarité
C27H35N6O8P	0.61
C33H47N6O8P	0.61
C28H37N6O8P	0.60
C27H36BrN6O8P	0.60
C25H29N6O8P	0.60
C27H33N6O8P	0.60
C26H33N6O8P	0.60
C24H29N6O8P	0.60
C22H25N6O8P	0.60
C25H31N6O8P	0.60
C32H31N6O8P	0.60
C43H65N6O9P	0.60
C23H26FN6O7P	0.60
C28H30FN6O7P	0.59
C27H38N5O8P	0.59
C30H33N6O8P	0.59
C27H37N6O6P	0.59
C21H23N6O8P	0.59

4.5 Validation des résultats

Notre objectif est d'évaluer la capacité de notre programme de virtual screening DL-VS à identifier des molécules prometteuses pour les tests biologique *in vitro* et *in vivo* tests pour le traitement du COVID-19. Afin de valider les activités biologiques des molécules prometteuses identifiées par DL-VS, nous nous appuyons sur la base de données PubChem. PubChem, une ressource incontournable dans le domaine de la recherche biomédicale, recense une vaste collection de données sur les molécules chimiques, incluant leurs propriétés, structures et interactions biologiques.[61] La simplicité d'utilisation de PubChem permet de rapidement identifier les informations pertinentes pour chaque molécule recommandée par notre programme DL-VS. Prenons l'exemple de la molécule C27H35N6O8P recommandée par notre programme pour les tests biologique pour le traitement du COVID-19. Une recherche rapide dans PubChem révèle que cette molécule a déjà fait l'objet d'études approfondies démontrant son activité antivirale contre le COVID-19. Ces résultats confirment le potentiel thérapeutique de la molécule C27H35N6O8P et renforcent la pertinence des recommandations de DL-VS. La figure 4.5 montre les activité antivirale de C27H35N6O8P contre le SCOVID-19

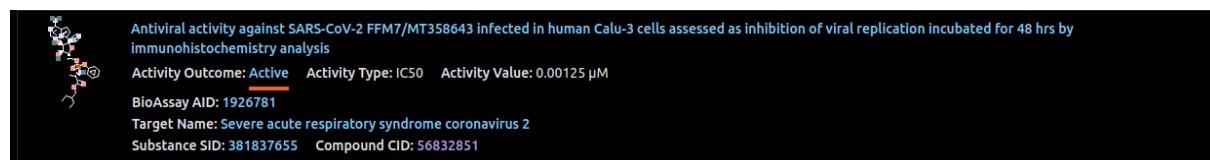


FIGURE 4.5 – Activité antivirale de C27H35N6O8P contre le COVID-19

Un autre exemple de la molécule C43H65N6O9P , également identifiée par DL-VS

comme prometteuse pour le traitement du COVID-19. Des études publiées dans le site de pubchem ont démontré que la molécule C43H65N6O9P inhibe efficacement la réPLICATION du SARS-CoV-2, le virus responsable du COVID-19.

Comme le montre la figure 4.6, la molécule C43H65N6O9P, ces résultats prometteurs suggèrent que cette molécule pourrait être un candidat thérapeutique efficace contre le COVID-19 et justifient des recherches plus approfondies.

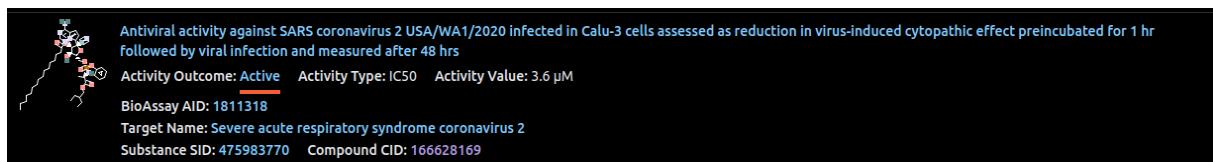


FIGURE 4.6 – Activité antivirale de C43H65N6O9P contre le COVID-19

La molécule C26H33N6O8P, également identifiée par DL-VS comme prometteuse pour le traitement du COVID-19. La figure 4.7 montre les activités biologiques de ce molécule avec le coronavirus, ces résultat remarquables renforcent le potentiel thérapeutique et confirment la capacité de DV-VS à identifier des candidats thérapeutiques prometteurs contre le COVID-19.



FIGURE 4.7 – Activité antivirale de C26H33N6O8P contre le COVID-19

4.6 Le problème de temps d'exécution

Notre programme, conçu pour virtual screening en vue de tests biologiques, rencontre un problème majeur d'augmentation exponentielle du temps d'exécution avec l'accroissement du nombre de molécules analysées. Par exemple, le traitement de 30 000 molécules prend 2h sur une machine doté de 8 coeurs CPU, tandis que 191 562 molécules nécessitent une journée entière.

Cette limitation entrave considérablement l'utilisation du programme à grande échelle, car il existe plus de 166 milliards de molécules dans la nature, et le criblage (screening) de toutes ces molécules dans un délai raisonnable est indispensable pour identifier de nouveaux candidats thérapeutiques.[28]

L'augmentation des caractéristiques de la machine n'a pas permis de résoudre le problème de manière significative.

Parmi les solutions envisagées, l'utilisation de techniques de parallélisme pour distribuer le traitement sur plusieurs processeurs(optimisation).

4.7 Optimisation

En informatique, le temps d'exécution est lié à la complexité du programme ou à la taille des données. Dans notre cas, il s'agit d'un jeu de données volumineux avec un grand nombre d'attributs (plus que 360 attributs) et un nombre important d'instances. Face à cette complexité, nous allons explorer l'optimisation et la minimisation du temps d'exécution en utilisant des méthodes de programmation parallèle, comme nous le verrons dans la section suivante.

Nous allons étudier ce qu'est la programmation parallèle sur des GPU, ses avantages et comment elle peut optimiser et réduire le temps d'exécution des programmes.

4.7.1 Programmation parallèle

La programmation parallèle exploite la puissance de plusieurs processeurs ou ordinateurs pour exécuter des tâches simultanément, réduisant ainsi le temps d'exécution global.[1][62] Dans le contexte de l'optimisation et de la programmation parallèle, nous allons définir les unités de traitement graphique (GPU). Au départ, les cartes graphiques étaient surtout utilisées pour la simulation 3D et aux jeux vidéo. Mais grâce à leur puissance et à leurs performances croissantes, ils sont désormais utilisés pour exécuter des algorithmes complexes, tels que ceux de l'intelligence artificielle, traitement d'image, manipulation et les opérations sur les matrices, calcul intensif ou l'analyse de données volumineuses.[1]

4.7.2 Définition des cartes graphique

Les cartes graphiques, également connues sous le nom de GPU, sont conçues selon une architecture parallèle. Elles intègrent de multiples unités de calcul, optimisées pour le traitement en flux, ce qui se traduit par une capacité accrue à traiter une plus grande quantité d'informations par unité de temps.[62]

Les GPU sont composés d'unités de traitement parallèle appelées multiprocesseurs. Chaque multiprocesseur contient plusieurs grilles, et chaque grille est divisée en plusieurs blocs. À l'intérieur de chaque bloc, on trouve 32 threads, qui sont des unités d'exécution parallèle. Les threads sont assignés à des portions spécifiques de données et exécutent en parallèle des traitements sur une partie[1]. La figure 4.8 illustre l'architecture générale des GPU

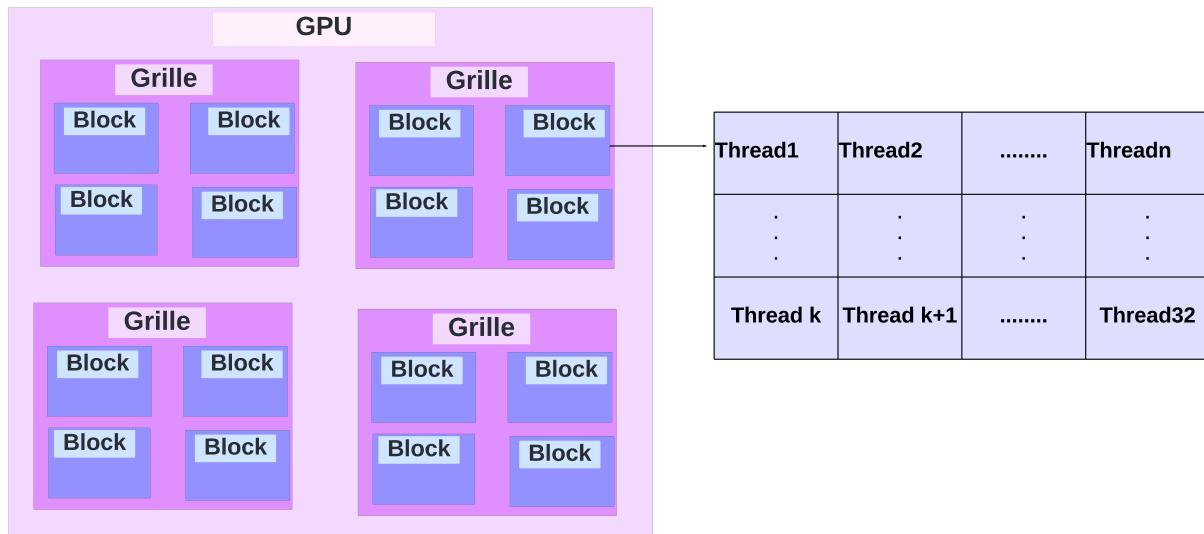


FIGURE 4.8 – Architecture générale des GPU

4.7.3 Fonctionnement des GPU

Pour pouvoir exécuter des programmes sur GPU, il est nécessaire d’identifier les parties du programme qui peuvent être exécutées en parallèle sur le GPU [1]. Une fois les parties parallélisables identifiées, le code du programme doit être divisé en deux parties distinctes :

- **Code CPU** : Cette partie du code s’exécute sur le CPU et gère les tâches séquentielles et les interactions avec l’environnement externe [63].
- **Code GPU Kernel (noyau)** : Cette partie du code, appelée Kernel "noyau", il est lancé sur le GPU par le CPU. Elle contient les instructions parallélisées qui seront exécutées par les threads du GPU.[63][64]

Lorsque le CPU appelle le kernel, le GPU lance l’exécution de ce dernier, les threads sont assignés à des blocs et à des grilles. Chaque thread exécute une instruction sur un ensemble de données spécifique.[64] Les threads d’un même bloc peuvent accéder à la mémoire partagée [63], ce qui leur permet de partager des données entre eux. Les threads de différents blocs accèdent à la mémoire globale, qui est partagée par tous les threads du GPU [63] [1]. La figure 4.9 illustre l’organisation des mémoires dans un GPU. Cette organisation permettent d’optimiser les performances des calculs parallèles.

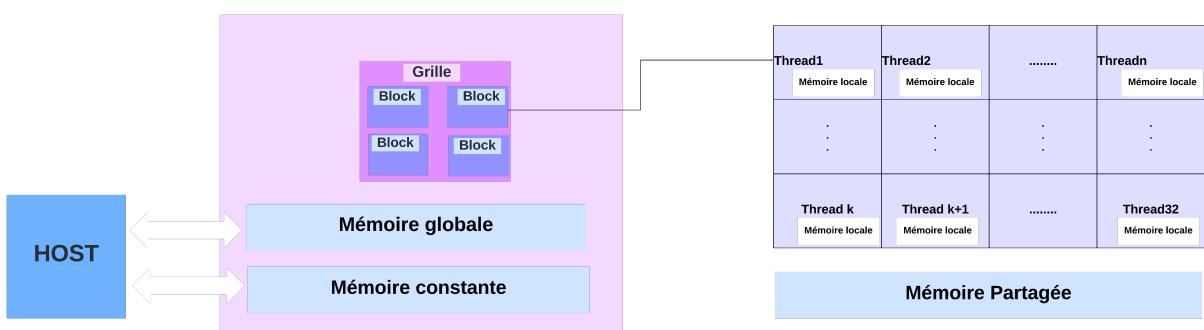


FIGURE 4.9 – Les différents niveaux de mémoire

4.8 Programmation cuda

4.8.1 Définition

Cuda¹ est une plate-forme informatique parallèle développée par NVIDIA.[65] C'est une langage de programmation permettant d'exploiter la puissance de calcul combinée du processeur central (CPU) et du processeur graphique (GPU) au sein d'un même programme.[66] Dans ce modèle, le CPU est appelé hôte et le GPU est appelé Device. CUDA est particulièrement populaire pour le calcul haute performance sur GPU grâce à sa facilité d'utilisation et à ses performances élevées.[67]

4.8.2 PyCUDA

PyCUDA est une bibliothèque logicielle Python qui offre un accès simple à l'API de calcul parallèle CUDA de Nvidia. Elle permet aux développeurs d'exploiter la puissance des GPU NVIDIA pour accélérer des applications scientifiques et de calcul intensif.[68] PyCUDA se distingue des autres wrappers de l'API CUDA par sa gestion automatique de la mémoire et des ressources, ce qui simplifie considérablement le développement d'applications CUDA. Elle offre également un certain nombre de fonctionnalités avancées, telles que la compilation de code CUDA à partir de code Python et l'exécution de noyaux CUDA sur plusieurs GPU.[69]

4.9 Le pseudo-code de kernel GPU pour DL-VS

Le kernel CUDA nous aide à paralléliser le calcul de similarité entre un vecteur finger-print de candidat et les fingerprints de 120 molécules de référence. Il prend en entrée la matrice qui regroupe les fingerprints des 120 molécules de référence et un vecteur finger-print de candidat. Chaque thread CUDA se positionne sur un fingerprint spécifique parmi les 120 et calcule la similarité avec le fingerprint candidat. Le résultat est enregistré dans un vecteur de sortie à la case correspondant à indice **idx**. L'indice global **idx** de chaque thread est calculé en fonction de l'indice du thread et de l'indice du bloc. Les variables pour les différentes mesures de similarité (cosinus, Jaccard/Tanimoto, intersection) sont initialisées avant le calcul de la similarité. Pour chaque élément de la ligne (déterminée par **idx**), la similarité est calculée en fonction de la métrique spécifiée (**metric**). Le résultat est stocké dans le tableau **result** en fonction de la métrique utilisée.

1. Cuda : CUDA est l'abréviation de Compute Unified Device Architecture

Algorithm 4 Compute Similarity Kernel CUDA

Require: $matrix$, $vector$, $result$, $rows$, $cols$, $metric$

```

1:  $idx \leftarrow \text{threadIdx.x} + \text{blockDim.x} * \text{blockIdx.x}$ 
2: if  $idx < rows$  then
3:    $similarity \leftarrow 0.0$ 
4:    $norm\_vector \leftarrow 0.0$ 
5:    $norm\_row \leftarrow 0.0$ 
6:    $intersection \leftarrow 0.0$ 
7:    $union\_val \leftarrow 0.0$ 
8:   for  $j \leftarrow 0$  to  $cols - 1$  do
9:      $val \leftarrow matrix[idx * cols + j]$ 
10:     $vec\_val \leftarrow vector[j]$ 
11:    if  $metric == 0$  then
12:       $similarity \leftarrow similarity + val * vec\_val$ 
13:       $norm\_vector \leftarrow norm\_vector + vec\_val * vec\_val$ 
14:       $norm\_row \leftarrow norm\_row + val * val$ 
15:    else if  $metric == 1$  then
16:       $intersection \leftarrow intersection + \min(val, vec\_val)$ 
17:       $union\_val \leftarrow union\_val + \max(val, vec\_val)$ 
18:    else if  $metric == 2$  then
19:       $similarity \leftarrow similarity + val * vec\_val$ 
20:    end if
21:   end for
22:   if  $metric == 0$  then
23:      $result[idx] \leftarrow similarity / (\sqrt{norm\_vector} * \sqrt{norm\_row})$ 
24:   else if  $metric == 1$  then
25:      $result[idx] \leftarrow intersection / union\_val$ 
26:   else if  $metric == 2$  then
27:      $result[idx] \leftarrow similarity$ 
28:   end if
29: end if

```

Pour pouvoir utiliser le kernel CUDA défini, il est nécessaire de préparer les données, lance le kernel CUDA, et récupère les résultats. Toutes ces fonctionnalités sont définies dans la fonction calculate_similarity_gpu. Voici un résumé de son fonctionnement :

- **Préparation des données** : La matrice et le vecteur sont convertis en tableaux NumPy de type float32. Un tableau pour les résultats est initialisé.
- **Allocation de mémoire sur le GPU** : La mémoire nécessaire pour la matrice, le vecteur et les résultats est allouée sur le GPU.
- **Transfert des données vers le GPU** : Les données de la matrice et du vecteur sont copiées de la mémoire du CPU vers la mémoire du GPU.
- **Configuration et lancement du kernel CUDA** : Le kernel CUDA compute_similarity est configuré avec une taille de bloc de 256 threads et est lancé pour calculer les similarités.

- **Récupération des résultats** : Les résultats calculés sur le GPU sont copiés de la mémoire du GPU vers la mémoire du CPU.

Algorithm 5 Calculate Similarity using GPU

Require: *matrix, vector, metric*

```
1: matrix  $\leftarrow$  np.array(matrix, dtype = np.float32)  
2: vector  $\leftarrow$  np.array(vector, dtype = np.float32)  
3: rows, cols  $\leftarrow$  matrix.shape  
4: result  $\leftarrow$  np.zeros(rows, dtype = np.float32)  
5: matrix_gpu  $\leftarrow$  cuda.mem_alloc(matrix.nbytes)  
6: vector_gpu  $\leftarrow$  cuda.mem_alloc(vector.nbytes)  
7: result_gpu  $\leftarrow$  cuda.mem_alloc(result.nbytes)  
8: cuda.memcpy_htod(matrix_gpu, matrix)  
9: cuda.memcpy_htod(vector_gpu, vector)  
10: cuda.memcpy_htod(result_gpu, result)  
11: func  $\leftarrow$  mod.getfunction("compute_similarity")  
12: block_size  $\leftarrow$  256  
13: grid_size  $\leftarrow$  (rows + block_size - 1)  
14: func(matrix_gpu, vector_gpu, result_gpu, np.int32(rows), np.int32(cols), np.int32(metric), block_size, 1, 1), grid = (grid_size, 1))  
15: cuda.memcpy_dtoh(result, result_gpu)  
     result
```

4.9.1 Définition des fonctions CUDA utilisées dans la fonction calculate_similarity_gpu

1. **cuda.mem_alloc** : Cette fonction alloue de la mémoire sur le GPU. Elle prend en argument la taille en octets de la mémoire à allouer et retourne un pointeur vers la mémoire allouée sur le GPU.
2. **cuda.memcpy_htod** : Cette fonction copie des données de la mémoire du CPU (Host) vers la mémoire du GPU (Device). htod signifie "Host to Device".

3. **cuda.memcpy_dtoh** : Cette fonction copie des données de la mémoire du GPU (Device) vers la mémoire du CPU (Host). **dtoh** signifie "Device to Host".
4. **mod.get_function** : Cette fonction récupère une référence à une fonction CUDA compilée (kernel) dans la fonction func. Elle prend en argument le nom de la fonction CUDA telle qu'elle a été définie dans le code du kernel.

4.10 Comparaison du temps d'exécution entre le GPU et CPU

Pour pouvoir comparer les performances, nous avons calculé le temps d'exécution de notre architecture DL-VS sur GPU et sur CPU. Nous avons modifié la taille des données d'une exécution à l'autre. Le graphique 4.10 compare le temps d'exécution (CPU) à une machine dotée de 16 coeurs et le temps d'exécution de la même machine sur une carte graphique RTX3060.

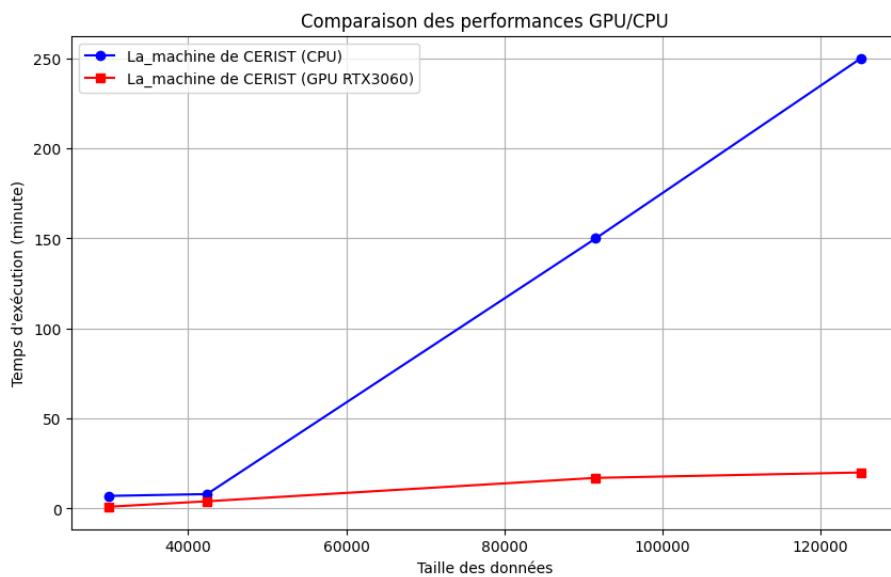


FIGURE 4.10 – Le temps d'exécution sur GPU et sur CPU

Le GPU montre une performance supérieure par rapport au CPU, avec des temps d'exécution beaucoup plus faibles pour le traitement des mêmes volumes de données.

Les deux configurations montrent une augmentation linéaire des temps d'exécution avec l'augmentation de la taille des données. Cependant, la pente est beaucoup plus raide pour le CPU que pour le GPU, indiquant une meilleure scalabilité du GPU pour des volumes de données importants. En conclusion, pour des tâches nécessitant un traitement de grandes quantités de données, l'utilisation du GPU est nettement plus avantageuse que l'utilisation du CPU seul. Cela démontre l'efficacité du GPU pour des applications intensives en calcul, offrant une réduction significative du temps de traitement. La figure 4.11 illustre l'accélération du calcul obtenue en comparant le temps d'exécution sur CPU et GPU. On observe clairement que l'accélération augmente avec la taille des données, démontrant la capacité du GPU à gérer efficacement des volumes de données importants. Cette supériorité s'explique par l'architecture parallèle du GPU, conçue pour traiter si-

multanément de multiples tâches, contrairement au CPU qui fonctionne de manière séquentielle.

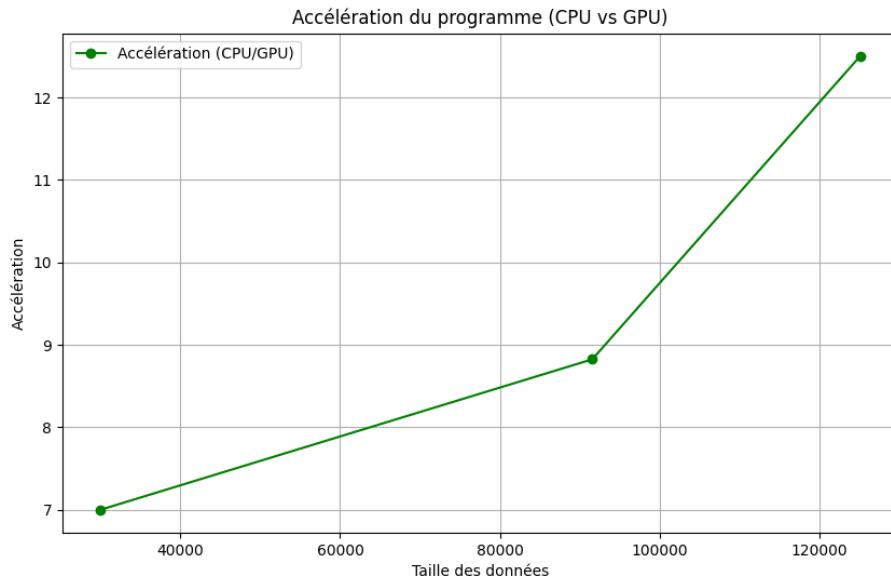


FIGURE 4.11 – Accélération du programme (CPU vs GPU)

4.11 Comparaison entre Autodock MPI-Vina et DL-VS

4.11.1 Autodock MPI-Vina

MPI-Vina est une version parallèle d'AutoDock-Vina² conçue pour accélérer considérablement les tâches de virtual screening sur les grappes de calcul haute performance. MPI-Vina utilise la norme Message Passing Interface MPI³ pour répartir la charge de travail sur plusieurs processeurs ou ordinateurs au sein d'un cluster. Il est optimisé pour tirer parti de la puissance de traitement parallèle offerte par les grappes de calcul, réduisant considérablement le temps nécessaire au criblage virtuel par rapport à un seul processeur.[71] Autodock MPI-Vina est un programme de virtual screening basé sur la structure utilisé pour identifier des ligands potentiels ayant une affinité élevée pour un récepteur cible. Il s'agit d'une méthode computationnelle qui permet de prédire l'interaction entre une molécule de ligand et un site de liaison protéique.[72]

4.11.2 Fonctionnement de Autodock MPI-Vina

Autodock MPI-Vina repose sur une application de docking moléculaire pour prédire l'interaction entre des ligands et un récepteur. Il permet d'estimer les activités biologiques des ligands en calculant leur affinité pour le récepteur, prédire la structure 3D du complexe ligand-récepteur en identifiant la conformation la plus stable de l'interaction. Les ligands avec les scores de liaison les plus bas (meilleure affinité) sont sélectionnés pour les tests biologique. L'introduction de MPI permet d'optimiser ce processus en minimisant le temps

2. AutoDock Vina : est un programme open-source pour simuler la liaison de petites molécules (ligands) à de plus grandes molécules (récepteurs).

3. MPI : est une norme de communication qui permet de distribuer les calculs sur plusieurs processeurs ou coeurs d'un ordinateur ou d'un cluster.[70]

d'exécution. Il utilise une approche maître-esclave, où le maître envoie la structure de la protéine réceptrice à tous les esclaves et distribue les ligands un par un. Les esclaves exécutent le processus de docking moléculaire sur chaque ligand et envoient les résultats au maître. Ces résultats comprennent le complexe formé entre le ligand et le récepteur, ainsi que les paramètres d'affinité de liaison, d'énergie de liaison et de site de liaison. Le maître regroupe ensuite tous les résultats des esclaves et sélectionne les complexes ayant la plus petite affinité de liaison pour les recommander pour des tests biologiques (in vivo test et in vitro test).

4.11.3 Comparaison

Avant de comparer les performances et le temps d'exécution des architectures DL-VS et Autodock MPI-Vina, nous commencerons par comparer leur fonctionnement et les techniques d'optimisation employées. Le tableau 4.2 présente une comparaison de l'architecture DL-VS et de l'architecture Autodock MPI-Vina.

TABLE 4.2 – Comparaison entre Autodock MPI-Vina et DL-VS

	Autodock MPI-Vina	DL-VS
Type de virtual screening	SBVS	LBVS
Fonctionnement	basé sur le docking moléculaire	basé sur le calcul de similarité et le deep learning
Optimisation	MPI	GPU

Temps d'exécution

Afin de comparer les performances des deux architectures en termes de temps d'exécution, nous analyserons les données des tableaux 4.3 et 4.4.

- **Autodock MPI-Vina :** La table 4.3 montre la variation de temps d'exécution par rapport au nombre des processus pour tester Autodock MPI-Vina sur 420 molécule.

TABLE 4.3 – La variation de temps d'exécution en fonction de nombre de processeurs pour Autodock MPI-Vina

Nombre de processeurs	Temps d'exécution (minutes)
2	207.296
4	107.998
6	71.386
10	39.578
12	32.416
14	25.068

- **DL-VS :** La table 4.4 montre la variation de temps d'exécution par rapport à la taille des données pour DL-VS

TABLE 4.4 – *La variation de temps d'exécution en fonction de la taille des données pour DL-VS*

Nombre des molécules	Temps d'exécution (minutes)
30000	4
42378	9
91562	16
125112	26

En comparant les tableaux 4.3 et 4.4, nous constatons que AutoDock MPI-Vina utilise un nombre limité de molécules candidates (420), tandis que DL-VS explore un espace beaucoup plus large (30 000 à 125 000 molécules). Malgré cela, le temps d'exécution des deux méthodes est presque identique. Cela suggère effectivement que DL-VS pourrait être plus performante en termes de temps d'exécution.

AutoDock MPI-Vina est plus traditionnel et utilise des méthodes de docking moléculaire classiques, tandis que DL-VS exploite les avantages du deep learning et de calcul de similarité.

4.12 Conclusion

DLVS est une architecture conçue pour améliorer considérablement les performances et la fiabilité du virtual screening. Elle repose sur plusieurs principes clés comme Puissance des modèles de Deep Learning, Combinaison de méthodes de similarité, l'exécution sur GPU et la programmation CUDA sont utilisées pour optimiser le processus de calcul. Les recommandations de DLVS sont validées par comparaison avec les données de PubChem. Cela garantit la fiabilité des résultats.

CONCLUSION GÉNÉRALE

Conclusion Générale

Le travail présenté dans ce mémoire se concentre sur les objectifs, les contributions majeures, les résultats obtenus et les perspectives futures de notre travail sur le virtual screening avec l'apprentissage profond.

Objectifs et Contributions

L'objectif principal de ce mémoire était de développer une architecture de virtual screening (VS) basée sur l'apprentissage profond afin d'améliorer le processus de découverte de médicaments. Nous avons proposé une architecture innovante appelée DL-VS, intégrant des modèles de deep learning pour le calcul de la similarité moléculaire et la prédiction de l'activité biologique des molécules.

Les contributions majeures de ce mémoire sont les suivantes :

- **Développement de l'architecture DL-VS** : Une architecture robuste et scalable pour le virtual screening utilisant des réseaux de neurones convolutifs (CNN) et des réseaux de neurones entièrement connectés.
- **Calcul de Similarité Multi-Métrique** : Intégration de plusieurs types de similarité (cosinus, Tanimoto) pour améliorer la précision des prédictions.
- **Optimisation des Performances** : Mise en œuvre d'algorithmes optimisés pour le traitement parallèle sur GPU, réduisant ainsi significativement le temps de calcul.

Résultats Obtenus

Les résultats de nos expérimentations montrent que l'architecture DL-VS est capable de prédire avec une haute précision les interactions moléculaires, surpassant les méthodes traditionnelles de virtual screening. Nous avons observé une amélioration notable dans les recommandations de molécules candidates pour des tests biologiques.

Perspectives Futures

Les travaux futurs pourraient se concentrer sur plusieurs axes pour améliorer et étendre l'architecture DL-VS :

CONCLUSION GÉNÉRALE

- **Amélioration de la Base de Données** : Enrichir la base de données avec des structures moléculaires plus diversifiées pour augmenter la robustesse du modèle.
- **Exploration de Nouveaux Modèles** : Expérimenter avec d'autres architectures de deep learning, comme les réseaux de neurones en graphes (GNN), pour capturer des relations plus complexes entre les molécules.
- **Utilisation de MapReduce** : Optimiser le processus de DL-VS sur un cluster de GPU en utilisant le paradigme MapReduce pour distribuer et paralléliser les calculs, réduisant ainsi davantage le temps de traitement.

En conclusion, ce mémoire a démontré le potentiel des techniques d'apprentissage profond et de HPC pour révolutionner le virtual screening et accélérer le processus de découverte de nouveaux médicaments. Les perspectives futures offrent de nombreuses opportunités pour continuer à améliorer cette approche et contribuer de manière significative à la recherche pharmaceutique.

BIBLIOGRAPHIE

- [1] Hocine SAADI. “Metaheuristics on GPU Graphics Processor : Application to Molecular Docking”. Thèse de doct. University Djillali Liabès of Sidi Bel-Abbès, Exact Sciences Faculty, Computer Science Department, EEDIS Laboratory, 2019-2020.
- [2] Talia B. KIMBER, Yonghui CHEN et Andrea VOLKAMER. “Deep Learning in Virtual Screening : Recent Applications and Developments”. In : *International Journal of Molecular Sciences* 22.4435 (2021). DOI : 10.3390/ijms22094435.
- [3] Dr. BOUBENIA MOHAMED. *BIG DATA (Outils et Ecosystème pour la bio-Informatique)*. Rapp. tech. USTHB, 2023.
- [4] Clément FOUCHER. “Méthodologie de conception pour la virtualisation et le déploiement d'applications parallèles sur plateforme reconfigurable matériellement”. Theses. Université Nice Sophia Antipolis, 2012. URL : <https://theses.hal.science/tel-00777511>.
- [5] S. ERICKSEN. *Scaling Virtual Screening to Ultra-Large Virtual Chemical Libraries*. Rapp. tech. OSG Virtual School Showcase, 2021.
- [6] Eduardo Habib Bechelane MAIA et al. “Structure-Based Virtual Screening : From Classical to Artificial Intelligence”. In : *Frontiers in Chemistry* 8.343 (2020). DOI : 10.3389/fchem.2020.00343.
- [7] Tiago Alves de OLIVEIRA et al. “Virtual Screening Algorithms in Drug Discovery : A Review Focused on Machine and Deep Learning Methods”. In : *Drugs Drug Candidates* (2023). DOI : 10.3390/ddc2020017.
- [8] Yaowen GU et al. “Employing Molecular Conformations for Ligand-Based Virtual Screening with Equivariant Graph Neural Network and Deep Multiple Instance Learning”. In : *Molecules* (2023).
- [9] Adrià CERETO-MASSAGUÉ et al. “Molecular fingerprint similarity search in virtual screening”. In : *Methods* 71 (2015). Virtual Screening, p. 58-63. ISSN : 1046-2023. DOI : <https://doi.org/10.1016/j.ymeth.2014.08.005>. URL : <https://www.sciencedirect.com/science/article/pii/S1046202314002631>.
- [10] Shengchao LIU et al. “Practical Model Selection for Prospective Virtual Screening”. In : *Journal of Chemical Information and Modeling* 59.282-293 (2019). DOI : 10.1021/acs.jcim.8b00363.

BIBLIOGRAPHIE

- [11] Maritza HERNANDEZ et AL. “A Quantum-Inspired Method for Three-Dimensional Ligand-Based Virtual Screening”. In : (2019). DOI : <https://www.researchgate.net/publication/330845050>.
- [12] Thomas G. KRISTENSEN et AL. “METHODES DE CRIBLAGE VIRTUEL BASÉ SUR LA SIMILARITÉ”. In : (2013). DOI : <https://doi.org/10.5936/csbj.201302009>.
- [13] René de GELDER. “Prédiction des cocristaux par les réseaux de neurones artificiels”. In : (2020). DOI : <https://doi.org/10.1002/ange.202009467>.
- [14] Hochschule DÜSSELDORF. “Analysis of Biological Screening Data and Molecular Selectivity Profiles Using Fingerprints and Mapping Algorithms”. In : (2008). DOI : [10.13140/RG.2.1.1085.9680](https://doi.org/10.13140/RG.2.1.1085.9680).
- [15] BA SOZIALARBEIT-PÄDAGOGIK. “Analyse des données de criblage biologique et des profils de sélectivité moléculaire à l'aide d'empreintes digitales et d'algorithmes de cartographie”. Thèse de doct. 2008. DOI : [10.13140/RG.2.1.1085.9680](https://doi.org/10.13140/RG.2.1.1085.9680).
- [16] DEMIR et AL. “ToDD : Topological Compound Fingerprinting in Computer-Aided Drug Discovery”. In : *Advances in Neural Information Processing Systems*. Sous la dir. de S. KOYEJO et al. T. 35. Curran Associates, Inc., 2022, p. 27978-27993. URL : https://proceedings.neurips.cc/paper_files/paper/2022/file/b31f6d65f2584b3c4347148db36fe07f-Paper-Conference.pdf.
- [17] Jingjing WANG et al. “MIFNN : Molecular Information Feature Extraction and Fusion Deep Neural Network for Screening Potential Drugs”. In : *Current Issues in Molecular Biology* 44.11 (2022), p. 5638-5654. ISSN : 1467-3045. DOI : [10.3390/cimb44110382](https://doi.org/10.3390/cimb44110382). URL : <https://www.mdpi.com/1467-3045/44/11/382>.
- [18] Carlos GARCIA-HERNANDEZ, Alberto FERNANDEZ et Francesc SERRATOSA. “Ligand-Based Virtual Screening Using Graph Edit Distance as Molecular Similarity Measure”. In : *Journal of Chemical Information and Modeling* 59.1410-1421 (2019). DOI : [10.1021/acs.jcim.8b00820](https://doi.org/10.1021/acs.jcim.8b00820).
- [19] MIRANDA-QUINTANA et AL. In : *Informatique moléculaire* 40 (2021), p. 2060017.
- [20] Daniel PROBST et Jean-Louis REYMOND. “A probabilistic molecular fingerprint for big data settings”. In : *Journal of cheminformatics* 10 (2018), p. 1-12.
- [21] Fouaz BERRHAIL, Hacene BELHADEF et Mohammed HADDAD. “Deep Convolutional Neural Network to Improve the Performances of Screening Process in Ligand-Based Virtual Screening”. In : *Expert Systems with Applications* (2022). DOI : [10.1016/j.eswa.2022.117287](https://doi.org/10.1016/j.eswa.2022.117287).
- [22] Naomie ALTALIB Mohammed Khaldoon et Salim. “Écran virtuel basé sur la similarité utilisant des méthodes d'apprentissage en profondeur siamoises améliorées”. In : *ACS oméga* 7 () .
- [23] *Criblage virtuel basé sur QSAR : avancées et applications dans la découverte de médicaments*. <https://www.frontiersin.org/journals/pharmacology/articles/10.3389/fphar.2018.01275/full>.
- [24] Paul D LYNE. “Criblage virtuel basé sur la structure : un aperçu”. In : *Découverte de médicaments aujourd'hui* 7 () .
- [25] Claudia MENDOZA-BARRERA et AL. “Amarrage protéine-protéine et protéine-ligand”. In : (2013).

- [26] Vishal KUMAR Neeraj et Acharya. “Avances dans les approches de criblage virtuel basées sur l'intelligence artificielle pour le big data”. In : *Revues de recherche médicale* 44 () .
- [27] Ryo KUNIMOTO, Jürgen BAJORATH et Kazumasa AOKI. “From traditional to data-driven medicinal chemistry : A case study”. In : *Drug Discovery Today* 27.8 (2022). ISSN : 1359-6446. DOI : <https://doi.org/10.1016/j.drudis.2022.04.017>.
- [28] Horacio et all BANEGAS-LUNA. “Une revue des outils Web de criblage virtuel basés sur des ligands et des algorithmes de criblage dans de grandes bases de données moléculaires à l'ère du big data”. In : *Future chimie médicinale* 10 () .
- [29] Nithin BUDUMA, Nikhil BUDUMA et Joe PAPA. *Fundamentals of deep learning*. 2022. URL : <https://books.google.com/books?hl=en&lr=&id=2e5vEAAAQBAJ&oi=fnd&pg=PT5&dq=Fundamentals+of+deep+learning&ots=3ZI3uHkOHq&sig=CrLabF5EF7h4rAEtmzvNezq3mPs>.
- [30] Ian GOODFELLOW, Yoshua BENGIO et Aaron COURVILLE. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [31] Luiz Anastacio et al ALVES. *Les réseaux de neurones graphiques comme outil potentiel pour améliorer les programmes de dépistage virtuel*.
- [32] *Réseaux de neurones artificiels pour l'apprentissage automatique – Tous les aspects que vous devez connaître*. <https://data-flair.training/blogs/artificial-neural-networks-for-machine-learning/>.
- [33] Wenxiao YANG. *Deep Learning*. Rapp. tech. Department of Mathematics, University of Illinois at Urbana-Champaign, 2022.
- [34] SEJAL JAISWAL. *Multilayer Perceptrons in Machine Learning : A Comprehensive Guide*. <https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning>. Accessed : April 17, 2024.
- [35] Suresh YADAM, Phani Kumar SINGAMSETTY et Sabitha YADAM. “Nngeomvs : A Geometric Based Multi-Feature Integrated Deep Neural Network Classifier for Ligand-Similarity Based Virtual Screening”. In : *Available at SSRN* 4717859 () .
- [36] M. DAI, M. F. DEMIREL, Y. LIANG et al. “Représentez graphiquement les réseaux de neurones pour une prédiction précise et interprétable des propriétés des matériaux polycristallins”. In : *npj Comput Mater* 7 (2021), p. 103. DOI : 10.1038/s41524-021-00574-w. URL : <https://doi.org/10.1038/s41524-021-00574-w>.
- [37] C. CHEN, Y. ZUO, W. YE et al. “Apprentissage des propriétés de matériaux ordonnés et désordonnés à partir de données multi-fidélité”. In : *Nat Comput Sci* 1 (2021), p. 46-53. DOI : 10.1038/s43588-020-00002-x. URL : <https://doi.org/10.1038/s43588-020-00002-x>.
- [38] Nicholas HALLIWELL. “Évaluer et améliorer la qualité des explications de la prédiction des liens du réseau neuronal des graphiques sur les graphiques de connaissances”. Thèse de doct. URL : <https://theses.hal.science/tel-03907696>.
- [39] Rafiqul Zaman JABBAR H et Khan. “Méthodes pour éviter le sur-ajustement et le sous-ajustement dans l'apprentissage automatique supervisé (étude comparative)”. In : *Dispositifs informatiques, de communication et d'instrumentation* 70 () .

- [40] Xue YING. “An Overview of Overfitting and its Solutions”. In : *Journal of Physics : Conference Series* 1168.2 (fév. 2019), p. 022022. DOI : 10.1088/1742-6596/1168/2/022022. URL : <https://dx.doi.org/10.1088/1742-6596/1168/2/022022>.
- [41] Vinay Kumar et all KOLLURI Johnson et Kotte. “Réduire le problème de surajustement dans l’apprentissage automatique à l’aide d’une nouvelle méthode de régularisation L1/4”. In : *2020 4e Conférence internationale sur les tendances de l’électronique et de l’informatique (ICOEI)*(48184).
- [42] *Concepts de surajustement et de sous-ajustement dans l’apprentissage automatique.* <https://ezako.com/en/overfitting-and-underfitting-concepts-in-machine-learning/>.
- [43] *Sous-apprentissage et surapprentissage dans l’apprentissage automatique.* <https://www.baeldung.com/cs/ml-underfitting-overfitting>.
- [44] Francesc SERRATOSA. “Graph edit distance : Restrictions to be a metric”. In : *Pattern Recognition* 90.250-256 (2019). ISSN : 0031-3203. DOI : <https://doi.org/10.1016/j.patcog.2019.01.043>. URL : <https://www.sciencedirect.com/science/article/pii/S0031320319300639>.
- [45] Benoit PLAYE. “Approches d’apprentissage automatique pour le dépistage virtuel des médicaments”. Thèse de doct. URL : <https://pastel.hal.science/tel-02186833>.
- [46] Lewis MARTIN. “Amarrage itératif de pointe avec régression logistique et empreintes digitales Morgan”. In : (2021).
- [47] Chunyang WU et al. “Improving Interpretability and Regularization in Deep Learning”. In : *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.2 (2018), p. 256-265. DOI : 10.1109/TASLP.2017.2774919.
- [48] Documentation de RDKit. <https://www.rdkit.org/docs/>.
- [49] Documentation de Python. <https://docs.python.org/>.
- [50] Documentation de Visual Studio Code. <https://code.visualstudio.com/docs>.
- [51] Documentation de Pandas. <https://pandas.pydata.org/docs/>.
- [52] Documentation de NumPy. <https://docs.scipy.org/>.
- [53] Documentation de Matplotlib. <https://readthedocs.org/projects/matplotlib/>.
- [54] Tutoriel de RDKit. <https://docs.readthedocs.io/en/stable/config-file-v2.html>.
- [55] Documentation de scikit-learn. <https://scikit-learn.org/0.21/documentation.html>.
- [56] About - PubChem - National Institutes of Health (NIH). <https://pubchem.ncbi.nlm.nih.gov/docs/about/>.
- [57] Human Metabolome Database (HMDB). <https://hmdb.ca/>.
- [58] What is ChEMBL ?/CHEMBL. <https://www.ebi.ac.uk/training/online/courses/chembl-quick-tour/what-is-chembl/>.
- [59] CHEBI/À propos de ChEBI. <https://www.ebi.ac.uk/chebi/aboutChebiForward.do>.

BIBLIOGRAPHIE

- [60] M Waqar AHMED. <https://medium.com/@m.waqar.ahmed/understanding-mean-absolute-error-mae-in-regression-a-practical-guide-26e80ebb97df>. 2023.
- [61] <https://pubchem.ncbi.nlm.nih.gov/>.
- [62] Cen CHEN et al. “GFlink : An In-Memory Computing Architecture on Heterogeneous CPU-GPU Clusters for Big Data”. In : *IEEE Transactions on Parallel and Distributed Systems* 29.6 (2018), p. 1275-1288. DOI : [10.1109/TPDS.2018.2794343](https://doi.org/10.1109/TPDS.2018.2794343).
- [63] H. SAADI, N. Nouali TABOUDJEMAT, A. RAHMOUN et al. “Parallélisation efficace basée sur GPU du calcul de solvatation pour le problème d’amarrage aveugle”. In : *Journal of Supercomputing* 76.1980–1998 (2020). DOI : [10.1007/s11227-019-02834-5](https://doi.org/10.1007/s11227-019-02834-5). URL : <https://doi.org/10.1007/s11227-019-02834-5>.
- [64] John NICKOLLS et William J. DALLY. “The GPU Computing Era”. In : *IEEE Micro* 30.2 (2010), p. 56-69. DOI : [10.1109/MM.2010.41](https://doi.org/10.1109/MM.2010.41).
- [65] Shane COOK. *Programmation CUDA : guide du développeur sur le calcul parallèle avec GPU*. 2012.
- [66] David LUEBKE. “CUDA : Scalable parallel programming for high-performance scientific computing”. In : *2008 5th IEEE International Symposium on Biomedical Imaging : From Nano to Macro*. 2008, p. 836-838. DOI : [10.1109/ISBI.2008.4541126](https://doi.org/10.1109/ISBI.2008.4541126).
- [67] *Qu'est-ce que CUDA ?* <https://support.nvidia.eu/hc/fr/articles/4850516229266-Qu-est-ce-que-CUDA>.
- [68] Andreas KLÖCKNER et al. “PyCUDA and PyOpenCL : A scripting-based approach to GPU run-time code generation”. In : *Parallel Computing* 38.3 (2012), p. 157-174. ISSN : 0167-8191. DOI : <https://doi.org/10.1016/j.parco.2011.09.001>. URL : <https://www.sciencedirect.com/science/article/pii/S0167819111001281>.
- [69] <https://pypi.org/project/pycuda/>.
- [70] Jack J WALKER David W et Dongarra. “MPI : une interface standard de transmission de messages”. In : *Superordinateur* 12 (), p. 56-68.
- [71] Md Mokarrom HOSSAIN. *MPI-Vina : MPI based parallel implementation of Auto-dock Vina*. Rapp. tech. 2014.
- [72] Mohammad Mahdi JAGHOORI, Boris BLEIJLEVENS et Silvia D OLABARRIAGA. “1001 Ways to run AutoDock Vina for virtual screening”. In : *Journal of computer-aided molecular design* 30.237–249 (2016).