Validating and Benchmarking Deep Learning Model for Protein-Ligand Binding Affinity Prediction

by

Aparna C

Under the Supervision of Dr. N Arul Murugan

Indraprastha Institute of Information Technology Delhi

May 2023

Validating and Benchmarking Deep Learning model for Protein-Ligand Binding Affinity Prediction

by

Aparna C

Submitted

in partial fulfillment of the requirements for the degree of

Master of Technology

to

Indraprastha Institute of Information Technology Delhi

May 2023

Certificate

This is to certify that the thesis titled "**Validating and Benchmarking Deep Learning Model for Protein-Ligand Binding Affinity Prediction**" being submitted by

Aparna C to the Indraprastha Institute of Information Technology Delhi for the award of the Master of Technology is an original research work carried out by her under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

May, 2023

Dr. N. Arul Murugan
Department of Computational Biology
Indraprastha Institute of Information Technology Delhi
New Delhi 110 020

Acknowledgements

I would like to express my sincere gratitude to my thesis supervisor, Dr. N Arul Murugan, for his valuable mentorship and constant support throughout the work. The completion of this thesis would not have been possible without his expert knowledge and encouragement.

I would like to thank all the faculty and staff at IIIT, Delhi, for their consistent assistance throughout the entire process.

I am thankful to my fellow batchmates for their encouragement and support. I also thank Prateek Paul from Ph.D. 2022 batch for his valuable contribution to this work.

Finally, I would like to thank my family, who have been my constant source of strength and inspiration, and without them, this academic journey would not have been possible.

Abstract

Drug discovery is a complex, time-consuming, and challenging process, but the introduction of computational methods, especially machine learning, and deep learning models, has accelerated the entire process. OnionNet is a deep Convolutional Neural Network model that showed promising results in predicting the protein-ligand binding affinities based on the structural features of protein-ligand complexes. As the performance of any machine learning/deep learning model depends on the quality of the datasets used, we need to validate and benchmark the model using various datasets to ensure its accuracy and reliability. The OnionNet model is trained using the PDBbind v2016 dataset. The prediction power of the model is evaluated with the PDBbind v2016 core set, the BAPPL dataset, and the BindingDB dataset. The performance of OnionNet is impressive on the PDBbindv2016 dataset and BAPPL dataset, which have high-quality experimentally determined 3D protein-ligand structures. The performance of the model is below average on the BindingDB dataset with 3D structures of ligands generated by the program Vconf. Still, it is better when compared to the performance of AutoDock and AutoDock Vina. The performance of the OnionNet, AutoDock, and AutoDock Vina in finding the best ligand binding pose is also evaluated, and AutoDock Vina outperformed the others.

Keywords: OnionNet, Deep Learning, PDB, PDBbind, BindingDB, BAPPL Dataset, AutoDock, AutoDock Vina, Drug Discovery

Contents

List of Figures

CHAPTER 1

1 Introduction

1.1 Drug Discovery

The drug discovery process is very time-consuming and highly expensive process that involves the identification of a potential target involved in a disease and finding a possible drug candidate that interacts with the target from a vast chemical space. The drug candidate should have high binding affinity and high binding specificity along with the optimum ADMET (Absorption, Distribution, Metabolism, Excretion, Toxicity) properties. Over the years, the development in the field of Computer Science and Computational methods has helped in reducing the time and cost of drug prediction. Computational drug discovery is an interdisciplinary field that combines the principles of chemistry, biology, and computer science to accelerate the drug discovery process. Several computational methods are used in drug discovery, like molecular docking, virtual screening, ligand-based drug design, structure-based design, machine learning, and deep learning, which helped to develop new drugs for a wide range of diseases. Among these, Machine Learning and Deep Learning approaches have made a significant impact on the drug discovery field as they are faster, can be combined with other computational methods and can deal with large datasets.

1.2 Drug-like Properties

Drug design mainly aims at optimizing the drug-like properties of the ligand like binding affinity, binding specificity, ADMET properties, solubility in water and lipids, and bioavailability.

1.2.1 Binding Affinity

Binding Affinity measures how effectively a ligand binds to its target molecule.

$\Delta G = RT\ln Ki$ where R is the Universal Gas Constant (8.314 JK^{-1} mol^{-1}),

T is the Temperature in Kelvin,

and Ki is the Inhibition Constant

1.2.2 Binding Specificity

Binding Specificity is the ability of a ligand molecule to attach specifically to the target while avoiding the off-targets

1.2.3 ADMET Properties

The other important drug properties are absorption, distribution, metabolism, excretion, and toxicity, collectively called ADMET properties. Solubility in water and lipids and bioavailability are other important drug properties.

1.3 Computational Methods

1.3.1 Virtual screening

The drug discovery and development process start with identifying a target molecule associated with a particular disease. Following this, a ligand that can effectively interact with this specific target in the human body to prevent or cure the disease is discovered. Virtual screening is an important computational method in drug discovery that helps identify small molecules that can bind to the target from a large chemical library. This method is fast and efficient compared to High Throughput Screening (HTS). There are mainly two types of virtual screening methods: Ligand-based virtual screening and Structure-based virtual screening[1].

1.3.1.1 Ligand-based virtual screening

Ligand-based virtual screening is based on the similarity property principle, which states that similar compounds have similar properties[2]. This method identifies small compounds from a chemical library that may bind to the target or receptor based on their similarities to a reference compound (that is already known to bind to the target). The similarity between ligands can be calculated from their 1D descriptors (fragment counts),2D descriptors (topological descriptors),3D descriptors (geometrical descriptors), and Quantum chemical descriptors. Ligand-based virtual screening is computationally cheap when compared to structure-based virtual screening. This method is used when the 3D structure of the target protein/enzyme is not available or difficult to determine[3].

1.3.1.2 Structure-based virtual screening

We have seen that ligand-based virtual screening is computationally cheap when compared to Structure-based virtual screening, but it has certain disadvantages. The similarity property principle that similar compounds have similar properties may not hold true when compounds are bound to a target molecule. The activity of similar ligands may be different even when bound to the same target protein. In such

scenarios, we need to consider the protein structure also. Structure-based virtual screening involves using the 3D structure of the receptor (obtained using X-ray Crystallography or NMR spectroscopy) to identify potential ligands. Ligands can be docked with the 3D structure of the receptor to find the best binding pose (the most stable configuration of a ligand within the binding site of the receptor), and binding energy can be calculated and, based on this ligand with the best binding affinity can be identified. There are different scoring functions for calculating the binding energy of the docked protein-ligand compound, and they are proportional to the binding affinity of the ligand to the target protein. The scoring functions are mainly classified into physics-based, knowledge-based and empirical and machine learning based[4].

a) Physics-based

The physics-based scoring function is also known as the force-field-based scoring function. Here the binding energies are calculated as the sum of the various interactions between the ligand and the protein like van der Waal's, hydrogen bonding, electrostatic, solvation, and entropy.

b) Knowledge-based

The knowledge-based scoring function is based on the comparison of atom pairs interaction in the protein-ligand complexes whose three-dimensional structures are available. The protein-ligand complexes are ranked based on the potential mean force calculated from the frequency of appearance of different atom-atom contacts.

c) Empirical scoring function

The empirical scoring function calculates the binding affinity based on a weighted sum of terms that defines various interactions between protein and ligand like van der Waal's, electrostatic, hydrogen bonding, and solvation[5].

d) Machine Learning based scoring function

The machine learning-based scoring function predicts the protein-ligand binding affinity by analyzing and learning from the data (known protein-ligand complexes) on which the model is trained. The model uses machine learning algorithms like Support Vector Machine, Random Forest, etc., and the dataset consists of molecular descriptors(features) which describe the physical and chemical properties of the protein-ligand complexes. During training, the model learns the relationship between the molecular descriptors and the experimentally calculated binding affinities. The trained model can predict the binding affinity of new protein-ligand complexes based on their molecular descriptors[6].

The deep learning-based scoring function, which is a subset of machine learning-based scoring, is widely used as it can learn from complex data and can more accurately predict the protein-ligand binding affinity. It uses deep neural networks to learn the relationship between the input features of the protein-ligand complexes and their binding affinities. A Convolutional Neural Network (CNN) is a deep learning-based model which can interpret the relationship between the 3D structures of protein-ligand complexes and their binding affinities. A Graph Neural Network (GNN) models the protein and ligand structures as graphs and learns the association between graphs and binding affinities[7].

1.4 Scope of the current study

The Machine Learning and Deep Learning scoring functions have proven to be effective in predicting the protein-ligand binding affinity, and their accuracy is high when compared to the traditional empirical scoring functions. The major drawback with Machine Learning and Deep Learning models is that their accuracy depends on the quality and diversity of the data on which they are applied. Therefore, it is very important that such models need to be validated with different experimental data to ensure their accuracy, reliability, and consistency. The OnionNet is a deep convolutional neural network that has shown promising results in predicting the protein-ligand binding affinity. This model is trained on a large dataset of protein-ligand

complexes with known binding affinities and it can predict binding affinities for new protein-ligand complexes based on their structural features. Though the model's accuracy is high, as mentioned earlier, we must benchmark and validate the model with various datasets to ensure its reliability. To validate and benchmark the model we have to evaluate the performance of the model in predicting the binding affinity (energetics) and in identifying the best binding poses (structural analysis).

CHAPTER 2

2 Materials and Methods

2.1 OnionNet

OnionNet is a deep convolutional neural network developed for protein-ligand binding affinity prediction. CNNs are powerful compared to traditional machine learning approaches as they can identify complex patterns from large datasets. This model uses an onion-like structure to learn the intermolecular interactions between the ligand and protein atoms and also groups local and non-local interactions as per the distance range. Once trained, it predicts the negative logarithms (pKa) of the dissociation constants (Kd) or half inhibition concentrations (IC50) or inhibition concentrations (Ki)[8].

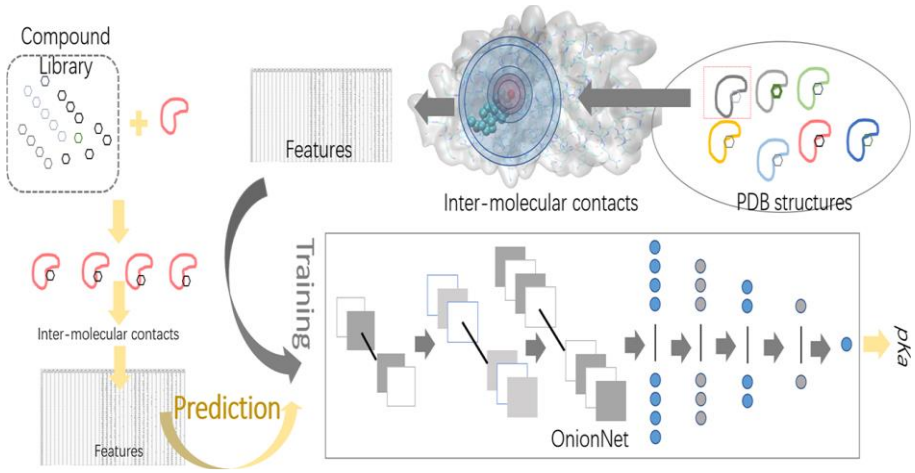


Fig 1: OnionNet

(OnionNet: a Multiple-Layer Intermolecular-Contact-Based Convolutional Neural Network for Protein–Ligand Binding Affinity Prediction, Liangzhen Zheng, Jingrong Fan, and Yuguang Mu, *ACS Omega* **2019** *4* (14), 15956-15965, DOI: 10.1021/acsomega.9b01997)

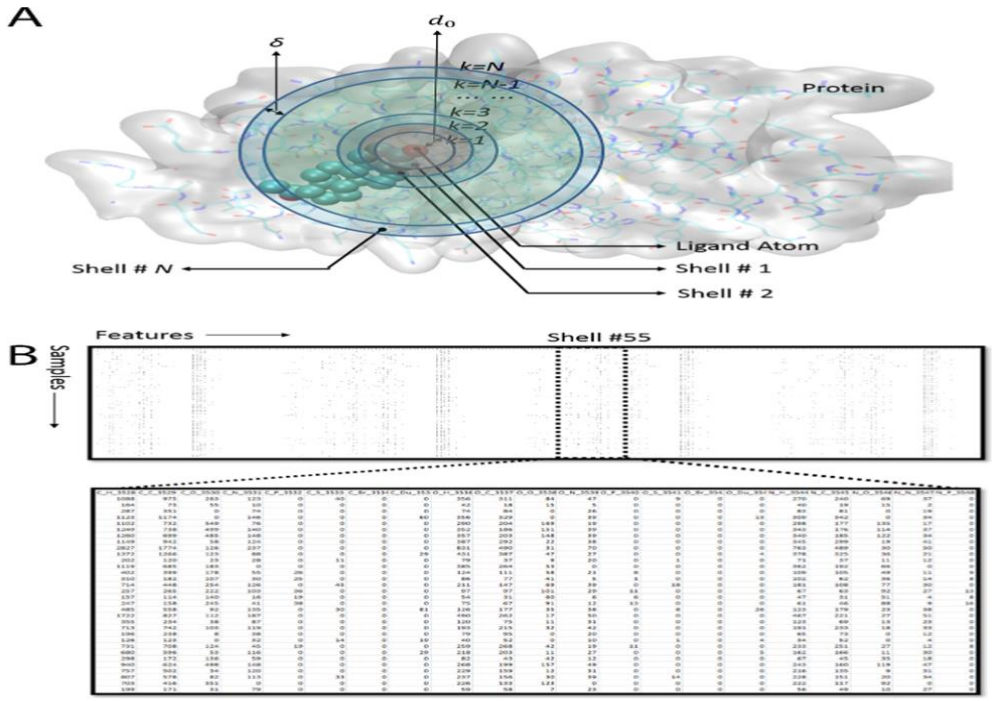


Fig 2: Featurization of Protein-Ligand Complexes

(OnionNet: a Multiple-Layer Intermolecular-Contact-Based Convolutional Neural Network for Protein–Ligand Binding Affinity Prediction, Liangzhen Zheng, Jingrong Fan, and Yuguang Mu, ACS Omega 2019 4 (14), 15956-15965, DOI: 10.1021/acsomega.9b01997)

The OnionNet model is trained using the protein-ligand 3D structures and binding affinities from the PDBbind database. For each of the protein-ligand complexes, the features are generated based on the intermolecular interaction between the protein-ligand atoms. In the feature extraction process, we consider each ligand atom as the center and define N number of shells around it with a thickness of δ. The element pair-specific contact numbers are calculated for the ligand atoms and protein atoms in each shell. In this model, eight elements (C, N, O, H, P, S, HAX, and Du) are considered for calculating the protein-ligand atom pair-specific contacts. Here, HAX denotes any of these halogen atoms- Cl, F, Br, I. Du stands for Dummy, representing all other elements. Therefore, for each shell, there are 64 features representing intermolecular interaction between the protein and ligand. The distance between the ligand atom and

the first shell (k=1) is defined as d0, and the minimal distance from the ligand atom to any other shell is calculated as (k-1) $*$ δ + d0 where 2 ≤ k ≤N. The contact between atom in protein (r) and atom in ligand(l) is 1 if the distance dr,l between them is within range of (k-2) $*$ δ + d0 ≤ dr,l < (k-1)$*$ δ + d0, otherwise 0. N is defined as 60 shells with d0 = 1.0 Å and δ =0.5 Å, resulting in 3840 features. Pearson Correlation Coefficient (R), Root Mean Square Error (RMSE) and MAE (Mean Absolute Error) are used to evaluate the performance of OnionNet. R measures the strength and direction of the linear relationship between two variables (Here pKa_{true} and $pKa_{predicted}$). It is calculated as the covariance between pKa_{true} and $pKa_{predicted}$ divided by the product of their standard deviations. RMSE measures the difference between predicted values($pKa_{predicted}$) and actual values(pKa_{true}). It is calculated as the square root of the average of the square of the difference between the predicted and actual values. MAE is the average of absolute difference between predicted values and actual values.

2.2 Dataset for OnionNet Training, Validation and Testing

2.2.1 PDBbind

PDBbind is a database that is curated from Protein Data Bank, and it contains the structure of biomolecular complexes along with their experimentally measured binding affinity data (like Ki, Kd, or IC50) [9]. The dataset maintains binding data for protein-ligand, protein-protein, protein-nucleic acid and nucleic acid-ligand complexes. In our study, we are using only the structural information and binding affinity data of protein-ligand complexes. Each protein-ligand complex consists of a protein molecule (in the PDB format) and ligand molecule (in the mol2 or sdf format). OnionNet is trained and tested with PDBbindv2016 dataset.

2.2.1.1 PDBbind Dataset Preparation

The PDBbindv2016 consists of three sets of data- general set, refined set, and core set. The general set, which forms the main content of the PDBbind dataset, has all the complexes with experimentally measured binding data. The refined set is a collection of better-quality protein-ligand complexes from the general set. A core set is a set of

high-quality protein-ligand complexes which are used for benchmarking various docking and scoring functions. 285 protein-ligand complexes in the core set are extracted and kept as the testing set. The protein-ligand complexes that are common in both the refined set and core set are removed from the refined set, and 1132 complexes are randomly selected for the validation set. The remaining 11866 complexes in the general set (after removing the complexes present in validation and test sets) are used for training. The binding data (Ki/Kd/IC50) for each complex is extracted from the Index_general_PL_data.2016 and converted into pKa = $-\log_{10} K_x$, where K_x represents Ki, Kd, or IC50.

Protein Preparation-The protein files are in the pdb format, and all the water molecules and ion are removed from the protein.

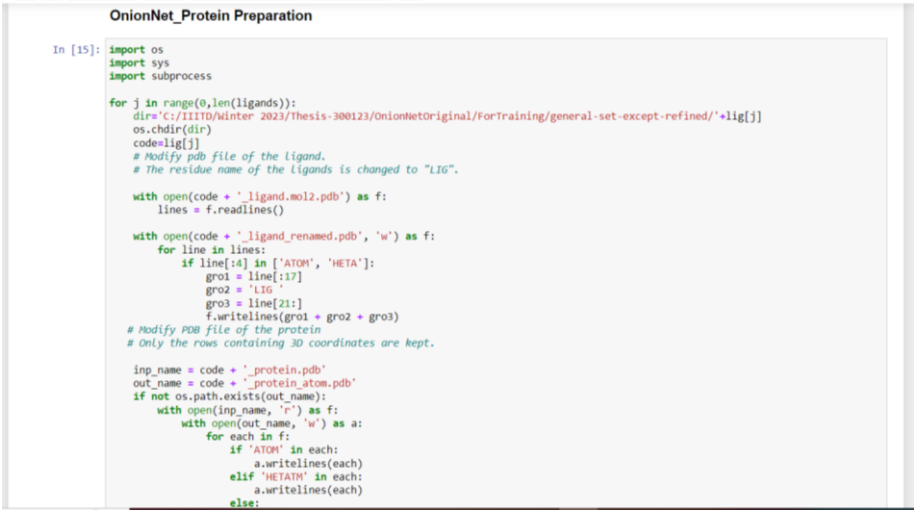


Fig 3: OnionNet-Protein Preparation

Ligand Preparation- The ligand files in the mol2 format are converted to pdb format using the OpenBabel software. The name of the ligand in all the ligand files are changed to ‘LIG’.

OnionNet_DatasetPrep_Mol_to_PDB

In [ ]:

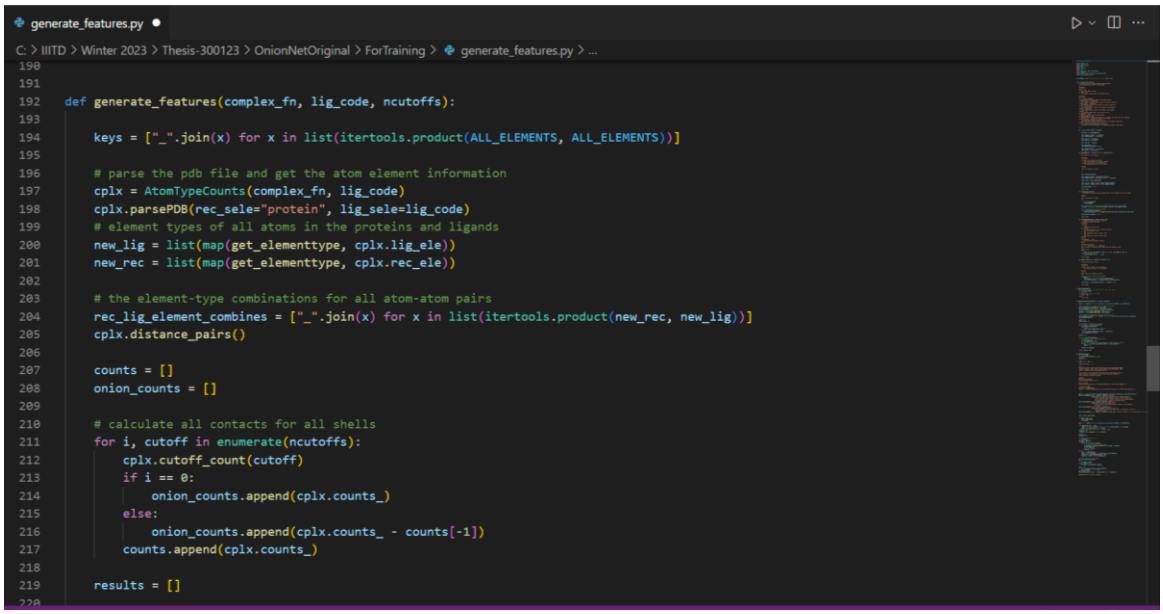
```
import openbabel
import os
import sys
import subprocess

with open("generalset_except_refined_pl_complx_names.txt") as f:          #ligand.txt contains all the complex names
    lines=f.readlines()
lig=[]
for i in range(0,len(lines)):
    lig.append(lines[i].strip())
    #print(lig[i])

for j in range(0,len(lig)):
    dir='C:/IIITD/Winter 2023/Thesis-300123/OnionNetOriginal/ForTraining/general-set-except-refined/'+lig[j]
    os.chdir(dir)
    #for f in os.listdir(dir):
        #print(f)
    #Command to convert mol2 file to pdb file using OpenBabel
    cmd='obabel'+' ' +lig[j]+'_ligand.mol2' + ' '+'-O'+' ' +lig[j] +'_ligand.mol2.pdb'
    print(cmd)
    subprocess.call(cmd)
```

Fig 4: Ligand Preparation

2.2.1.2 Training, Validation and Testing

The protein-ligand files are formed by combining the processed pdb files of proteins and ligands. These files are then used to generate features for training and testing OnionNet using the following command. The file input.dat contains the filenames of all the protein-ligand complexes.

python generate_features.py -inp input.dat -out features.csv

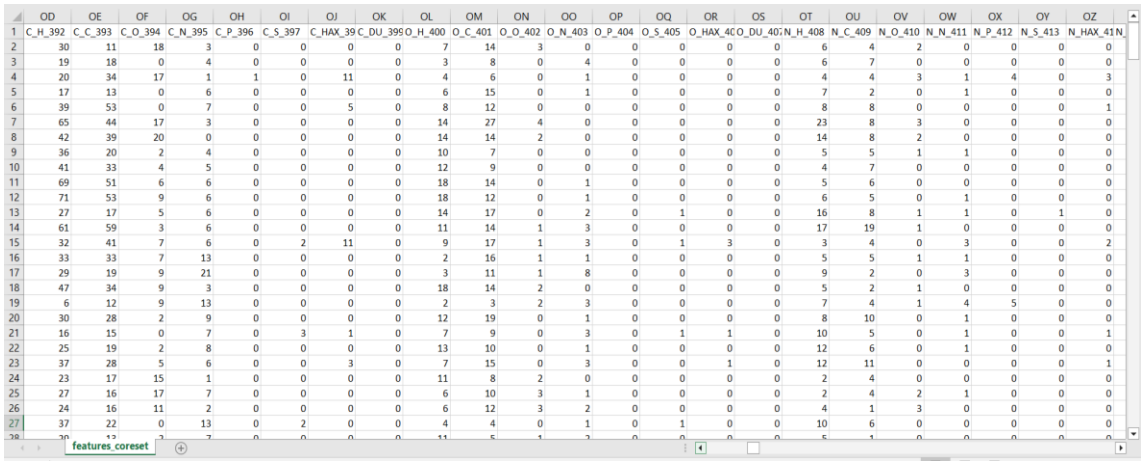


generate_features.py

C: > IIITD > Winter 2023 > Thesis-300123 > OnionNetOriginal > ForTraining > generate_features.py > ...

```
def generate_features(complex_fn, lig_code, ncutoffs):

    keys = ["_".join(x) for x in list(itertools.product(ALL_ELEMENTS, ALL_ELEMENTS))]

    # parse the pdb file and get the atom element information
    cplx = AtomTypeCounts(complex_fn, lig_code)
    cplx.parsePDB(rec_sele="protein", lig_sele=lig_code)
    # element types of all atoms in the proteins and ligands
    new_lig = list(map(get_elementtype, cplx.lig_ele))
    new_rec = list(map(get_elementtype, cplx.rec_ele))

    # the element-type combinations for all atom-atom pairs
    rec_lig_element_combines = ["_".join(x) for x in list(itertools.product(new_rec, new_lig))]
    cplx.distance_pairs()

    counts = []
    onion_counts = []

    # calculate all contacts for all shells
    for i, cutoff in enumerate(ncutoffs):
        cplx.cutoff_count(cutoff)
        if i == 0:
            onion_counts.append(cplx.counts_)
        else:
            onion_counts.append(cplx.counts_ - counts[-1])
        counts.append(cplx.counts_)

    results = []
```

Fig 5: generate_features.py



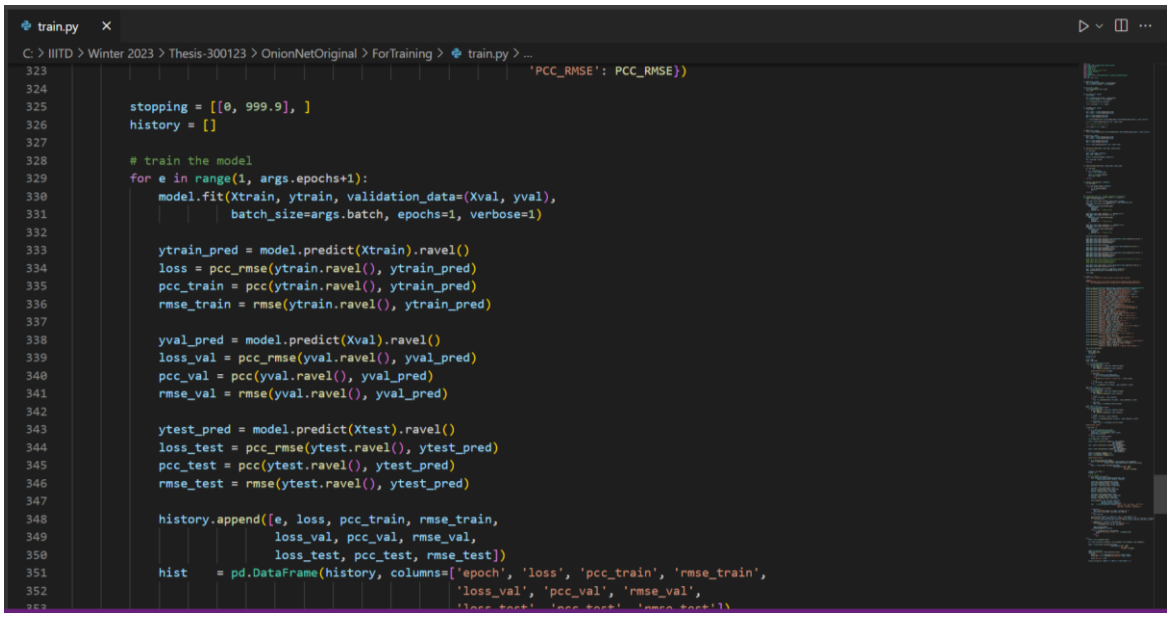
	OD	OE	OF	OG	OH	OI	OJ	OK	OL	OM	ON	OO	OP	OQ	OR	OS	OT	OU	OV	OW	OX	OY	OZ
1	C_H_392	C_C_393	C_O_394	C_N_395	C_P_396	C_S_397	C_HAX_39	C_DU_399	O_H_400	O_C_401	O_O_402	O_N_403	O_P_404	O_S_405	O_HAX_4(	O_DU_407	N_H_408	N_C_409	N_O_410	N_N_411	N_P_412	N_S_413	N_HAX_41N
2	30	11	18	3	0	0	0	0	7	14	3	0	0	0	0	0	6	4	2	0	0	0	0
3	19	18	0	4	0	0	0	0	3	8	0	4	0	0	0	0	6	7	0	0	0	0	0
4	20	34	17	1	1	0	11	0	4	6	0	1	0	0	0	0	4	4	3	1	4	0	3
5	17	13	0	6	0	0	0	0	6	15	0	1	0	0	0	0	7	2	0	1	0	0	0
6	39	53	0	7	0	0	5	0	8	12	0	0	0	0	0	0	8	8	0	0	0	0	1
7	65	44	17	3	0	0	0	0	14	27	4	0	0	0	0	0	23	8	3	0	0	0	0
8	42	39	20	0	0	0	0	0	14	14	2	0	0	0	0	0	14	8	2	0	0	0	0
9	36	20	2	4	0	0	0	0	10	7	0	0	0	0	0	0	5	5	1	1	0	0	0
10	41	33	4	5	0	0	0	0	12	9	0	0	0	0	0	0	4	7	0	0	0	0	0
11	69	51	6	6	0	0	0	0	18	14	0	1	0	0	0	0	5	6	0	0	0	0	0
12	71	53	9	6	0	0	0	0	18	12	0	1	0	0	0	0	6	5	0	1	0	0	0
13	27	17	5	6	0	0	0	0	14	17	0	2	0	1	0	0	16	8	1	1	0	1	0
14	61	59	3	6	0	0	0	0	11	14	1	3	0	0	0	0	17	19	1	0	0	0	0
15	32	41	7	6	0	2	11	0	9	17	1	3	0	1	3	0	3	4	0	3	0	0	2
16	33	33	7	13	0	0	0	0	2	16	1	1	0	0	0	0	5	5	1	1	0	0	0
17	29	19	9	21	0	0	0	0	3	11	1	8	0	0	0	0	9	2	0	3	0	0	0
18	47	34	9	3	0	0	0	0	18	14	2	0	0	0	0	0	5	2	1	0	0	0	0
19	6	12	9	13	0	0	0	0	2	3	2	3	0	0	0	0	7	4	1	4	5	0	0
20	30	28	2	9	0	0	0	0	12	19	0	1	0	0	0	0	8	10	0	1	0	0	0
21	16	15	0	7	0	3	1	0	7	9	0	3	0	1	1	0	10	5	0	1	0	0	1
22	25	19	2	8	0	0	0	0	13	10	0	1	0	0	0	0	12	6	0	1	0	0	0
23	37	28	5	6	0	0	3	0	7	15	0	3	0	0	1	0	12	11	0	0	0	0	1
24	23	17	15	1	0	0	0	0	11	8	2	0	0	0	0	0	2	4	0	0	0	0	0
25	27	16	17	7	0	0	0	0	6	10	3	1	0	0	0	0	2	4	2	1	0	0	0
26	24	16	11	2	0	0	0	0	6	12	3	2	0	0	0	0	4	1	3	0	0	0	0
27	37	22	0	13	0	2	0	0	4	4	0	1	0	1	0	0	10	6	0	0	0	0	0

features_coreset

Fig 6: Features generated

The OnionNet model is trained using the features, and the output files DNN_Model.h5 and StandardScaler.model are generated, which are used to predict the binding affinity of any new protein-ligand complex.

python train.py -fn_train features_train.csv -fn_validate features_validate.csv -fn_test features_test.csv -y_col pKa train 1



train.py

C: > IIITD > Winter 2023 > Thesis-300123 > OnionNetOriginal > ForTraining > train.py > ...

```
                                                    'PCC_RMSE': PCC_RMSE})

    stopping = [[0, 999.9], ]
    history = []

    # train the model
    for e in range(1, args.epochs+1):
        model.fit(Xtrain, ytrain, validation_data=(Xval, yval),
                  batch_size=args.batch, epochs=1, verbose=1)

        ytrain_pred = model.predict(Xtrain).ravel()
        loss = pcc_rmse(ytrain.ravel(), ytrain_pred)
        pcc_train = pcc(ytrain.ravel(), ytrain_pred)
        rmse_train = rmse(ytrain.ravel(), ytrain_pred)

        yval_pred = model.predict(Xval).ravel()
        loss_val = pcc_rmse(yval.ravel(), yval_pred)
        pcc_val = pcc(yval.ravel(), yval_pred)
        rmse_val = rmse(yval.ravel(), yval_pred)

        ytest_pred = model.predict(Xtest).ravel()
        loss_test = pcc_rmse(ytest.ravel(), ytest_pred)
        pcc_test = pcc(ytest.ravel(), ytest_pred)
        rmse_test = rmse(ytest.ravel(), ytest_pred)

        history.append([e, loss, pcc_train, rmse_train,
                        loss_val, pcc_val, rmse_val,
                        loss_test, pcc_test, rmse_test])
        hist    = pd.DataFrame(history, columns=['epoch', 'loss', 'pcc_train', 'rmse_train',
                                                 'loss_val', 'pcc_val', 'rmse_val',
                                                 'loss_test', 'pcc_test', 'rmse_test'])
```

Fig 7: train.py

python predict.py -fn features.csv -scaler StandardScaler.model -weights DNN_Model.h5 -out pKA_predicted.csv

predict.py

C: > IIITD > Winter 2023 > Thesis-300123 > OnionNetOriginal > ForTraining > alpha0.8 > predict.py > ...

```
    parser.add_argument("-weights", type=str, default="DNN_Model.h5",
                        help="Output. The trained DNN model file to save. ")
    parser.add_argument("-out", type=str, default="predicted_pKa.csv",
                        help="Output. The predicted pKa values file name to save. ")

    args = parser.parse_args()

    if len(sys.argv) < 3:
        parser.print_help()
        sys.exit(0)

    scaler = joblib.load(args.scaler)

    Xtest = None

    if os.path.exists(args.fn):
        df = pd.read_csv(args.fn, index_col=0, header=0).dropna()

        if 'pKa' in df.columns:
            ytest = df['pKa'].values.ravel()
            df = df.drop(['pKa'], axis=1)
        else:
            ytest = None

        Xs = scaler.transform(df.values)
        Xs = pd.DataFrame(Xs)
        Xs.index = df.index
        Xs.columns = df.columns
    else:
        print("File not exist: ", args.fn)
```

Fig 8: predict.py

PDB	pKa_predicted
4trc	3.374
4i7j	3.38
5a81	5.393
1jq8	5.399
4n9c	3.873
3th9	8.011
2rd6	7.214
4ghi	7.693
1k22	8.659
1g46	8.425
3zln	7.677
4axd	4.468
1sqo	4.805
2xyt	7.745
2za5	7.142
5bv3	5.632
4afg	7.125
2pqb	5.854
3gy3	6.922
1ndw	6.755
3k99	5.141
3f48	6.794
3ml5	7.972
4dsy	6.273
2vvs	6.464
1f74	5.598

predicted_pKa_refinedset_180223

Fig 9: Predicted pKa (Binding Affinity)

2.3 Dataset for OnionNet Validation- based on energetics (Binding Affinity) prediction

2.3.1 Validating OnionNet using BAPPL Dataset

BAPPL Dataset is a set of 161 heterogeneous non-metallo protein-ligand complexes taken from the protein data bank (RCSB). This set comprises of 55 unique proteins like trypsin, alpha thrombin, etc. [10].

BAPPL Dataset (scfbio-iitd.res.in)

2.3.1.1 Dataset Preparation and Validation

161 protein-ligand complexes in the dataset are downloaded along with their experimental binding free energies (kcal/mol). The ligand name in all the complexes is changed to 'LIG'. The experimental binding energies are converted to their corresponding pKa values using the following formulas.

Ki =exp(-ΔG/RT) where ΔG is the binding free energy, R is the Universal Gas Constant (8.314 $JK^{-1}mol^{-1}$), and T is the temperature in Kelvin.

$pKa= -\log_{10}(Ki)$.

BAPPLDataset- Converting Experimental Binding Energy to pKa

In [ ]:
```
import os
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
import math

df=pd.read_csv("ExpBindingEnergy.csv")
g=df.iloc[0]['BindingAffinity(kcal/mol)']
Ki=[]
R=0.001985 #Universal Gas Constant
T=298.15  #Temperature
for j in range(len(df.index)):
    g=df.iloc[j]['BindingAffinity(kcal/mol)']
    Ki.append(np.exp(g/(R*T)))
df["Ki"]=Ki
pKa=[]
for j in range(len(df.index)):
    k=df.iloc[j]["Ki"]
    pKa.append(-(math.log(k,10)))
df["pKa"]=pKa
df.to_csv("ExpBindingEnergy_pKa.csv")
```

Fig 10: BAPPL Dataset- Converting Experimental Binding Energy to Binding Affinity

Features are generated from the protein-ligand complexes using the 'generate_features.py' and given as input to the OnionNet to predict the pKa values.

2.3.2 Validating OnionNet using BindingDB Dataset

BindingDB is a public database that is a compilation of protein-ligand complexes along with their binding affinity data. The data are mainly extracted from scientific literature and are curated to ensure accuracy and reliability. The database also includes information related to key experimental conditions like temperature, buffer composition, and pH. BindingDB consists of SD files with 2D compound structures and 3D compound structures, where 3D structures are computed by the program Vconf (www.verachem.com/products/vconf). Unlike PDBbind, BindingDB provides affinities for protein-ligand complexes even if there are no crystal structures available[11].

2.3.2.1 Dataset Preparation

We have downloaded 3D sd file of 18 ligands associated with Beta Secretase-1 (Bace-1) protein from BindingDB and PDB file for Bace-1 (3ufl) from PDB. From the sd files we have extracted the IC50 values for each ligand which is converted to pKa values. Then the sd file is converted into pdb file and further splitted the file into 18 pdb files (18 ligands) using OpenBabel. The ligand name in each ligand file is changed to 'LIG'. We created 18 protein-ligand complexes by merging the Bace-1 pdb file with each of the 18 ligands (pdb) files. These files are then used to generate features. In this case, features are not generated as the protein and ligand in each complex appeared to be in different space frames when visualized using PyMOL.

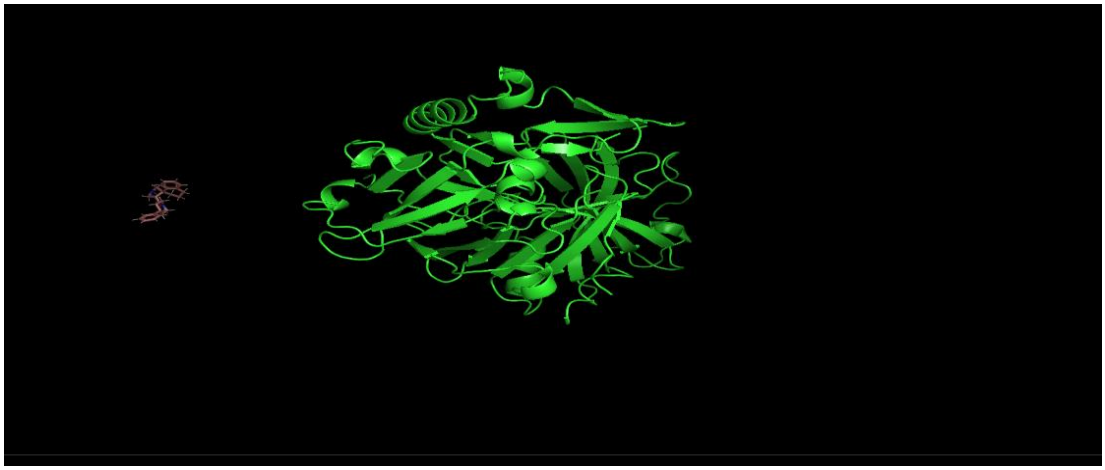


Fig 11: Visualizing Protein and Ligand using PyMOL

Here we have first used AutoDock Vina to find the 20 best binding poses of each ligand with Bace-1 protein. We have downloaded AutoDock Vina and the bace-1 protein file is loaded into the AutoDock Vina. All the water molecules and heteroatoms are deleted from the protein as they can interfere with docking and then polar hydrogen and Kollman charges are added. The protein file is then saved as a pdbqt file. In the case of ligands sd files are converted to pdb and then to pdbqt for docking. For each ligand and protein, we found the best grid parameters using grid box and generated a config.txt file which gives information of the active binding site in the protein.

"C:\vina.exe" –receptor protein.pdbqt -- ligand ligand1.pdbqt –config config.txt –log log.txt – out output.pdbqt

The output.pdbqt contains the details of all the 20 best poses, and log.txt contains information regarding the binding energy in each pose. We split the output.pdbqt to 20 pdbqt files and then they are converted to 20 pdb files. These 20 ligand poses are combined with the Bace-1 pdb file to form 20 protein-ligand complexes (for each ligand). Features are generated for all the 20 protein-ligand complexes, and OnionNet used these features to predict their binding affinities (pKa values). Out of these 20 complexes the protein-ligand (ligand pose) complex with best binding affinity is selected. This process is done for all the 18 ligands.

Similarly, we have used AutoDock4 to generate the best 20 ligand poses for each of the ligand with Bace-1 protein. A shell script 'dock.sh' is used to automate protein-ligand preparation and docking. Output.dlg file with details of 20 ligand poses is

obtained which is split into pdbqt files and then into pdb files and we then repeated the entire process mentioned above to validate OnionNet with the AutoDock4 generated ligand poses.

As a part of this validation, we also assessed the performance of AutoDock4. AutoDock4 generated the best 20 poses for each of the 18 ligands. In each case, we selected the best binding pose with the least binding energy. This binding energy value is converted to pKa (binding affinity) and compared with the experimental values obtained from BindingDB.

Ki =exp(-ΔG/RT) where ΔG is the binding free energy, R is the Universal Gas Constant (8.314 $JK^{-1}mol^{-1}$), and T is the temperature in Kelvin.

pKa= -log10(Ki)

Similarly, we have compared the best binding pose energies for each protein-ligand complex from AutoDock Vina with the experimental values from BindingDB.

2.4 Dataset for OnionNet Validation – based on structural analysis (Binding Pose)

Here we try to find whether the best binding affinity leads to the best binding pose of a ligand. The binding pose is the orientation of a ligand molecule within the binding pocket of a protein, and it depends on the interaction of the ligand with the protein. The determination of the best binding pose of a ligand is very important in computational drug discovery as it helps in designing better drugs with improved binding affinity and binding specificity. The accuracy of a predicted binding pose can be evaluated by comparing it with experimental data obtained from protein-ligand structures. Here we have compared the performance of OnionNet, AutoDock4, and AutoDock Vina in identifying the best ligand binding poses.

2.4.1 Dataset and Dataset Preparation

2.4.1.1 Protein Data Bank

The PDB database(RCSB.org) consists of 3D structures of macromolecules, mainly proteins that experimental methods have determined, such as X-ray crystallography, NMR spectroscopy, and electron microscopy[12].

BACE-1 (β-site APP cleaving enzyme I) plays a key role in the generation of β-amyloid, which are the main constituents of the amyloid plaques found in the brains of Alzheimer's disease patients. Due to its central role in the pathogenesis of Alzheimer's disease, BACE-1 has emerged as an important target for drug discovery efforts aimed at treating Alzheimer's disease. 403 BACE-1 (human) protein-ligand complexes are downloaded from the PDB database[13].

2.4.1.2 Dataset Preparation

After removing complexes with no proper ligand molecules (i.e., if only ions are present), there are 379 protein-ligand complexes. Each protein-ligand complex file (pdb file) is split into protein and ligand files. If there are multiple chain ids present in the protein file, only one chain id is selected, and the structural details corresponding to that chain id are retained in the file. Similarly, only the coordinates of the ligand molecule on the same chain id are kept in the ligand file. If conformers of residues or atoms are present in protein files and ligand files respectively, only one of the orientations is kept. Each of the protein and ligand files is combined together to form the protein-ligand complex. All these protein-ligand complexes are further aligned so that they lie in the same plane. The protein-ligand complexes are again split into protein and ligand files. MGLTools are used to prepare the proteins and ligands and convert them into pdbqt files. The grid box coordinates are calculated, and a config.txt is created. The config.txt also contains details like exhaustiveness, and num_mode. X, Y, Z centers are 30.536, 44.141, and 1.95, respectively and the size for X, Y and Z are 60, 60, 60, respectively. The number of modes is set as 20 and exhaustiveness as 8. A shell script 'dock_autodock_vina.sh' is used to automate the AutoDock Vina docking process.

```
#!/bin/bash

for i in $(cat files_lists)
do
file1=${i}_protein_file1.pdb
file2=${i}_ligand_file2.pdb
#echo"$i"
/home/neuron3/MGLTools-1.5.6rc3/mgltools_x86_64Linux2_1.5.6rc3/bin/pythonsh /home/neuron3/MGLTools-1.5.6rc3/mgltools_x86_64Linux2_1.5.6
rc3/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_receptor4.py -r /storage/aparna/PDBs_merged_BindingMode_15_04_23/$i/${i}_protein_fi
le1.pdb -A "hydrogens"
/home/neuron3/MGLTools-1.5.6rc3/mgltools_x86_64Linux2_1.5.6rc3/bin/pythonsh /home/neuron3/MGLTools-1.5.6rc3/mgltools_x86_64Linux2_1.5.6
rc3/MGLToolsPckgs/AutoDockTools/Utilities24/prepare_ligand4.py -l /storage/aparna/PDBs_merged_BindingMode_15_04_23/$i/${i}_ligand_file2
.pdb
/home/neuron3/bin/autodock_vina_1_1_2_linux_x86/bin/vina --config /storage/aparna/PDBs_merged_BindingMode_15_04_23/$i/config.txt --out
/storage/aparna/PDBs_merged_BindingMode_15_04_23/$i/${i}_ligand.pdbqt --log /storage/aparna/PDBs_merged_BindingMode_15_04_23/$i/${i}_li
gand_log.txt
 done
~
~
~
~
```

Fig 12: autodockvina.sh

'log.txt' and 'output.pdbqt' are the output files. The output.pdbqt file is split into 20 pdbqt files(20 best poses) and again into pdb files using OpenBabel. The protein.pdb file and ligand.pdb files are combined to form the protein-ligand complexes. As we consider 20 ligand poses for each ligand, we have 20 protein-ligand complexes for each protein-ligand file. These protein-ligand complexes are used to generate features that are given as input to OnionNet, and it predicts binding affinity based on these features. Then the protein-ligand-pose complex with the best binding affinity is selected for each protein-ligand complex. Root Mean Square Deviation (RMSD) between the best ligand pose and the original(experimental) ligand file from PDB is calculated.

RMSD = $\sqrt{\frac{1}{N}\Sigma\left(x_i - x_J\right)^2 + \left(yi - y_J\right)^2 + \left(z_i - z_J\right)^2}$ where x_i, y_i, z_i are the coordinates from the predicted ligand pose and x_j, yj, zj are the coordinates from the experimental ligand pose obtained from PDB. N is the total number of atoms in the ligand. If the RMSD value is ≤ 2 Å, the predicted binding pose is considered as good[14].

Here we also evaluated the performance of AutoDock4 and AutoDock Vina by comparing the best ligand binding pose in each protein-ligand complex with the experimental binding pose obtained from PDB. For Autodock4 a shell script 'dock.sh' is used to automate the protein-ligand preparation and docking. Output.dlg file with details of 20 ligand poses is obtained which is split into pdbqt files and then into pdb files.

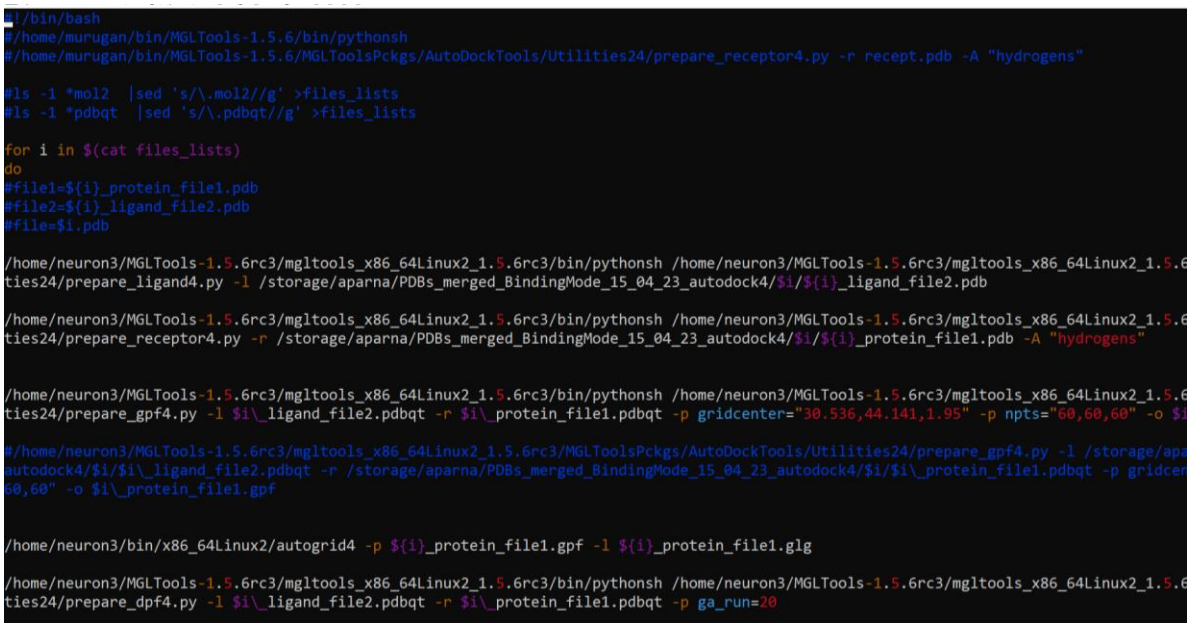


Fig 13: autodock.sh

X, Y, and Z centers are 30.536, 44.141, and 1.95, respectively, and the sizes for X, Y, and Z are 60, 60, and 60, respectively. The number of modes is set as 20.

2.5 System Requirements

This section mentions the minimum system requirements, like the operating system and the various tools/libraries needed for this study. The Operating Systems used are Fedora Linux 36 and Windows 10. Python is the programming language used, and MdTraj, Jupyter Notebook, AutoDock, AutoDock Vina, MGLTools, OpenBabel, and PyMOL are the tools/libraries used.

2.5.1 Python

Python is a high-level programming language that is used in various applications like data analysis, machine learning, scientific computing, etc. Python is simple, has easy-to-understand syntax, and has a large standard library, making it very popular. It can be run on various operating systems like Windows, Linux, etc. For Windows, we can download the latest version of Python from https://www.python.org/downloads/ and

install it. For Linux, we have to open the terminal and run the command '**sudo apt-get install python3**', and we can use the '**python --version**' command to check the version of Python installed. We used Python version 3.9.13 here. PIP is a standard package manager for Python which installs and manages packages that can be used with Python, and '**pip install package**' is the command for package installation. Here package can be numpy, pandas, or any other package as per requirement.

2.5.2 Jupyter Notebook

Jupyter Notebook is an open-source software that provides an interactive computing environment and supports multiple programming languages like Python, R, etc. To install and launch the Notebook, open the command prompt and run the following commands.

'pip install notebook'

'jupyter notebook'

2.5.3 MDTraj

MDTraj is a Python library used for analyzing Molecular Dynamics trajectories. It is very fast and efficient and can handle various file formats, including 'pdb'. In this study, MDTraj parses the protein and ligand 'pdb' files and extracts their atom coordinates, which are further used to calculate the distance between them. We can install MDTraj using the following command.

'pip install mdtraj'

2.5.4 MGLTools

MGLTools is a free software package used widely in computational chemistry and drug design which is developed by The Scripps Research Institute. It is based on Python

and is used for visualizing and analyzing molecular structures. MGLTools comprises a Python Molecular Viewer (PMV), AutoDockTools (ADT) – a set of PMV commands and Vision- a visual programming environment. It runs on Windows, Linux, and Mac and can be downloaded from http://mgltools.scripps.edu/. The scripts 'prepare_receptor4.py' and 'prepare_ligand4.py' in AutoDockTools are used for converting protein. pdb and ligand. pdb to pdbqt files respectively.

2.5.5 AutoDock

AutoDock is a molecular docking software that is widely used in drug discovery. It predicts the binding affinity and orientation of a ligand to a target protein. It uses Lamarckian Genetic Algorithm to explore the conformational space of the ligand and the receptor. AutoDock calculates the energies of different ligand poses in the protein binding pocket and finds the best pose with the lowest binding energy. AutoDock takes pdbqt files as inputs and gives output pdbqt files. PDBQT files are generated using MGLTools. In this study, we used AutoDock4, which is an improved version of AutoDock[15].

(https://autodock.scripps.edu/download-autodock4/)

2.5.6 AutoDock Vina

AutoDock Vina, another molecular docking software developed by The Scripps Research Institute, is used in this study. It is faster and better in docking calculations when compared to the older versions of Autodock. A local search optimization algorithm is used in AutoDock Vina. Like Autodock, AutoDock Vina also uses pdbqt files as input and gives pdbqt files as output[16].

(https://vina.scripps.edu/)

2.5.7 OpenBabel

OpenBabel is an open-source software that is used in computational chemistry to convert one file format to another. We have used OpenBabel to convert files from pdb to pdbqt format and also from mol to pdb format.

(https://openbabel.org/)

2.5.8 PyMOL

PyMOL is a tool by Schrodinger to visualize and analyze molecular structures. It supports a wide range of file formats, such as pdb, mol, etc., and provides tools for importing and exporting structures.

(https://pymol.org/2/)

CHAPTER 3

3 Results and Discussions

3.1 Training, Validating and Testing with PDBbind Dataset

As mentioned in the **PDBbind Dataset Preparation**, we have used 11866 protein-ligand complexes in the PDBbindv2016 for training the OnionNet model. 1132 complexes from the refined set are used for validation, and 285 complexes from the core set are used for testing. The model predicted pKa (binding affinity) for each complex. The predicted pKa values are plotted against the experimental pKa values obtained from PDBbind. We have also calculated R, RMSE, and MAE to evaluate the performance of the model.

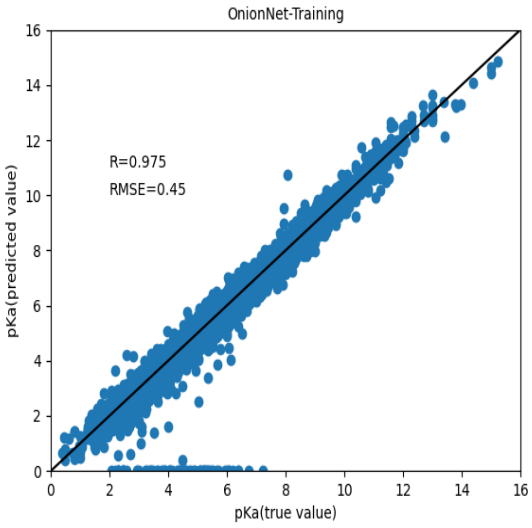


Fig 14: OnionNet-Training-Results

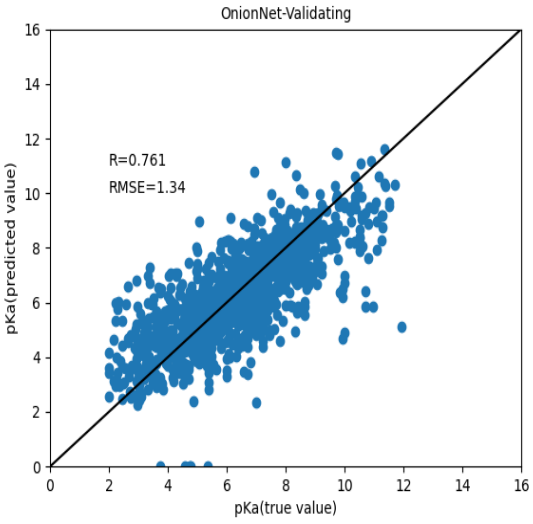


Fig 15: OnionNet-Validation-Results

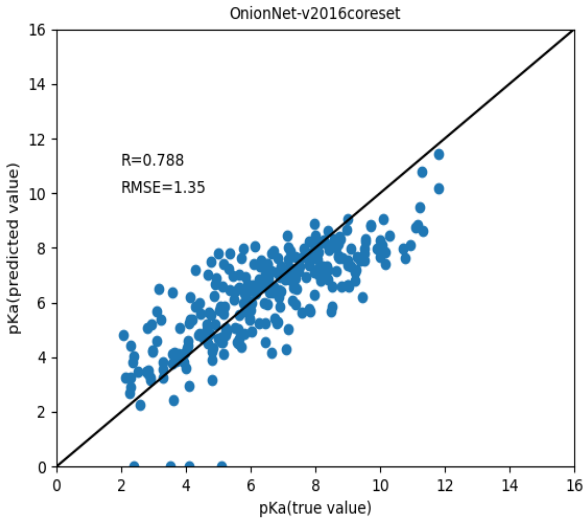


Fig 16: OnionNet-v2016Core set-Results

	Training Set	Validating Set	v2016 Core Set
R	0.975	0.761	0.788
RMSE	0.45	1.34	1.35
MAE	0.23	1.0	1.04

Fig 17: PDBbind Results

We got an R-value of 0.98 for the training set and 0.76 and 0.79 for Validating set and v2016 corset, respectively. Though 0.79 accuracy is less when compared to the R-value of the training set, it is better compared to most of the existing scoring functions.

3.2 Validating OnionNet Based on Energetics (Binding Affinity) Prediction

3.2.1 Validating using BAPPL Dataset

OnionNet predicted binding affinity for the 161 protein-ligand complexes from the BAPPL dataset and here also calculated the R, RMSE, and MAE values.

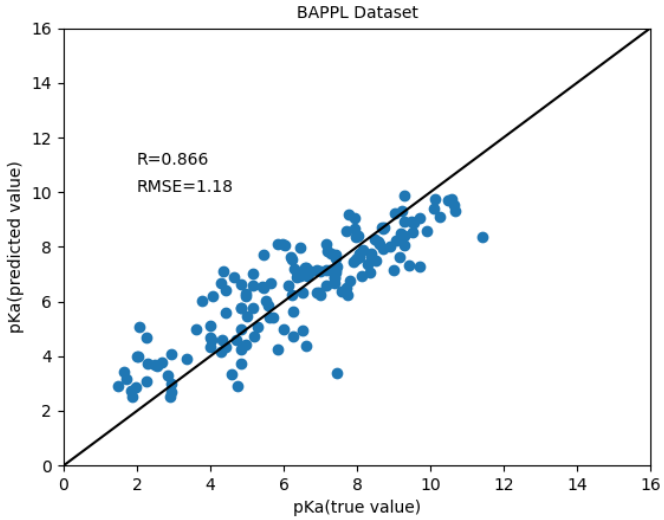


Fig 18: BAPPL Dataset Plot

Here the R is 0.866, which shows OnionNet performs well on this dataset which is a subset of PDB which has experimentally determined protein-ligand 3D structures. The RMSE value is also very close to 1.

BAPPL Dataset	
R	0.866
RMSE	1.18
MAE	0.95

Fig 19: BAPPL Dataset-Results

3.2.2 Validating using BindindDB Dataset

We have used Bace-1 protein and 18 ligands from BindingDB to validate the performance of OnionNet. The best ligand binding poses from Autodock and AutoDock Vina are calculated and corresponding protein-ligand complexes are formed and features generated from these complexes are given as inputs to the OnionNet separately. We have also evaluated the performance of AutoDock and AutoDock Vina against the experimental binding affinity values from BindingDB. For this we have converted the binding energies for the best ligand poses to Ki values (exp(-ΔG/RT)) and further converted Ki values to pKa values (-log10(Ki)).

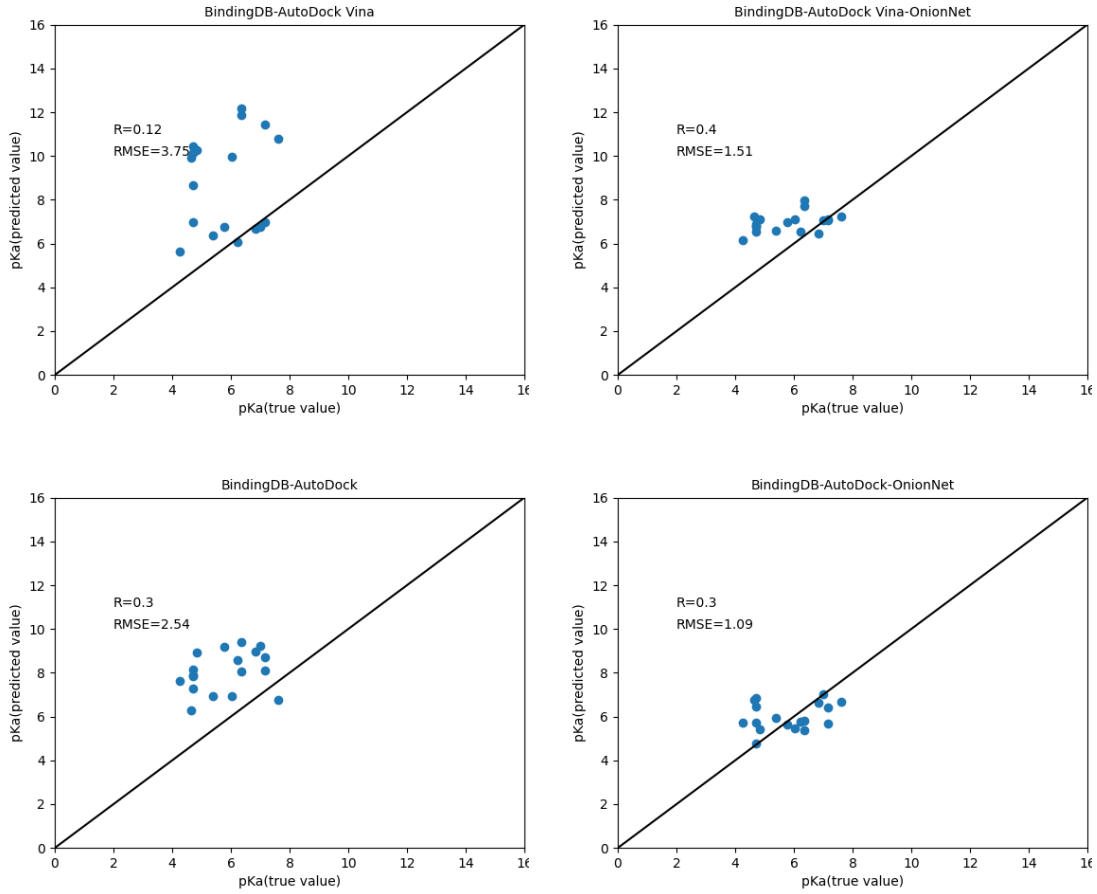


Fig 20: BindingDB Plots

	Autodock	AutoDock-OnionNet	AutoDock Vina	AutoDock Vina-OnionNet
R	0.3	0.3	0.12	0.4
RMSE	2.54	1.09	3.75	1.51
MAE	2.34	0.87	3.05	1.26

Fig 21: BindingDB Results

From the above graphs and table, we can see that protein-ligand binding affinity prediction is less than satisfactory in all four scenarios. Still, when compared with the performance of Autodock and AutoDock Vina, OnionNet performs better. We assume that the quality of 3D structures in BindingDB might have affected the performance of OnionNet. Unlike PDB, where the 3D structures are determined by experimental methods such as X-ray crystallography, NMR spectroscopy, and electron microscopy, in BindingDB, they are computed by the program, Vconf.

3.3 Validating OnionNet Based on Structural Analysis (Best Binding Pose)

Here we are trying to find whether the protein-ligand complex with the best binding affinity gives the best ligand binding pose also. We used 379 Bace1 complexes from PDB, and 3D ligand structures obtained from PDB are used as the reference. We compared the performance of OnionNet, Autodock, and AutoDock Vina in finding the best binding pose. We kept threshold value for RMSD between the experimental and predicted binding poses <= 2 Å.

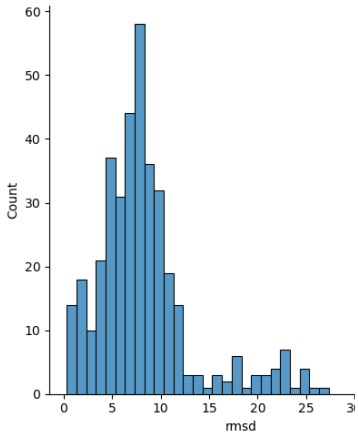


Fig 22:Binding Pose- OnionNet

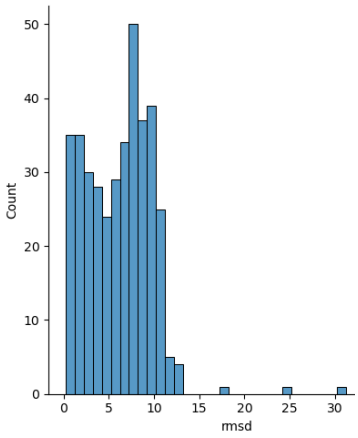


Fig 23: Binding Pose- AutoDock

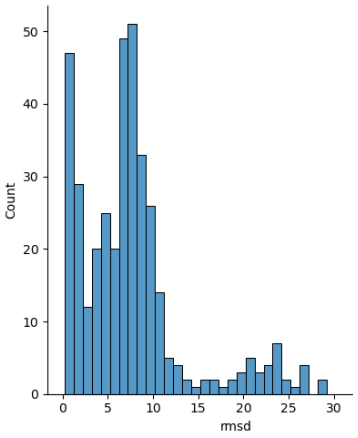


Fig 24: Binding Pose- AutoDock Vina

Number of ligand poses with RMSD<=2Å	OnionNet	Autodock4	AutoDock Vina
	25	61	70

Fig 25: Binding Pose Results

From the above results we can see AutoDock Vina performed better than OnionNet and Autodock4 in predicting the best binding poses. Though the performance of OnionNet is the best when it comes to binding affinity, it's poor in predicting the best binding pose of the ligands.

CHAPTER 4

4 Conclusion and Future Scope

4.1 Conclusion

The machine learning and deep learning models have helped to accelerate the process of drug discovery by predicting the drug properties but the performance of these models mainly depends on the quality of the dataset used. In this thesis, titled “Validating and Benchmarking Deep Learning Model for Protein-Ligand Binding Affinity Prediction”, we tested the CNN model OnionNet on different datasets and evaluated its performance in predicting the protein-ligand binding affinity and also evaluated its performance in identifying the best ligand pose. Initially the model was trained, validated and tested with PDBbind v2016 dataset and it has given Pearson Correlation Coefficient (R) of 0.98, 0.76 and 0.79 for training, validation and testing respectively. To evaluate the performance of OnionNet in predicting protein-ligand binding affinity we used BAPPL dataset and we got R value as 0.87. BAPPL dataset is a subset of Protein Data Bank (PDB) which provides high quality 3D structures of proteins and ligands using experimental methods such as X-ray crystallography, NMR spectroscopy, and electron microscopy. The OnionNet model uses structural features of protein and ligand to predict the binding affinity and we assume that model performs well on dataset with high quality 3D structures of proteins and ligands.

We further tested OnionNet with BindingDB dataset (Bace-1 and 18 ligands) whose data mainly comes from literature and it comprises binding affinities of protein-ligand complexes even when there are no crystal structures available. Here the performance of OnionNet with docked structures (used both Autodock and AutoDock Vina) of protein-ligand complexes (BindindDB) as input was below average. Still its performance was better when compared with AutoDock’s and AutoDock Vina’s performance in predicting protein-ligand binding affinity with the same dataset. Here the performance of OnionNet might have affected by the quality of the protein-ligand structures. Unlike PDB database, BindingDB includes low quality 3D structures of ligand molecules which are later converted to high quality using the program Vconf.

Next, we tried to see that whether the best protein-ligand binding affinity predicted by various scoring functions has the best ligand binding pose or not. For this we have used Bace-1 complexes from PDB dataset and the ligand structures from the PDB dataset are used as the reference and we compared the performance of OnionNet, AutoDock and AutoDock Vina. We have found that out of 379 complexes OnionNet, AutoDock and AutoDock Vina had 25, 61 and 70 complexes with RMSD value <= 2Å respectively. The shows that AutoDock Vina performed well in identifying the best ligand poses while performance of OnionNet was poor.

The overall results lead to the conclusion that OnionNet performs well on high quality 3D structures of protein-ligand complexes and in predicting energetics of the protein-ligand complexes but when it comes to the structural analysis its performance is not satisfactory.

4.2 Future Scope

OnionNet model uses only the structural features of the protein-ligand complexes for the prediction of protein-ligand binding affinity and accuracy of the model is highly dependent on the quality of the 3D structures of the protein-ligand complexes. The performance of the model can be improved by adding

1. Protein Features

- Amino acid sequence and composition
- Physicochemical properties of amino acids (e.g., hydrophobicity, charge)
- Secondary Structure
- Atomic level features (e.g., type of atoms, number of bonds)

2. Ligand Features

- Physicochemical properties of ligands (e.g., molecular weight, polarity)
- Molecular descriptor information (e.g., hydrogen bond acceptors/donors, aromatic rings)

Bibliography

[1] A. Gimeno *et al.*, “The Light and Dark Sides of Virtual Screening: What Is There to Know?,” *Int. J. Mol. Sci.*, vol. 20, no. 6, p. 1375, Mar. 2019, doi: 10.3390/ijms20061375.

[2] N. M. O’Boyle and R. A. Sayle, “Comparing structural fingerprints using a literature-based similarity benchmark,” *J. Cheminformatics*, vol. 8, no. 1, p. 36, Dec. 2016, doi: 10.1186/s13321-016-0148-0.

[3] S. P. Leelananda and S. Lindert, “Computational methods in drug discovery,” *Beilstein J. Org. Chem.*, vol. 12, pp. 2694–2718, Dec. 2016, doi: 10.3762/bjoc.12.267.

[4] G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe, “Computational Methods in Drug Discovery,” *Pharmacol. Rev.*, vol. 66, no. 1, pp. 334–395, Jan. 2014, doi: 10.1124/pr.112.007336.

[5] I. A. Guedes, F. S. S. Pereira, and L. E. Dardenne, “Empirical Scoring Functions for Structure-Based Virtual Screening: Applications, Critical Aspects, and Challenges,” *Front. Pharmacol.*, vol. 9, p. 1089, Sep. 2018, doi: 10.3389/fphar.2018.01089.

[6] J. Li, A. Fu, and L. Zhang, “An Overview of Scoring Functions Used for Protein–Ligand Interactions in Molecular Docking,” *Interdiscip. Sci. Comput. Life Sci.*, vol. 11, no. 2, pp. 320–328, Jun. 2019, doi: 10.1007/s12539-019-00327-w.

[7] R. Gupta, D. Srivastava, M. Sahu, S. Tiwari, R. K. Ambasta, and P. Kumar, “Artificial intelligence to deep learning: machine intelligence approach for drug discovery,” *Mol. Divers.*, vol. 25, no. 3, pp. 1315–1360, Aug. 2021, doi: 10.1007/s11030-021-10217-3.

[8] L. Zheng, J. Fan, and Y. Mu, “OnionNet: a Multiple-Layer Intermolecular-Contact-Based Convolutional Neural Network for Protein–Ligand Binding Affinity Prediction,” *ACS Omega*, vol. 4, no. 14, pp. 15956–15965, Oct. 2019, doi: 10.1021/acsomega.9b01997.

[9] R. Wang, X. Fang, Y. Lu, and S. Wang, “The PDBbind Database: Collection of Binding Affinities for Protein−Ligand Complexes with Known Three-Dimensional Structures,” *J. Med. Chem.*, vol. 47, no. 12, pp. 2977–2980, Jun. 2004, doi: 10.1021/jm030580l.

[10] T. Jain and B. Jayaram, “An all atom energy based computational protocol for predicting binding affinities of protein-ligand complexes,” *FEBS Lett.*, vol. 579, no. 29, pp. 6659–6666, Dec. 2005, doi: 10.1016/j.febslet.2005.10.031.

[11] M. K. Gilson, T. Liu, M. Baitaluk, G. Nicola, L. Hwang, and J. Chong, “BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology,” *Nucleic Acids Res.*, vol. 44, no. D1, pp. D1045–D1053, Jan. 2016, doi: 10.1093/nar/gkv1072.

[12] H. M. Berman, “The Protein Data Bank,” *Nucleic Acids Res.*, vol. 28, no. 1, pp. 235–242, Jan. 2000, doi: 10.1093/nar/28.1.235.

[13] S. L. Cole and R. Vassar, “The Alzheimer’s disease Beta-secretase enzyme, BACE1,” *Mol. Neurodegener.*, vol. 2, no. 1, p. 22, 2007, doi: 10.1186/1750-1326-2-22.

[14] N. A. Murugan, A. Podobas, D. Gadioli, E. Vitali, G. Palermo, and S. Markidis, “A Review on Parallel Virtual Screening Softwares for High-Performance Computers,” *Pharmaceuticals*, vol. 15, no. 1, p. 63, Jan. 2022, doi: 10.3390/ph15010063.

[15] G. M. Morris *et al.*, “AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility,” *J. Comput. Chem.*, vol. 30, no. 16, pp. 2785–2791, Dec. 2009, doi: 10.1002/jcc.21256.

[16] O. Trott and A. J. Olson, “AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading,” *J. Comput. Chem.*, p. NA-NA, 2009, doi: 10.1002/jcc.21334.