Youssef Ezz Eldeen Ezzat

MASTER'S DEGREE IN SECURITY ENGINEERING AND ARTIFICAL INTELLIGENCE (MESIIA)

MASTER'S Thesis FINAL PROJECT

Directed by Prof. Francesc Serratosa

Drug potency prediction of SARS-Cov2-Mpro based on Graph Convolutional Network

UNIVERSITAT ROVIRA i VIRGILI

Tarragona
September 5, 2024

Contents

1 Introduction

A virus encodes one or more proteases which are enzymes that spur the formation of new protein products, thus play crucial roles in virus replication , and are important targets for the design and development of potent antiviral agents or drugs. Binding affinity is the strength of the binding interaction between a single biomolecule (e.g., a virus protein) to its ligand or binding partner (e.g., a drug). it is a key to appreciating the intermolecular interactions driving biological processes and measured as part of the drug discovery process to help design drugs that bind their targets selectively and specifically.

The development of new drugs is costly, time consuming and often accompanied with safety issues. Drug repurposing can avoid the expensive and lengthy process of drug development by finding new uses for already approved drugs. In order to repurpose drugs effectively, it is useful to know which proteins are targeted by which drugs which is the definition of Drug-Target-Affinity(DTA)[1]

there is a strong motivation to build computational models that can estimate the interaction strength of new drug–target pairs based on previous drug–target experiments.

the DTA prediction problem as a regression task where the input is a drug–target pair and the output is a continuous measurement of binding affinity for that pair.

2 Abstract

A graph, in general, is a data structure depicting a collection of entities represented as nodes, and their pairwise relationships represented as edges. There is a growing interest in having graph-based techniques applied to machine learning, for instance, in biotechnology, they are used to represent drugs and proteins in order to predict their binding affinity. The aim of this master thesis is to apply graph regression techniques based on graph convolutional networks (GCN) to predict the binding affinity of drugs and the SARS-Cov2 Mpro.
Thus, the specific aim of this master thesis is to define and code in python a GCN to predict the binding affinity of a database generated in the URV composed of several pairs of the main protease of SARS-Cov2 and a drug.
The master will include the theoretical explanations of

- Binding affinity prediction
- Graph Convolutional Networks

And also the practical use of the following Python code https://github.com/YoussefEzz/GraphDTA_forked [3] to perform the prediction.

3 Drug Representation

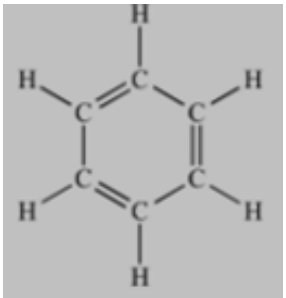
3.1 SMILES notation

Simplified Molecular Input Line Entry System (SMILES) was invented to represent molecules to be readable by computers. enabling several efficient applications, including fast retrieval and substructure searching. From the SMILES code, drug descriptors like the number of heavy atoms or valence electrons can be inferred and readily used as features for affinity prediction. One could also view the SMILES code as a string

- Methane: "C"

- Ethanol: "CCO"
- Benzene: "c1ccccc1"
- Glucose: "OC[C@@H]1OC@HC@@HC@H[C@H]1O"

for example the benzene molecule with the SMILES notation "c1ccccc1" has six carbon atoms in a circular and planar shape which can be inferred from the SMILES notation and it's worth mentioning that this is a famous example of an aromatic molecule where aromaticity is an important feature of stability.



4 Protein Representation

4.1 Protein Sequence

A protein sequence is the order of amino acids in a protein. Amino acids are the building blocks of proteins, and there are about 20 different amino acids that can be found in proteins. The sequence of amino acids in a protein determines its three-dimensional structure and its function. so The sequence is a string of ASCII characters which represent amino acids.
Each amino acid type is encoded with an integer based on its associated alphabetical symbol [e.g. Alanine (A) is 1, Cystine (C) is 3, Aspartic Acid (D) is 4 and so on], allowing the protein to be represented as an integer sequence. imagine a protein as a word. Each letter in the word represents an amino acid. The order of the letters in the word determines the meaning of the word. Similarly, the order of amino acids in a protein determines its function.
an example of a protein sequence, representing the first 7 amino acids of the protein insulin: **GIVEQCC...**
This sequence represents the following amino acids:

- **G**: Glycine
- **I**: Isoleucine
- **V**: Valine
- **E**: Glutamic acid
- **Q**: Glutamine
- **C**: Cysteine
- **C**: Cysteine

5 Binding affinity measurements

5.1 The kinase dissociation constant kd

It measures the equilibrium between the ligand(drug)-protein complex and the dissociated components.
It corresponds to the affinity which the ligand has for the binding site.
under usual conditions the dissociation constant gives the ligand concentration at which half of the protein molecules have ligand bound.
Ligands with higher, more favorable free energy of association bind "tighter" and therefore have greater preference for the bound state. Because **kd** is defined as a dissociation constant, higher affinity ligands have lower

kd values.
As an equilibrium constant, we can express it as the ratio of product concentrations over reactants:

$$\mathrm{PL} \underset{K_\mathrm{d}}{\rightleftharpoons} \mathrm{P} + \mathrm{L} \qquad K_\mathrm{d} = \frac{[P][L]}{[PL]}$$

where

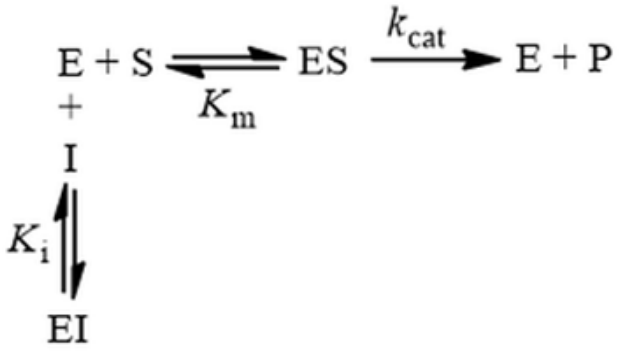
P is the free protein concentration

L is the free ligand concentration

PL is the protein-ligand complex

5.2 The kinase Inhibition Constant ki

represents the affinity of the drug molecule for its target receptor, specifically in the context of competitive inhibition.



where

E is the free enzyme concentration

I is the inhibitor or drug

S is the substrate or the molecules that an enzyme acts upon.

ES is the enzyme-substrate complex concentration (ES)

P is the product concentration which is the Protein in this case

km is the Michaelis constant is a kinetic parameter, not an equilibrium constant. It gives the substrate concentration at which half of the maximum enzymatic reaction rate is attained. It is determined not only by the substrate's binding affinity, but also by how quickly the enzyme-substrate complex is turned over into product.
It's a measure of the affinity of an enzyme for its substrate.

5.3 inhibitory concentration 50% IC50

the concentration at which the inhibitor causes a 50 inhibition of enzymatic activity less precise than **Ki** or **Kd**
A lower IC50 value indicates a higher affinity of the drug for the receptor

$$0.5 = \frac{K_\mathrm{m} + [S]}{K_\mathrm{m}\left(1 + \frac{\mathrm{IC_{50}}}{K_\mathrm{i}}\right) + [S]} \qquad \mathrm{IC_{50}} = K_\mathrm{i}\left(1 + \frac{[S]}{K_\mathrm{m}}\right)$$

where

S is the substrate or the molecules that an enzyme acts upon.

ki is The kinase Inhibition Constant

km is the Michaelis constant is a kinetic parameter, not an equilibrium constant. It gives the substrate concentration at which half of the maximum enzymatic reaction rate is attained. It is determined not only by the substrate's binding affinity, but also by how quickly the enzyme-substrate complex is turned over into product.
It's a measure of the affinity of an enzyme for its substrate.

6 Datasets

6.1 davis benchmark dataset

6.1.1 binding affinity measurement

davis dataset uses The kinase dissociation constant **kd** values after transformation into negative logarithm (base 10) logspace (pKd) using equation $pK_d = -\log_{10}(K_d/1e9)$. with values of **pkd** ranging ranging from 5.0 to 10.8 where (pKd=5) indicates weak or no interaction which corresponds to $kd = 10000$ nM or nanoMolar.

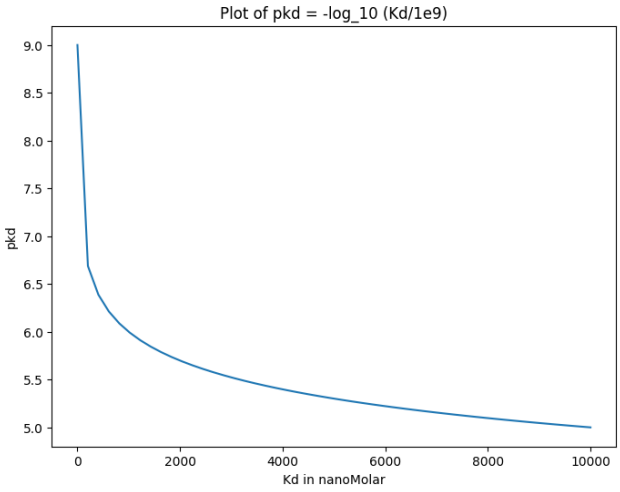


Figure 1: linear

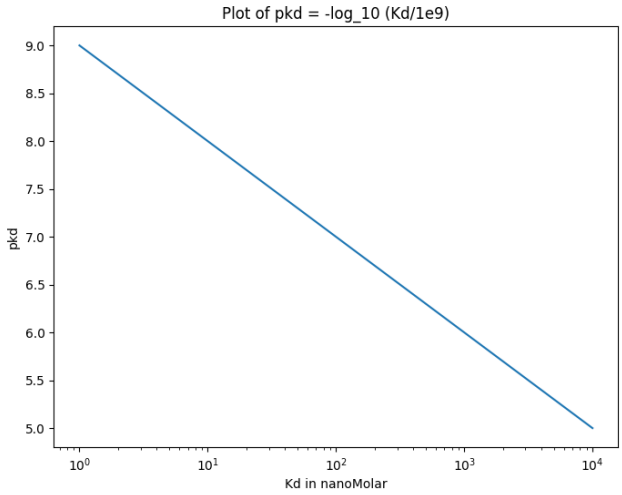
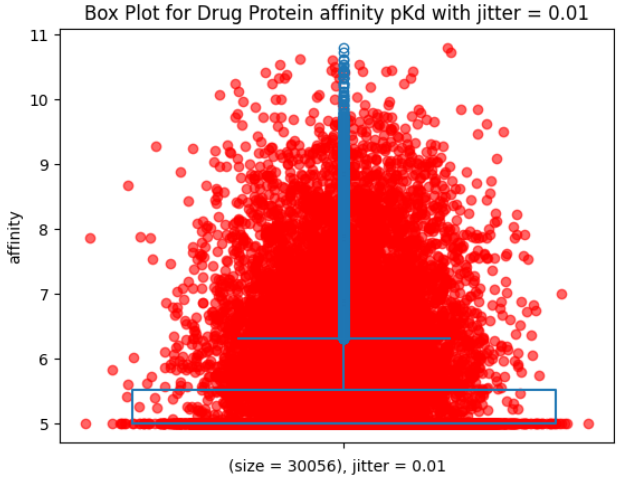
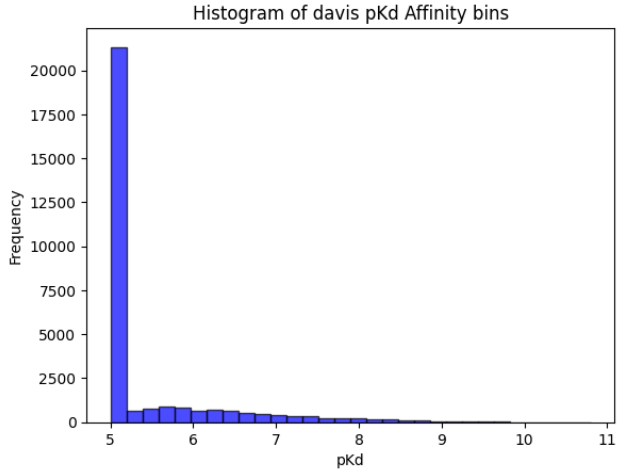


Figure 2: log

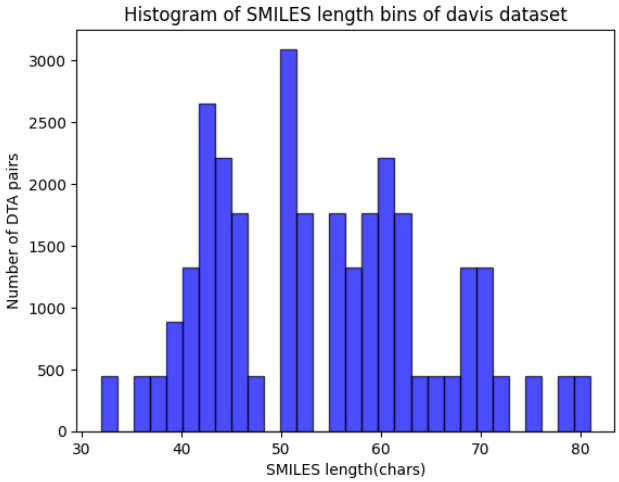
6.1.2 structure

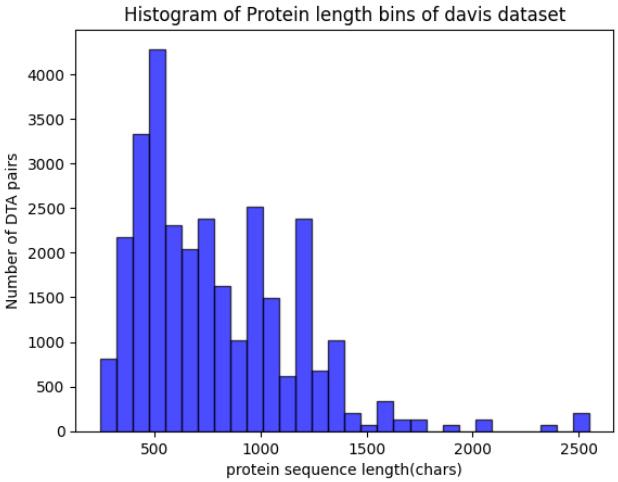
contains the pkd binding affinities for all pairs of 68 drugs and 442 targets, total of 30056 interactions 25047 train set + 5011 test set. 69% of which have affinity values of 10000 nM (pKd=5) indicating weak or no interaction.





also some statistics about davis dataset below show the histogram distribution of SMILES notation and protein sequence with respect to number of characters





6.2 kiba benchmark dataset

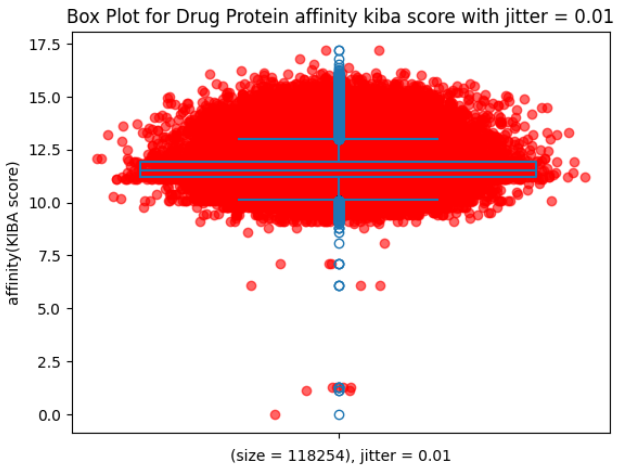
6.2.1 binding affinity measurement

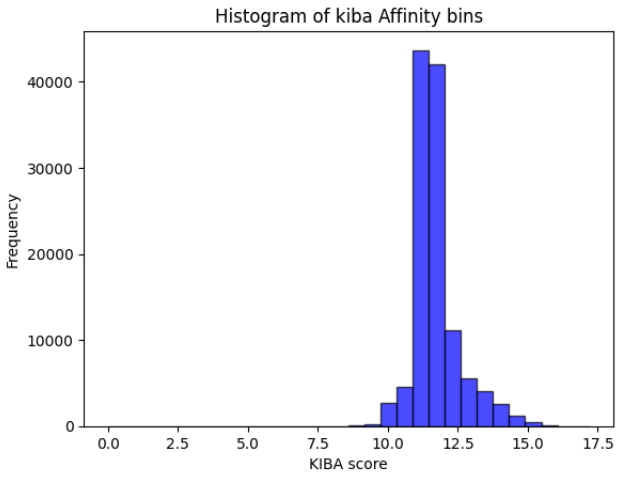
kiba dataset uses the so called KIBA score which is the integration of heterogeneous information from IC50, Ki , and Kd measurements into a single bioactivity score called by the following adjustments

$$\text{KIBA} = \begin{cases} K_i.\ \text{adj} & \text{if } IC_{50} \text{ and } K_i \text{ are present} \\ K_d.\ \text{adj} & \text{if } IC_{50} \text{ and } K_d \text{ are present} \\ (K_i.\ \text{adj} + K_d.\ \text{adj})/2 & \text{if } IC_{50},\ K_i, \text{ and } K_d \text{ are present} \end{cases}$$

6.2.2 structure

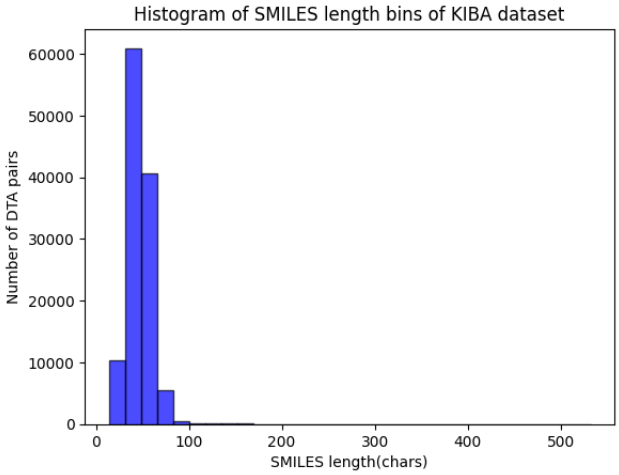
measured as KIBA scores and ranging from 0.0 least affinity to 17.2 highest affinity Total of 118257 interactions(98547 train set + 19710 test set) most interactions between 10 and 15 kiba score

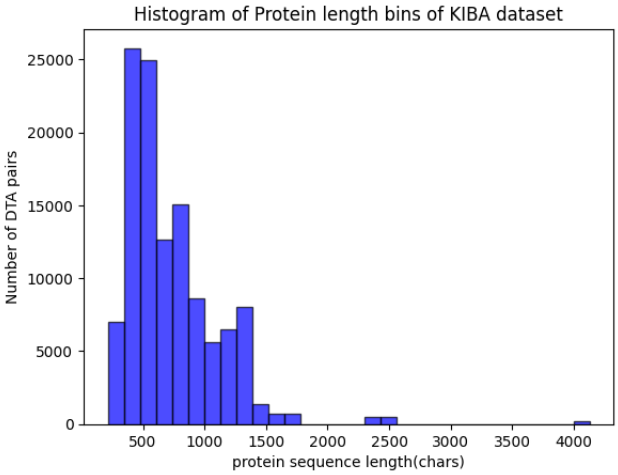




KIBA, on the other hand, has about three-times more interactions with KIBA scores. KIBA values are computed from the combination of heterogenous information sources such as IC50, Ki and Kd. the filtered version of the KIBA dataset is used, in which each protein and ligand has at least ten interactions.

also some statistics about kiba dataset below show the histogram distribution of SMILES notation and protein sequence with respect to number of characters





6.3 URV dataset

6.3.1 binding affinity measurement

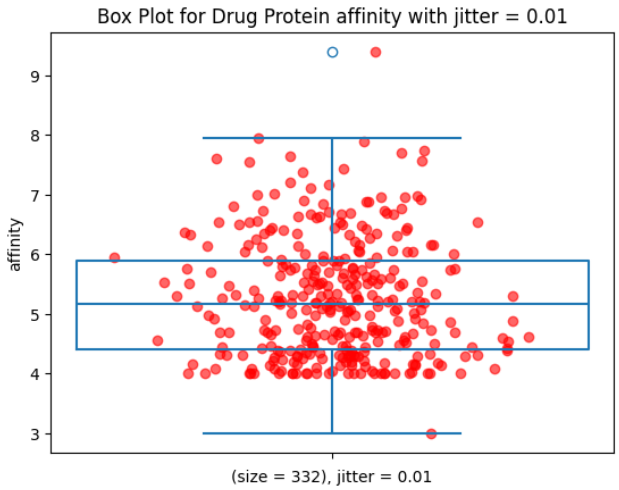
In URV-May-2024 database, the URV systematically collected all structures from the Protein Data Bank (PDB) [11] containing the SARS-CoV-2 Mpro protein - also known as the main protease or 3CL protease, is a key enzyme in the replication and transcription of the SARS-CoV-2 virus, which causes COVID-19 -, devoid of mutations and crystallized with a non-covalent inhibitor. Subsequently, we refined this dataset by selecting structures with available IC50 values from ChEMBL [12] and BindingDB [13] databases, resulting in a final set of 233 structures. For each structure, we obtained the inhibitor's structure in SDF format, the protein-inhibitor complex in PDB format, and the corresponding IC50 value for Mpro inhibition. IC50 represents the concentration of inhibitor required to inhibit 50% of enzyme activity. Additionally, we transformed IC50 values into pIC50, the negative logarithm (base 10) of the IC50 value, where a higher pIC50 value indicates a more potent inhibitor.

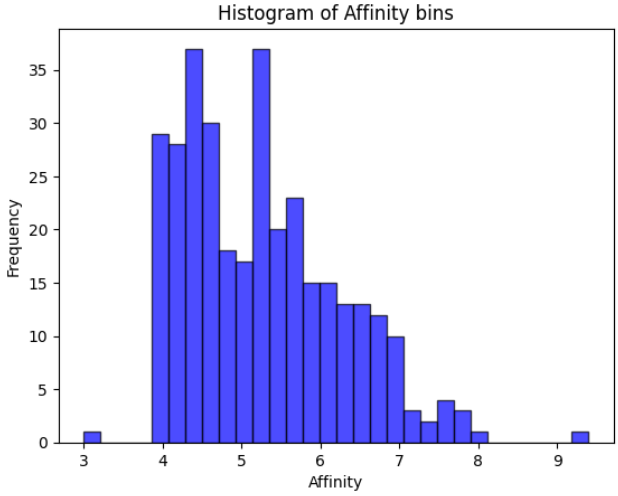
6.3.2 structure

affinity values are provided and some statistics about them are provided below but the measurement unit is unknown.

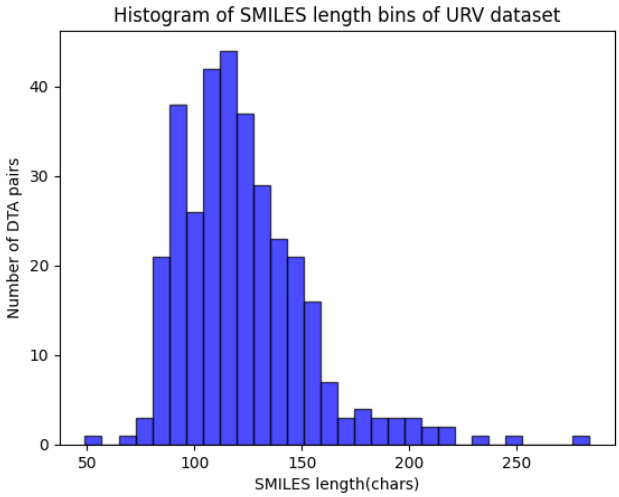
affinity values are provided in csv file linked with protein ID and the protein ID in turn is found in the names of the files of both ligand and the protein

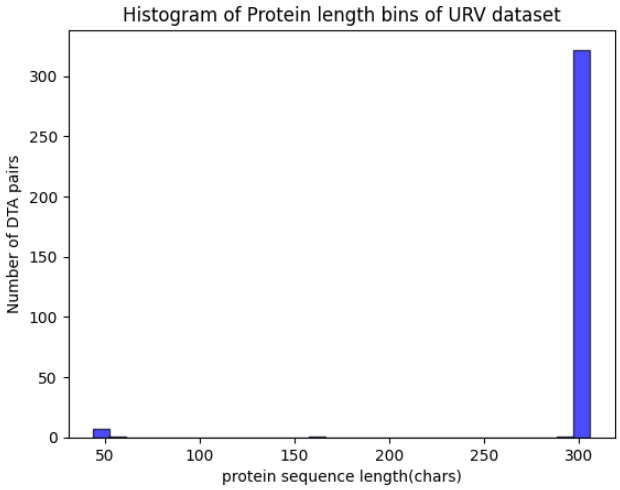
the following figures explain statistics of the affinity values of the 322 pairs, note that jitter was added to one version of the box plot for the sake of clarity





also some statistics about URV dataset below show the histogram distribution of SMILES notation and protein sequence with respect to number of characters





6.3.3 protein preprocessing

the protein data are initially provided as **Protein Data Bank(PDB)** files. PDB is a file format used to store 3D structural information about proteins. It is one of the most widely used formats for representing and exchanging structural data in bioinformatics and structural biology. PDB files contain atomic coordinates of atoms in a molecule, along with metadata and additional information such as experimental methods used to determine the structure.

the PDB file name is named after the protein ID. if a PDB file e.g. **6M2N_protein.pdb** is open with any text editor e.g. notepad++, the metadata including its ID, name, type are in the header and title tags as shown in figure 3 for protein ID **6M2N**

6M2N_protein.pdb

```
1  HEADER    VIRAL PROTEIN                           28-FEB-20   6M2N
2  TITLE     SARS-COV-2 3CL PROTEASE (3CL PRO) IN COMPLEX WITH A NOVEL INHIBITOR
3  COMPND    MOL_ID: 1;
4  COMPND   2 MOLECULE: 3C-LIKE PROTEINASE;
5  COMPND   3 CHAIN: A, B, C, D;
6  COMPND   4 SYNONYM: 3CL-PRO, 3CLP, MAIN PROTEASE, MPRO, NON-STRUCTURAL PROTEIN
7  COMPND   5 5, NSP5, SARS CORONAVIRUS MAIN PROTEINASE;
8  COMPND   6 EC: 3.4.22.69;
9  COMPND   7 ENGINEERED: YES
10 SOURCE    MOL_ID: 1;
11 SOURCE   2 ORGANISM_SCIENTIFIC: SEVERE ACUTE RESPIRATORY SYNDROME CORONAVIRUS
12 SOURCE   3 2;
13 SOURCE   4 ORGANISM_COMMON: 2019-NCOV;
14 SOURCE   5 ORGANISM_TAXID: 2697049;
15 SOURCE   6 EXPRESSION_SYSTEM: ESCHERICHIA COLI BL21(DE3);
16 SOURCE   7 EXPRESSION_SYSTEM_TAXID: 469008;
17 SOURCE   8 EXPRESSION_SYSTEM_STRAIN: BL21(DE3)
18 KEYWDS    SARS-COV-2, 3CL PRO, VIRAL PROTEIN
19 EXPDTA    X-RAY DIFFRACTION
20 AUTHOR    H.X.SU,W.F.ZHAO,M.J.LI,H.XIE,Y.C.XU
```

Figure 3: PDB file example.

more informtion can be viewed at the PDB website [4] for the same protein ID **6M2N**

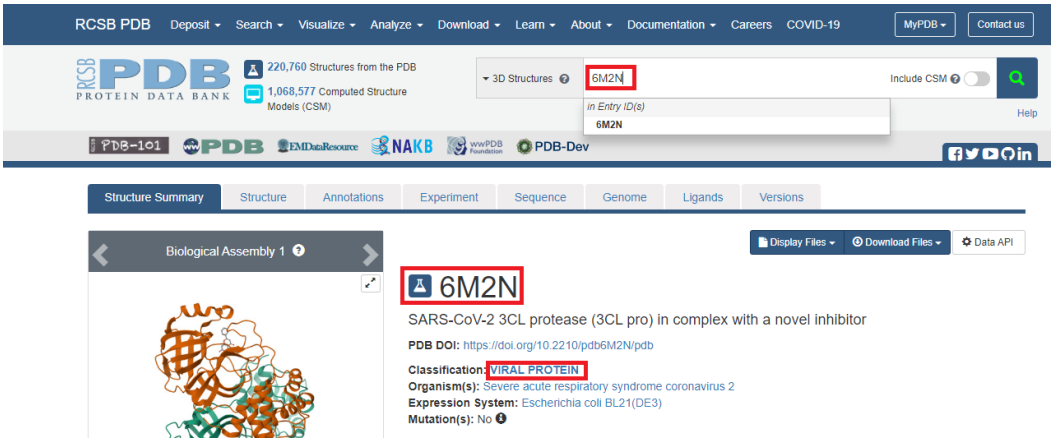


Figure 4: PDB file example.

a protein can be either Single Chain or Multi-Chain, so it can be composed of one chain of amino acids(sequence) - up to four - but usually two, we pick the longest sequence as the representation for the protein this is done using **Bio.PDB** module in the **Biopython** library that provides tools for working with Protein Data Bank (PDB) files in python. classes used are:

- **PDBParser** used to parse PDB files, create a structure object and read the atomic coordinates and other structural information from a PDB file and converts it into a hierarchical structure object, which can be easily manipulated and analyzed.
- **PDBBuilder** used to identify and construct polypeptides (chains of amino acids) from a structure object (such as a protein structure obtained from a PDB file)

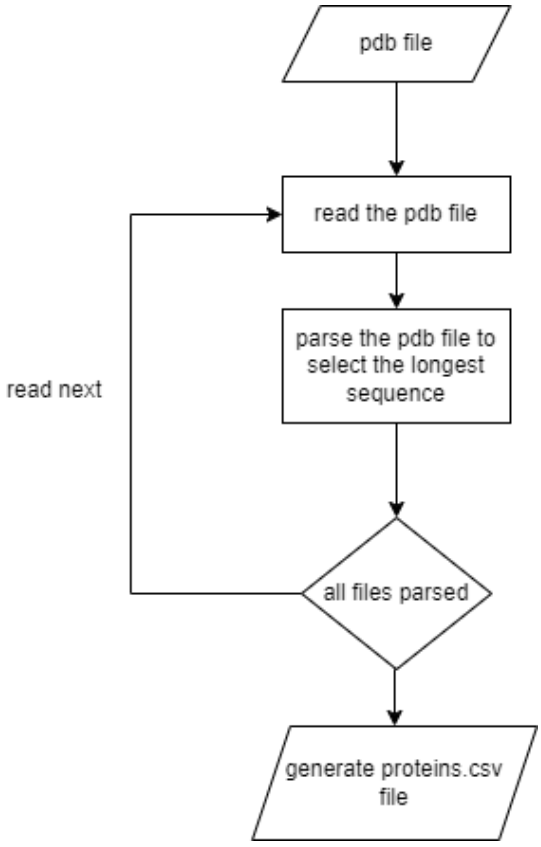


Figure 5: PDB file example.

6.3.4 drug(ligand) preprocessing

the ligand data are provided as **Structure Data file(SDF)** file format used for representing chemical compounds and their associated data. It is primarily used to store information about molecules, including their structure and various properties, in a structured way.

the SDF file name is named after the protein ID with which the ligand is paired. consider SDF file **6M2N_ligand.sdf** shown in fig 6 where its name does not indicate any information about the ligand but suggests it will be paired with previous protein to get information about the ligand itself, it can be open using a text editor e.g. notepad++ and at the end of the file information that identify the ligand can be found e.g. **chemical formula** and **Simplified Molecular Input Line Entry System(SMILES)** notation which is is a chemical notation that allows a user to represent a chemical structure in a way that can be used by the computer.

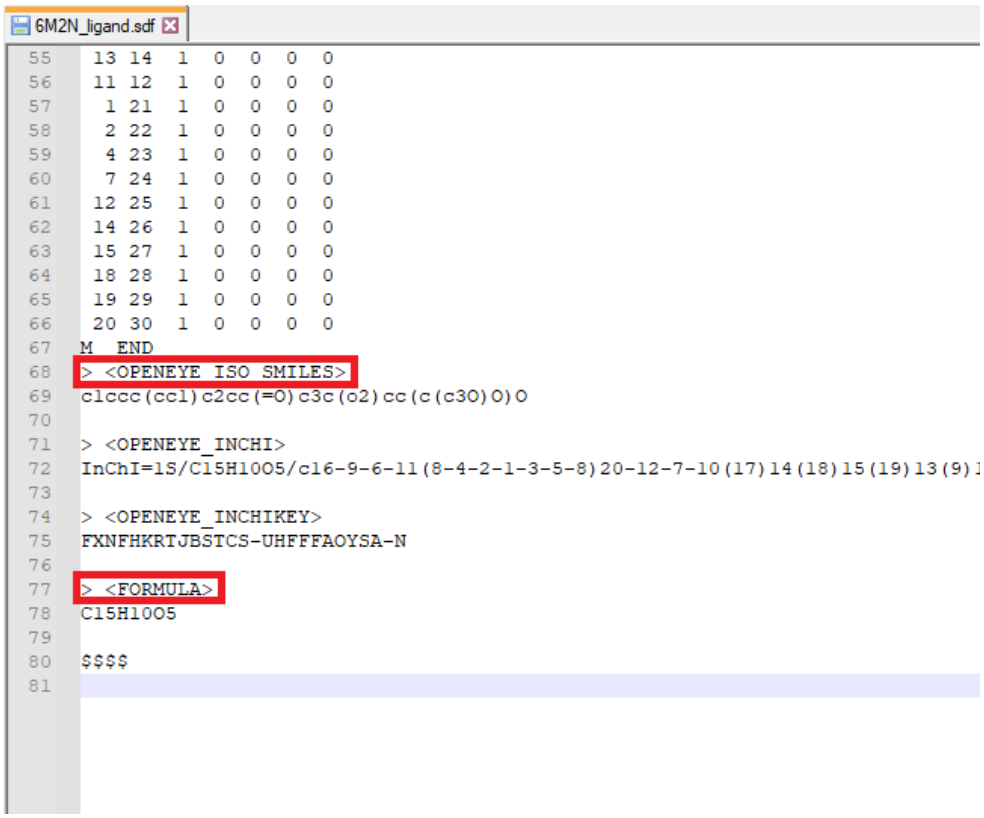


Figure 6: SDF file example.

while the chemical formula may not be unique, the SMILES notation is always unique. so it can be used in many websites e.g. the chemspider website[6] to search for more information about the ligand

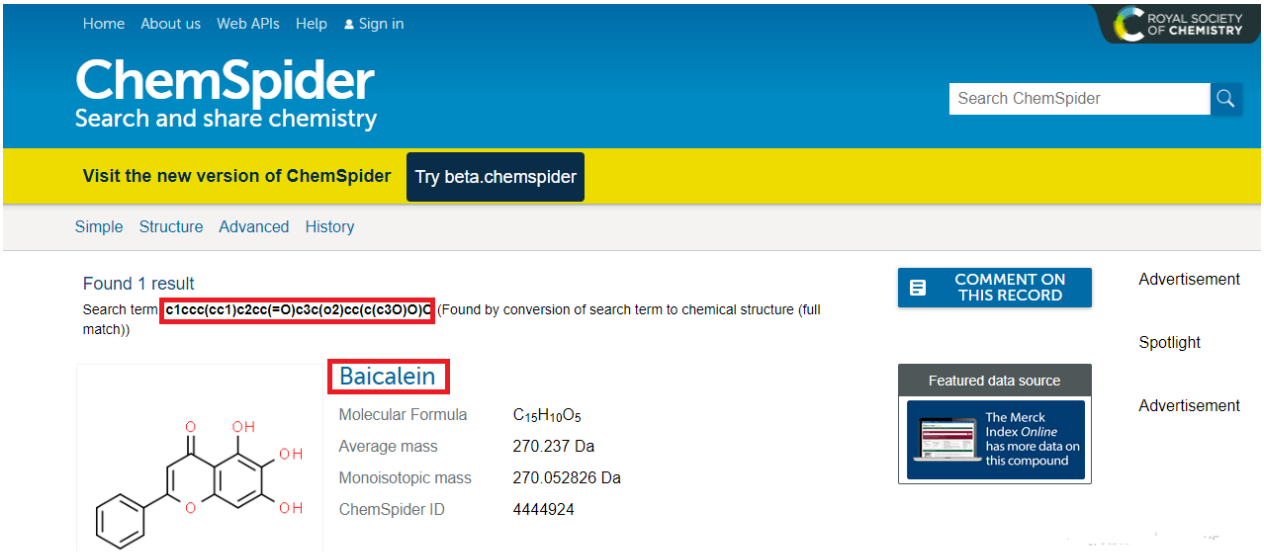


Figure 7: SDF website.

An SDF ligand file is read and first the molecule is validated(sanitized) to check if error exist in the

structure, if the structure is valid the SMILES notation is extracted and appended to valid ligands SDF file else it is excluded and added to invalid ligands SDF file.

The **rdkit.Chem** package [7] was used to read the molecule using Chem.SDMolSupplier function, validate it using Chem.SanitizeMol and convert it to SMILES notation using function Chem.MolToSmiles

- **Chem.SDMolSupplier** used to read the molecule.
- **Chem.SanitizeMol** used to validate the molecule
- **Chem.MolToSmiles** obtain the SMILES notation of the molecule

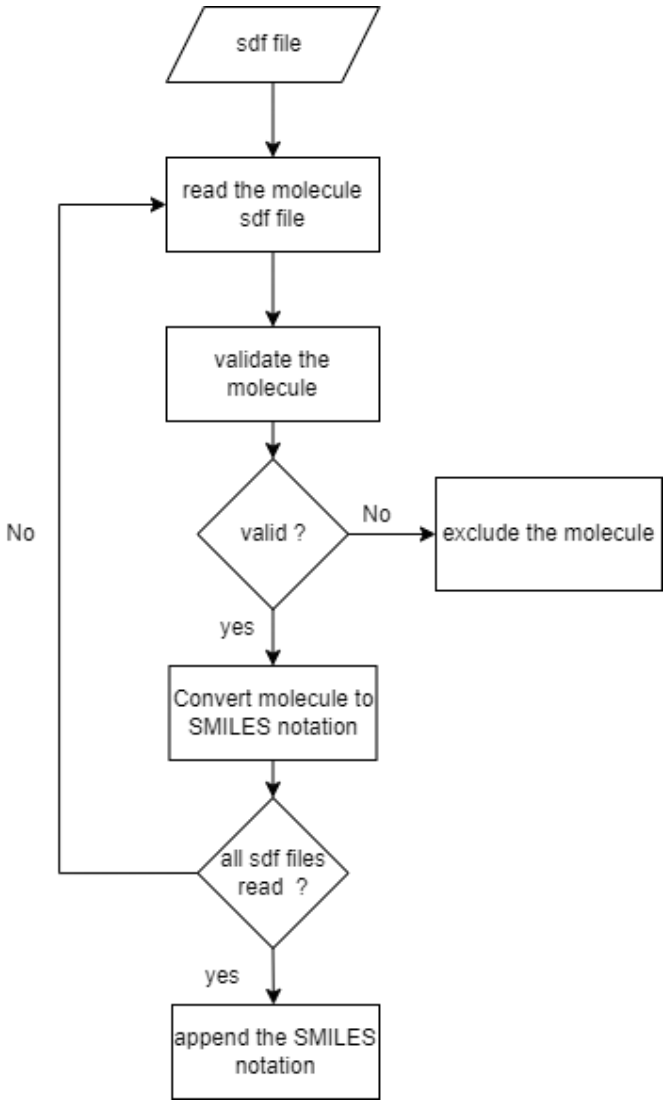


Figure 8: SDF workflow.

7 Previous Work and contribution

7.1 Previous Work

7.1.1 collaborative filtering (2017) [8]

The SimBoost model uses the affinity similarities among drugs and among targets to build new features. collaborative filtering is used in the following way:

- Gradient Boosting Machines (GBM): GBMs are powerful machine learning models that can capture complex relationships between features. SimBoost uses GBMs to learn the relationship between drug similarity, target similarity, and binding affinity.

- Similarity-Based Approach: SimBoost utilizes a similarity measure to identify drugs and targets that are similar to the query drug and target. This allows the model to leverage existing data on similar compounds and targets to make predictions for new ones.
- Read-Across: By combining GBMs with a similarity-based approach, SimBoost performs a "read-across" from known data to predict binding affinities for new drug-target pairs.

7.1.2 DeepDTA model (2018) [9]

DeepDTA uses a deep neural network architecture with two branches to learn complex relationships between drug and target features. It combines:

- convolutional neural networks (CNNs): for capturing local patterns in molecular structures of drugs.
- recurrent neural networks (RNNs): for capturing sequential information in protein sequences.

protein representation learned from RNN then combined with the drug representations learned by CNN to predict the binding affinity.

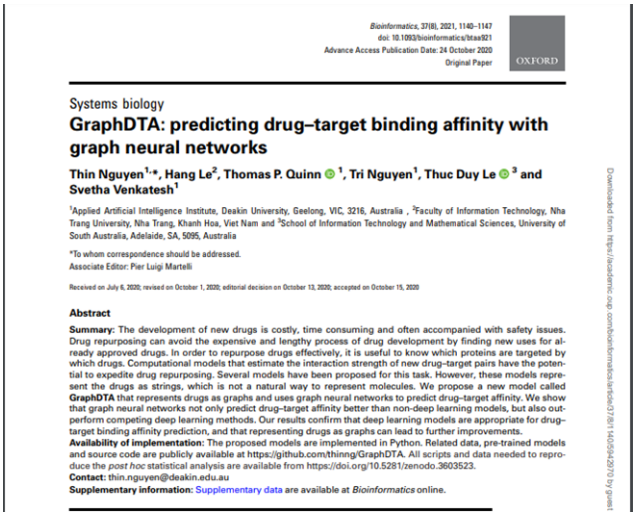
7.1.3 WideDTA model (2019) [10]

WIDEDTA uses CNNs to learn complex patterns from both the drug SMILES notation and target protein sequence representations. CNNs are particularly well-suited for capturing local patterns in molecular structures and protein sequences. in which the sequences of the drugs and proteins are first summarized as higher-order features.
WIDEDTA is considered an extension to and outperforms DEEPDTA.

7.2 Contribution

7.2.1 GraphDTA paper (2020)

a graph neural network architecture capable of directly modeling drugs as molecular graphs outperforms previous deep learning models. so This paper tests the hypothesis that a graph structure could yield a better representation for drugs directly modeling drugs as molecular graphs and the results show that it outperforms previous deep learning models



Bioinformatics, 37(8), 2021, 1140–1147
doi: 10.1093/bioinformatics/btaa921
Advance Access Publication Date: 24 October 2020
Original Paper

Systems biology

GraphDTA: predicting drug–target binding affinity with graph neural networks

Thin Nguyen[1,*], Hang Le[2], Thomas P. Quinn [1], Tri Nguyen[1], Thuc Duy Le [3] and Svetha Venkatesh[1]

[1]Applied Artificial Intelligence Institute, Deakin University, Geelong, VIC, 3216, Australia , [2]Faculty of Information Technology, Nha Trang University, Nha Trang, Khanh Hoa, Viet Nam and [3]School of Information Technology and Mathematical Sciences, University of South Australia, Adelaide, SA, 5095, Australia

*To whom correspondence should be addressed.
Associate Editor: Pier Luigi Martelli

Received on July 6, 2020; revised on October 1, 2020; editorial decision on October 13, 2020; accepted on October 15, 2020

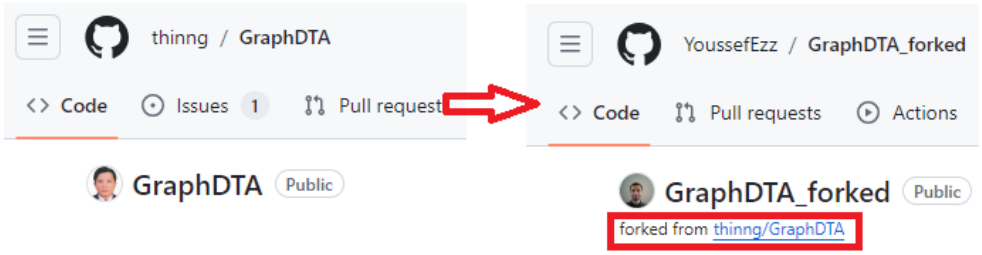
Abstract

Summary: The development of new drugs is costly, time consuming and often accompanied with safety issues. Drug repurposing can avoid the expensive and lengthy process of drug development by finding new uses for already approved drugs. In order to repurpose drugs effectively, it is useful to know which proteins are targeted by which drugs. Computational models that estimate the interaction strength of new drug-target pairs have the potential to expedite drug repurposing. Several models have been proposed for this task. However, these models represent the drugs as strings, which is not a natural way to represent molecules. We propose a new model called **GraphDTA** that represents drugs as graphs and uses graph neural networks to predict drug-target affinity. We show that graph neural networks not only predict drug–target affinity better than non-deep learning models, but also outperform competing deep learning methods. Our results confirm that deep learning models are appropriate for drug-target binding affinity prediction, and that representing drugs as graphs can lead to further improvements.
Availability of implementation: The proposed models are implemented in Python. Related data, pre-trained models and source code are publicly available at https://github.com/thinng/GraphDTA. All scripts and data needed to reproduce the *post hoc* statistical analysis are available from https://doi.org/10.5281/zenodo.3603523.
Contact: thin.nguyen@deakin.edu.au
Supplementary information: Supplementary data are available at *Bioinformatics* online.

7.2.2 Work done in the thesis

the work done includes the following:

- **fork the repository**:
  first the repository of the GraphDTA paper was forked in order to run the code and view the results.



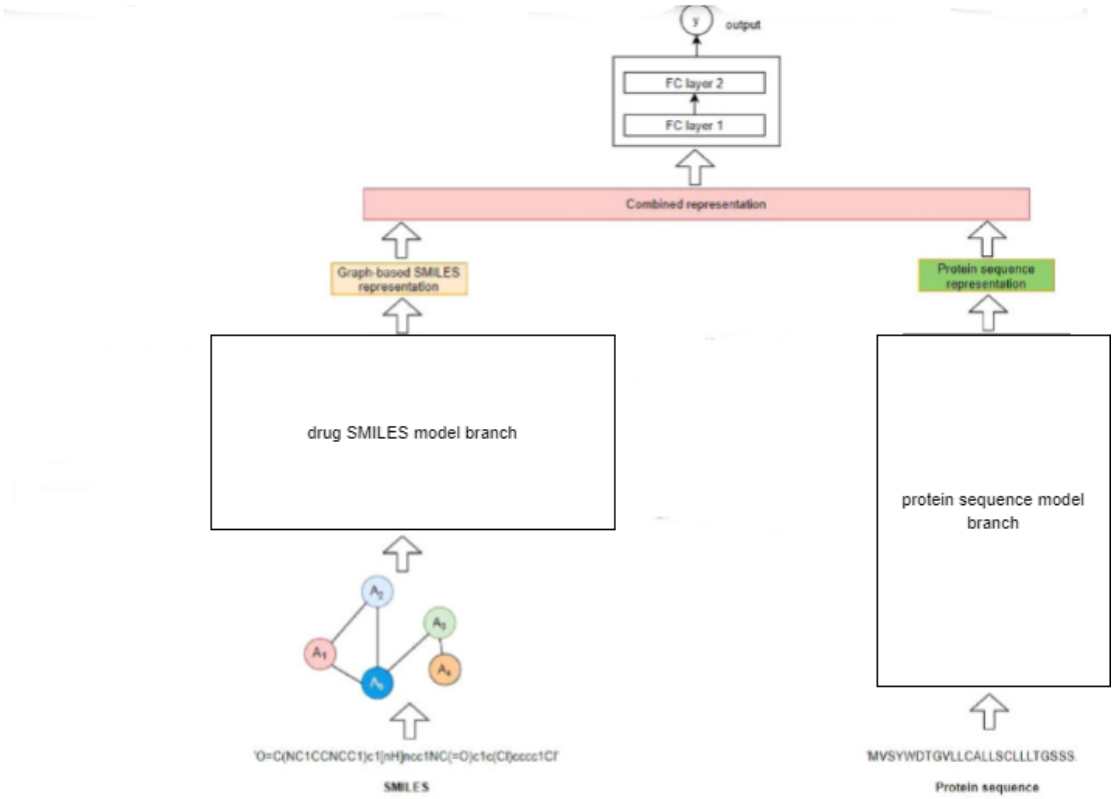
- **code refactoring**:
  second the code was refactored by writing several functions such as Train, Test and generate the URV database in order to write python notebook files that facilitates tuning the parameters and plotting the results by calling these functions instead of just using commands in terminal.

```
datasets = [['davis','kiba'][int(sys.argv[1])]]
modeling = [GINConvNet, GATNet, GAT_GCN, GCNNet][int(sys.argv[2])]
model_st = modeling.__name__

cuda_name = "cuda:0"
if len(sys.argv)>3:
    cuda_name = ["cuda:0","cuda:1"][int(sys.argv[3])]
print('cuda_name:', cuda_name)

TRAIN_BATCH_SIZE = 512
TEST_BATCH_SIZE = 512
LR = 0.0005
LOG_INTERVAL = 20
NUM_EPOCHS = 1000
```

```
def fit(dataset,
        model,
        cuda_name,
        TRAIN_BATCH_SIZE = 512,
        TEST_BATCH_SIZE = 512,
        LR = 0.0005,
        validation_size = 0.2,
        NUM_EPOCHS = 50,
        best_model_flag = True):
    # datasets = [['davis','kiba','urv'][int(sys.argv[1])]]
    # modeling = [GINConvNet, GATNet, GAT_GCN, GCNNet][int(sys.argv[2])]
    modeling = model
    model_st = modeling.__class__.__name__
    print('model_st:', model_st)
    # cuda_name = "cuda:0"
    # if len(sys.argv)>3:
    #     cuda_name = ["cuda:0","cuda:1"][int(sys.argv[3])]
    print('cuda_name:', cuda_name)
```

8 GraphDTA Network architecture

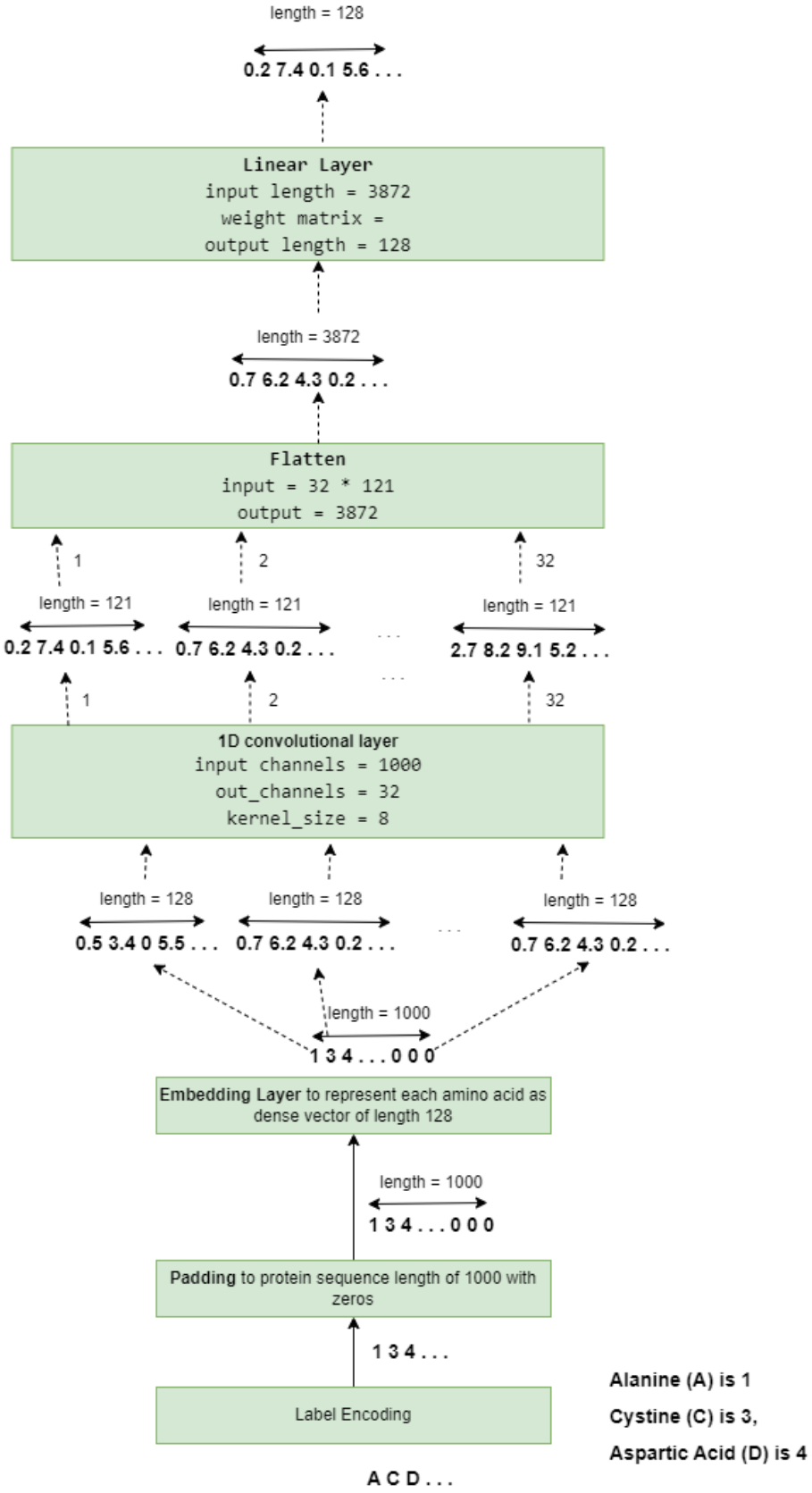
The overall architecture combines the information from the molecular graph and the protein sequence to make a prediction. it is a PyTorch implementation of a Graph Convolutional Network (GCN) based model



The architecture consists of two main branches the protein branch and the drug branch

8.1 Protein sequence branch

operates on the protein sequence input, which is represented as a sequence of integer indices of characters representing amino acids and the steps of processing are as follows:



1. **initial representation**: The protein sequence is a string of ASCII characters which represent amino acids.

2. **label encoding**: Each amino acid type is encoded with an integer based on its associated alphabetical symbol [e.g. Alanine (A) is 1, Cystine (C) is 3, Aspartic Acid (D) is 4 and so on], allowing the protein to be represented as an integer sequence.

3. **padding**: the sequence is cut or padded to a fixed length sequence of 1000 residues. In case a sequence is shorter, it is padded with zero values.

4. **embedding layer**: The protein sequence is passed through an embedding layer (embedding_xt) to convert the integer indices into a dense vector representation.

5. **1D convolutional layer**: The embedded sequence is then passed through a 1D convolutional layer (conv_xt_1) to extract features from the protein sequence.

6. **flattening**: The convolutional features are flattened and passed through a fully connected layer (fc1_xt) to obtain a fixed size output representation.

7. **linear layer**: The line effectively creates a linear layer that will take an input of size 3872 (the flattened output from the convolution) and produce an output of size 128

8.2 SMILES graph branch

operates on the molecular graph representation of the input SMILES. could be any of four models

8.2.1 GCN-based model

it uses graph convolutional(GCN) operater from paper "Semi-supervised Classification with Graph Convolutional Networks." in 2017 and it is impemented in torch geometric as **GCNConv**

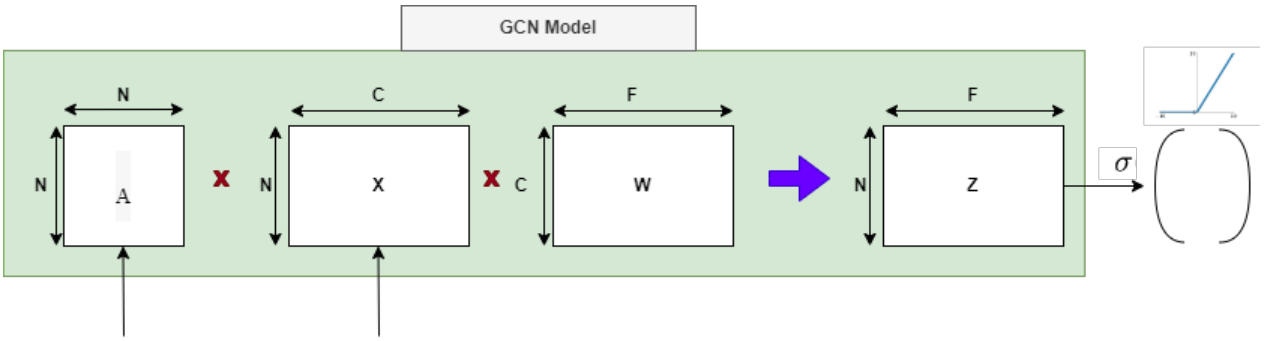


Figure 9: GCN operator

- **A** is the normalized adjacency matrix **N * N**
- **X** is node feature matrix **N * C**
- **W** weight matrix of size **C * F**
- **Z** node-level output matrix **N * F**

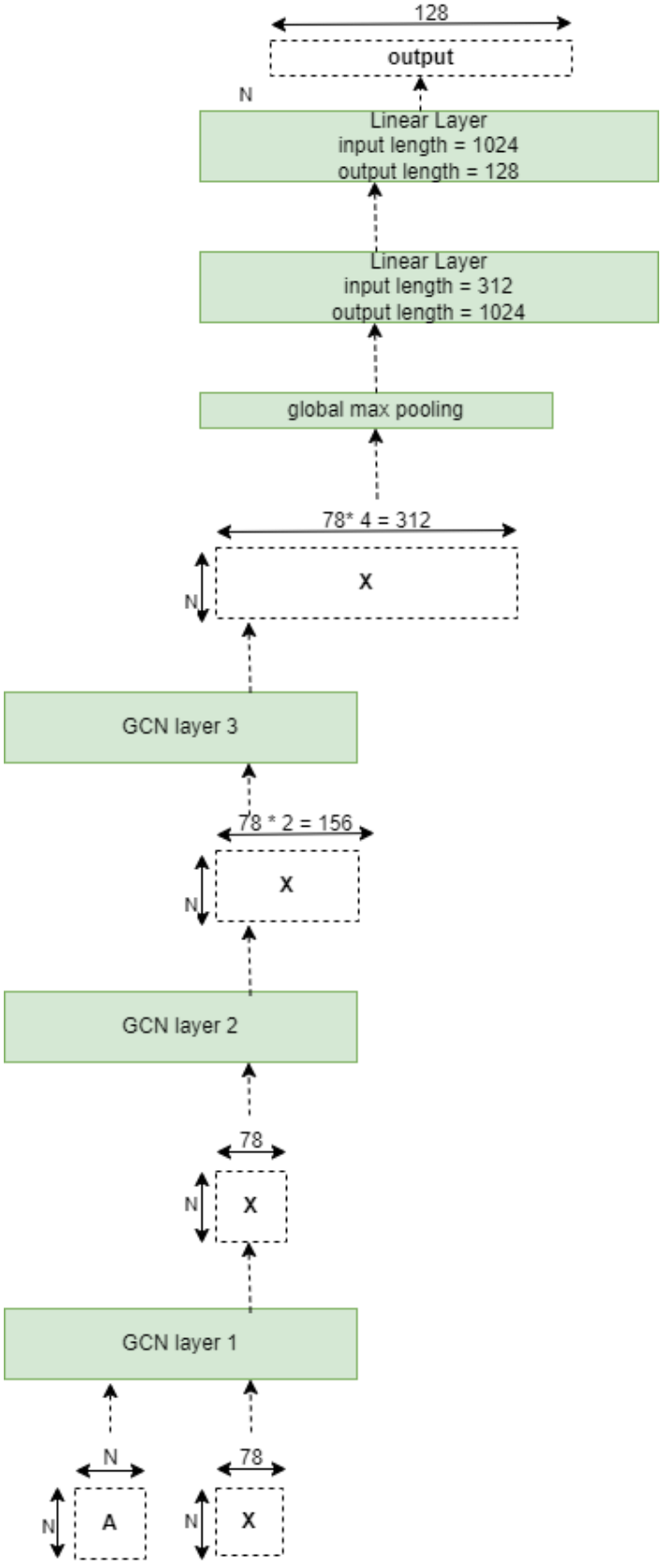
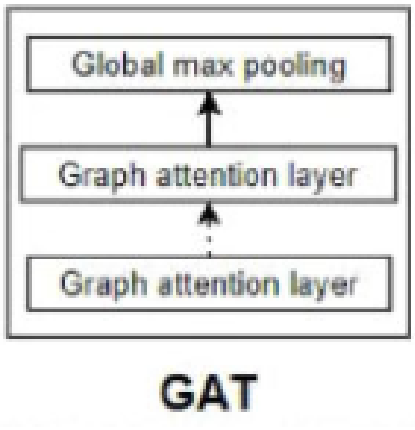


Figure 10: GCN-based model

8.2.2 GAT-based model

1. The model starts with two GAT layers, gcn1 and gcn2, which operate on the input graph data.
2. The first GAT layer, gcn1, takes the initial node features (num_features_xd) and outputs a representation with 10 heads, effectively increasing the feature dimensionality by a factor of 10.
3. The second GAT layer, gcn2, then reduces the feature dimensionality to output_dim.

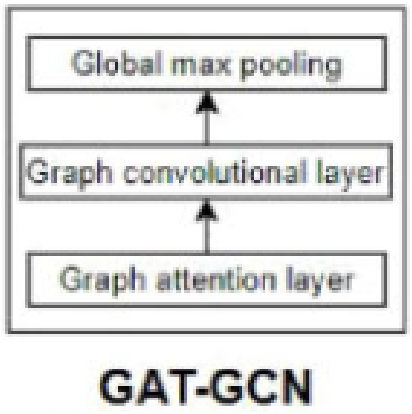
4. After the GAT layers, a fully connected layer fc_g1 is applied to the output of the second GAT layer.



8.2.3 GATGCN-based model

1. The first layer is a GATConv layer, which implements the Graph Attention Network mechanism. It takes the input node features (num_features_xd) and outputs features with the same dimensionality, but with 10 attention heads.

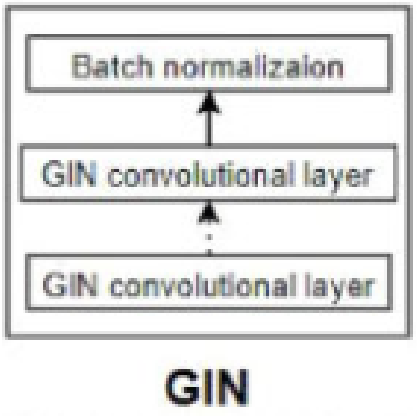
2. The second layer is a GCNConv layer, which applies a standard Graph Convolutional Network operation. It takes the concatenated output of the previous GAT layer (10 times the original number of features) and outputs the same number of features.



8.2.4 GIN-based model

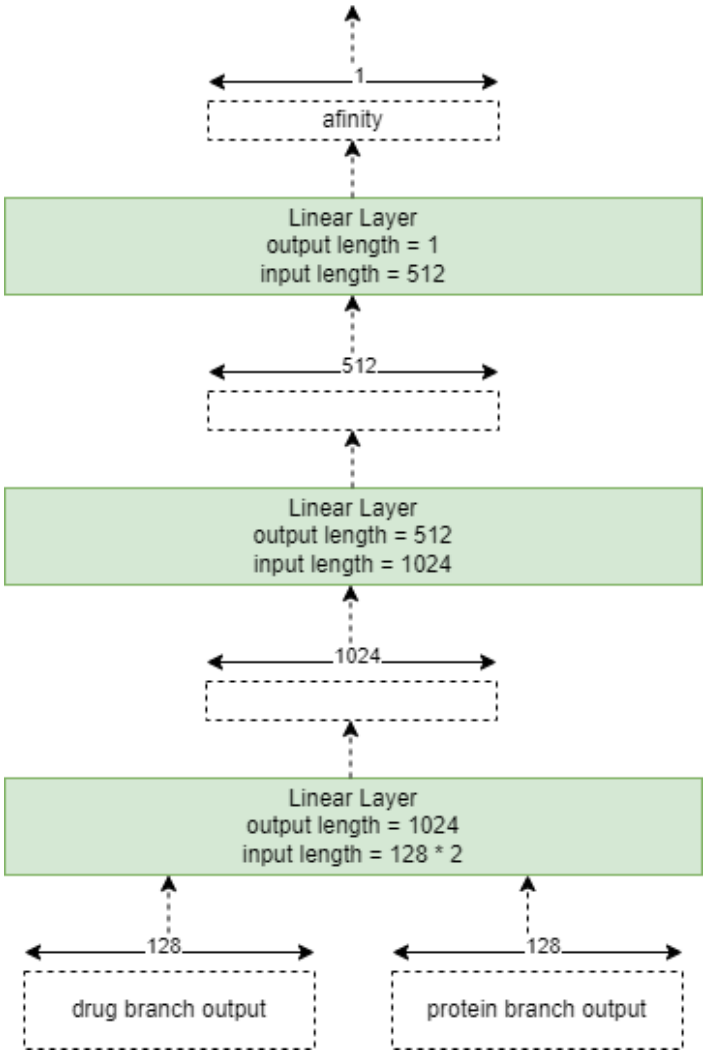
1. The model has 5 GINConv layers, each with a sequential neural network (nn1, nn2, ..., nn5) that consists of a Linear layer, a ReLU activation, and another Linear layer.

2. Each GINConv layer is followed by a BatchNorm1d layer to normalize the output.



8.3 combined fully connected layers

After obtaining the output representations from the two branches -protein and drug -, the model concatenates them and passes the combined features through a regular neural network composed of two fully connected layers linear layers to produce the final output. The model uses ReLU activations, dropout for regularization after each linear layer.



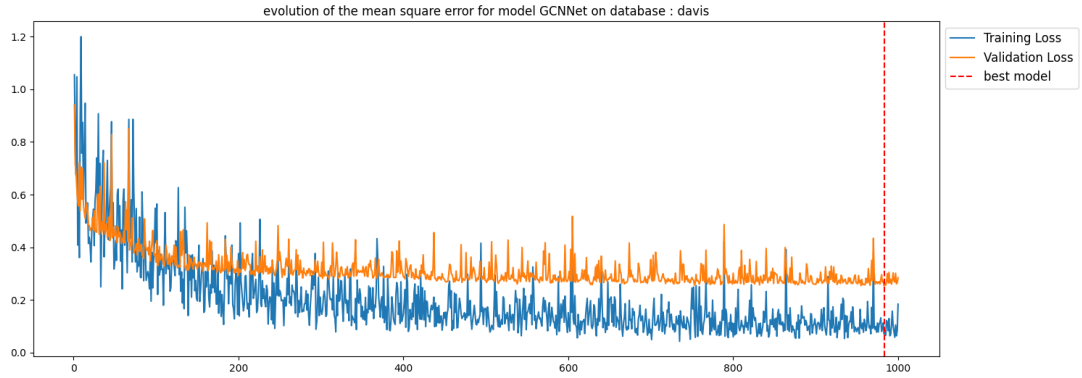
1. **drug branch output**: tensor of length 128 of the output of the drug graph branch
2. **protein branch output**: tensor of length 128 of the output of the protein branch
3. **Linear layer 1**: takes as input the concatenation of two tensors of length 256 - 128 each that represent the output of drug and protein branch - and produces a tensor of length 1024 by applying weight matrix 1024 * 256 followed by RELU activation function
4. **Linear layer 2**: take as input a tensor of length 1024 which is the output of the first linear layer and produces a tensor of length 512 by applying weight matrix 512 * 1024 followed by RELU activation function
5. **Linear layer 3**: take as input a tensor of length 512 which is the output of the second linear layer and produces a tensor of length 1 which is the output affinity by applying weight matrix 1 * 512

9 Results

9.1 for davis dataset

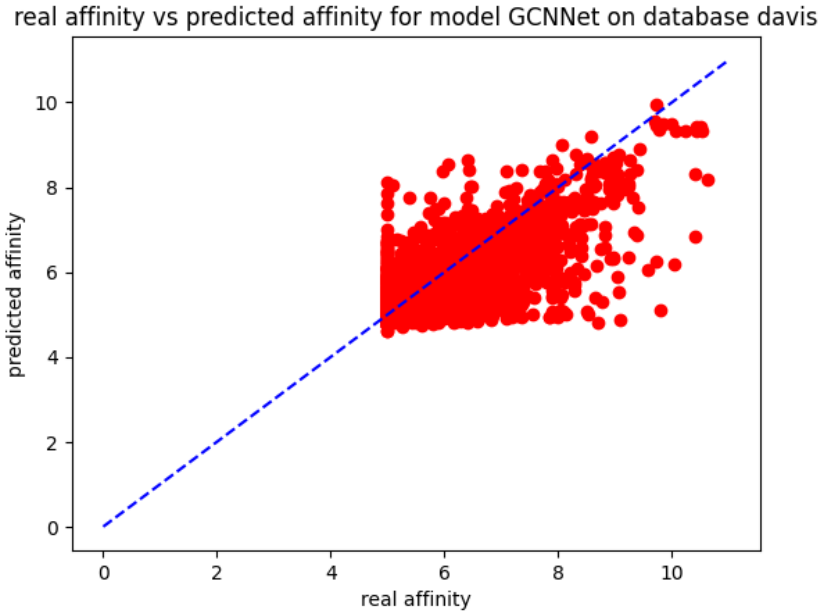
9.1.1 train and test of GCN-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



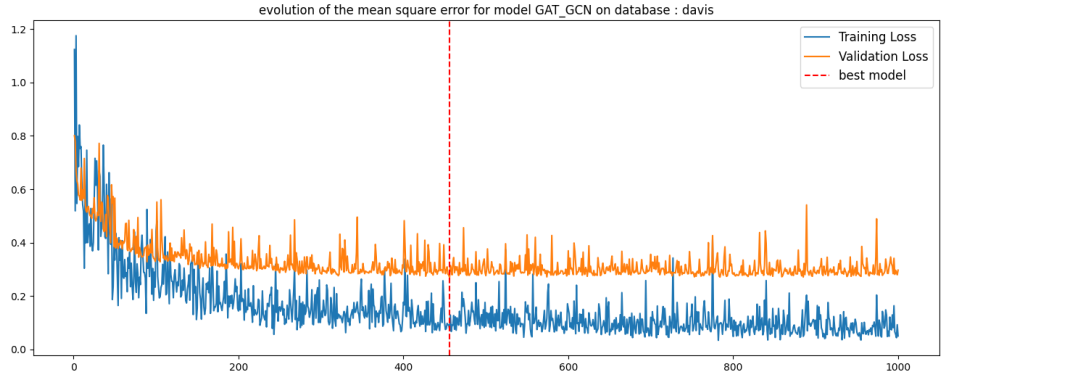
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	20036
validation size	5010
validation percentage	20.0 %
MSE	0.25293395

and the resulting predicted-actual curve is as follows:



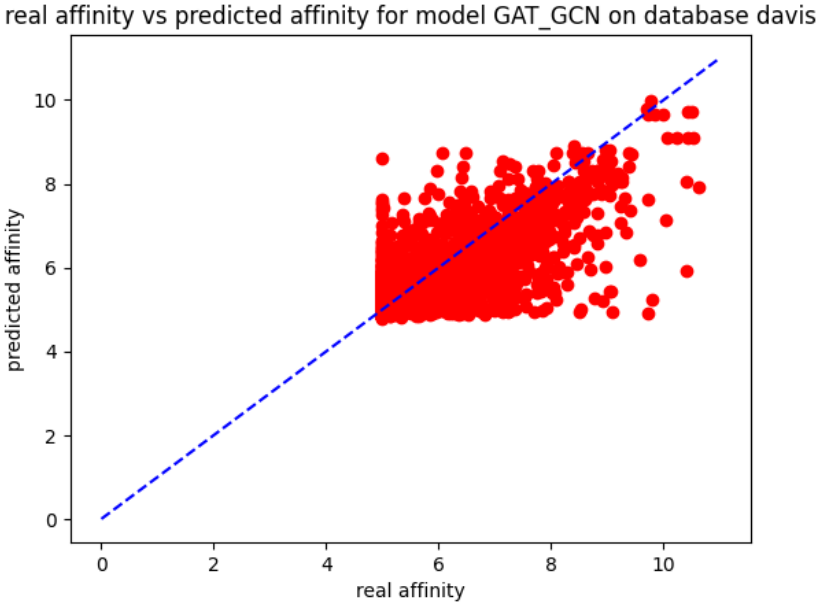
9.1.2 train and test of GATGCN-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



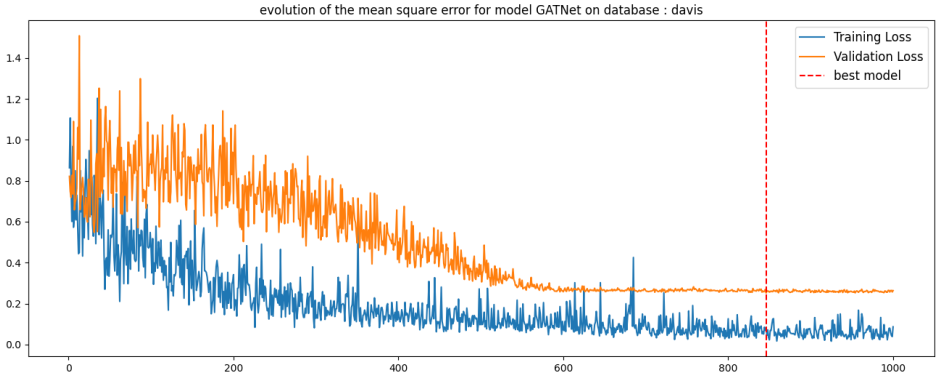
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	20036
validation size	5010
validation percentage	20.0 %
MSE	0.27028632

and the resulting predicted-actual curve is as follows:



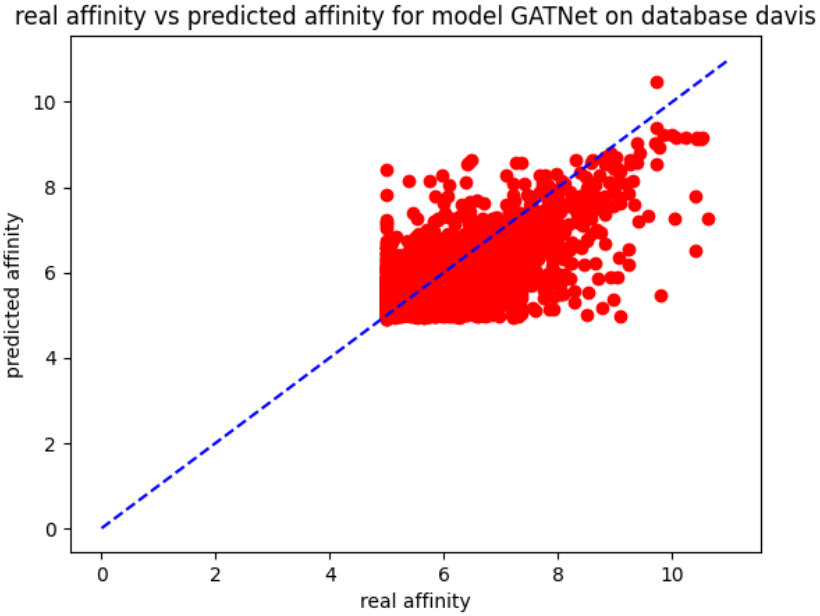
9.1.3 train and test GAT-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



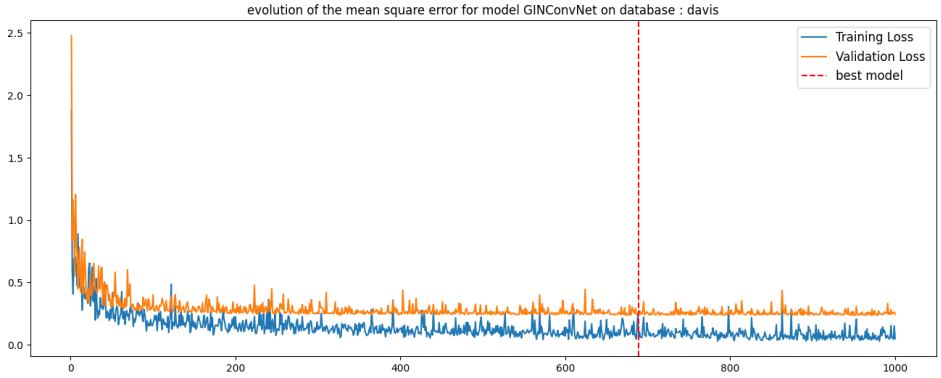
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	20036
validation size	5010
validation percentage	20.0 %
MSE	0.2513844

and the resulting predicted-actual curve is as follows:



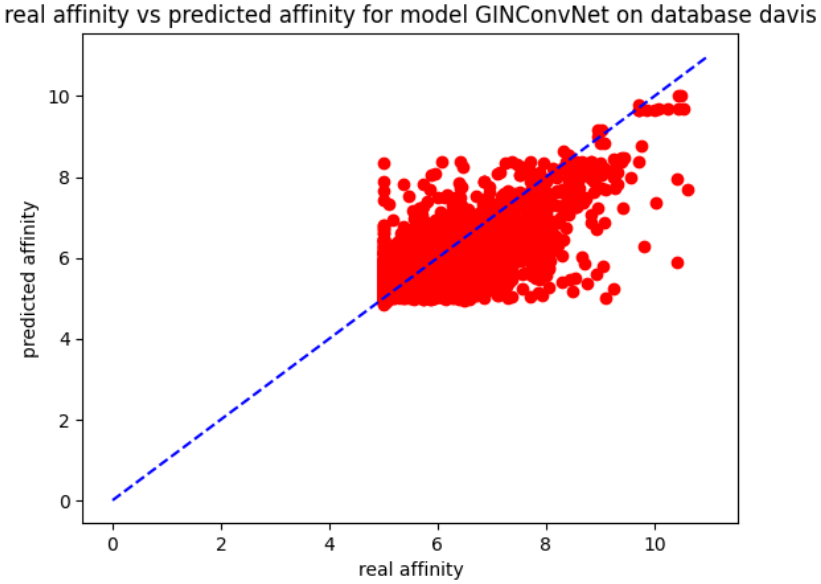
9.1.4 train and test GINCONV-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	20036
validation size	5010
validation percentage	20.0 %
MSE	0.23514226

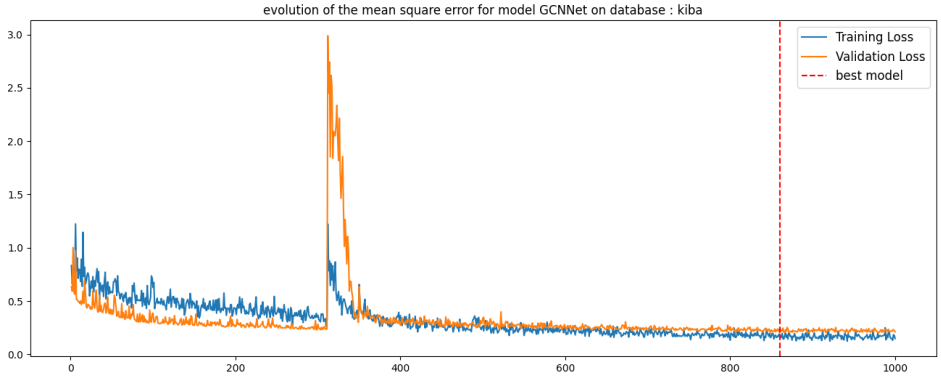
and the resulting predicted-actual curve is as follows:



9.2 for kiba dataset

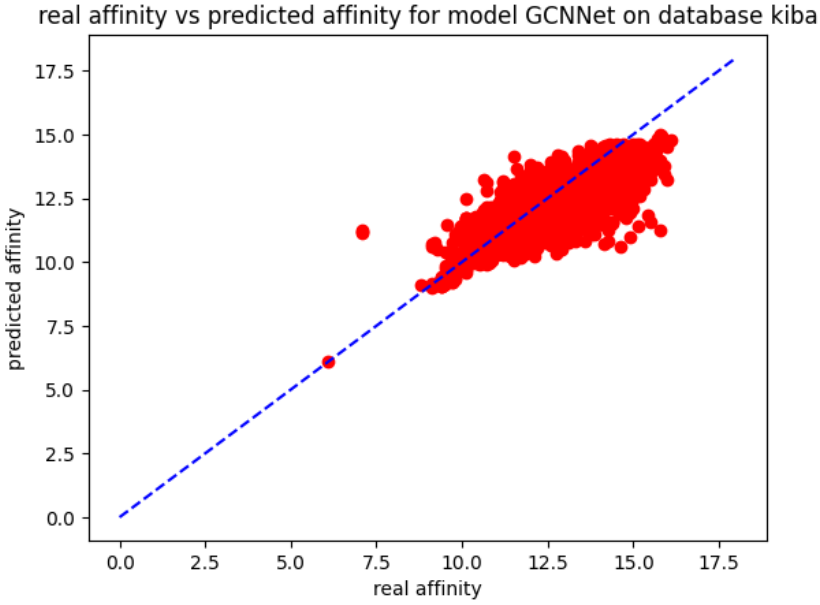
9.2.1 train and test GCN-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



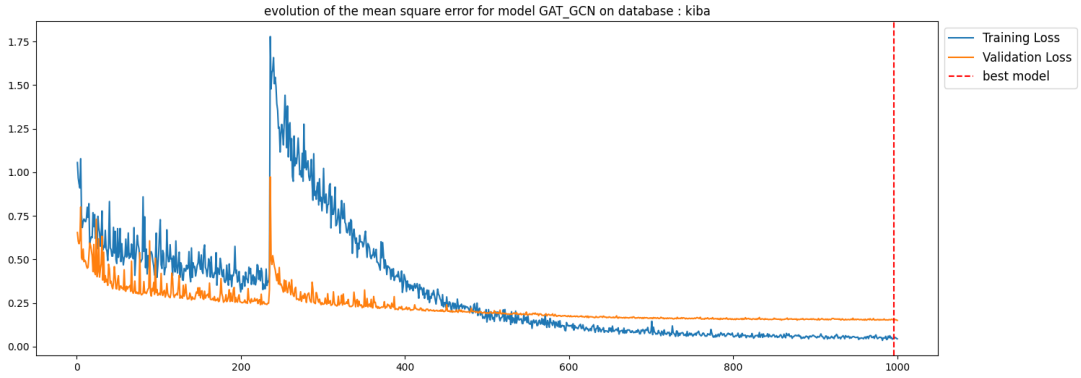
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	78836
validation size	19709
validation percentage	20.0 %
MSE	0.2024536

and the resulting predicted-actual curve is as follows:



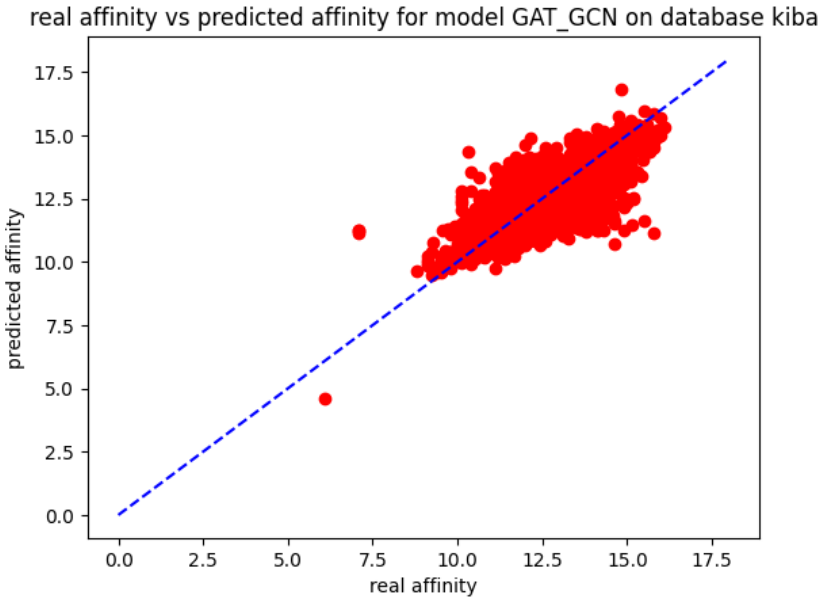
9.2.2 train and test GATGCN-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



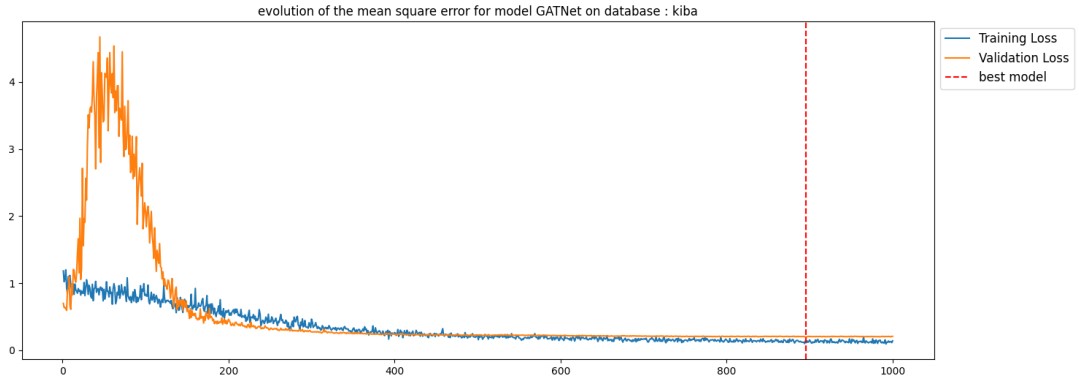
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	78836
validation size	19709
validation percentage	20.0 %
MSE	0.15026996

and the resulting predicted-actual curve is as follows:



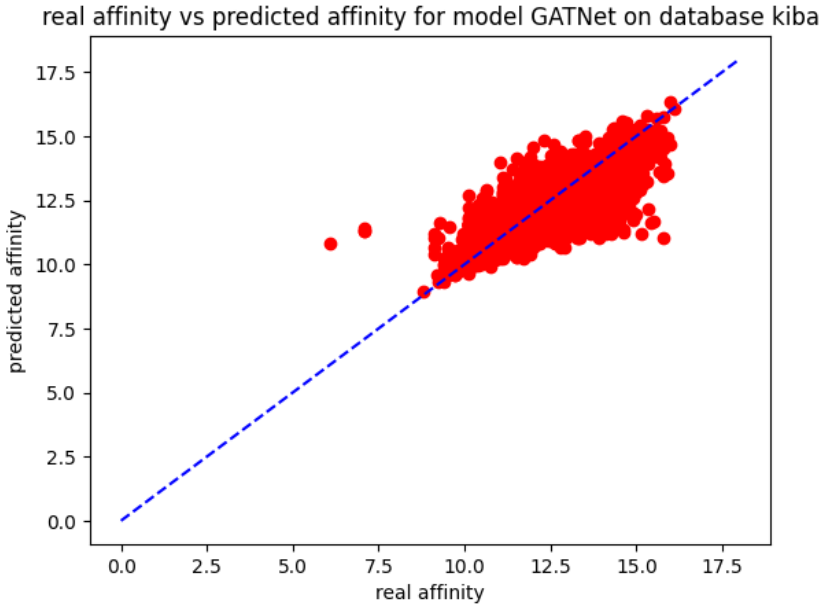
9.2.3 train and test GAT-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



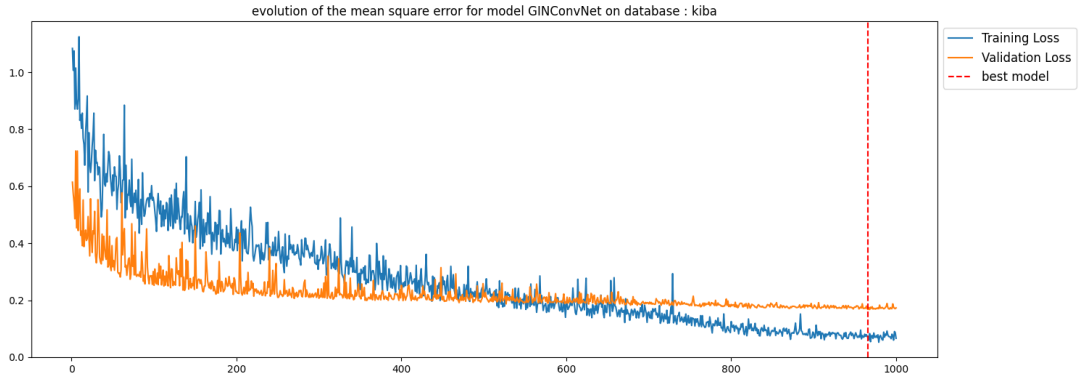
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	78836
validation size	19709
validation percentage	20.0 %
MSE	0.19964518

and the resulting predicted-actual curve is as follows:



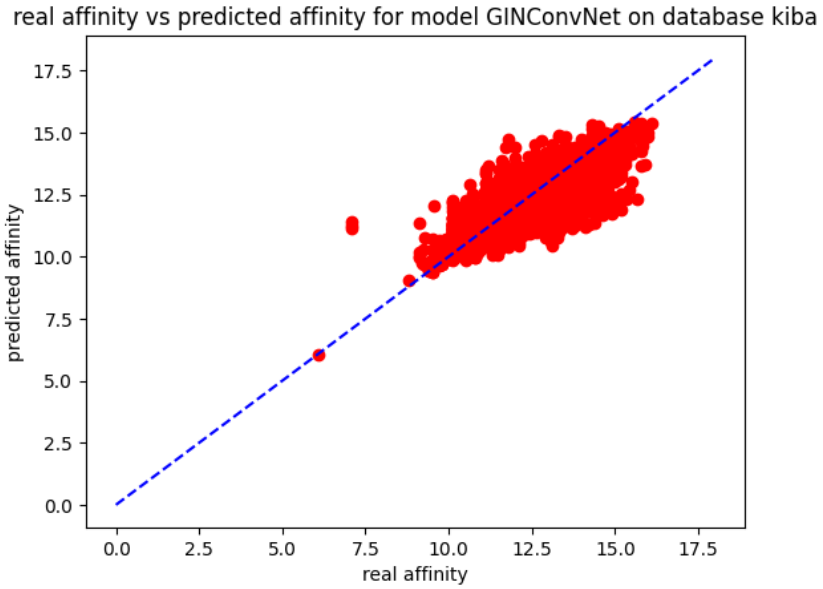
9.2.4 train and test GINCONV-based model

by training the network with the parameters suggested by the paper shown we obtain the error loss curve for training and validation set



optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	512
train size	78836
validation size	19709
validation percentage	20.0 %
MSE	0.1673416

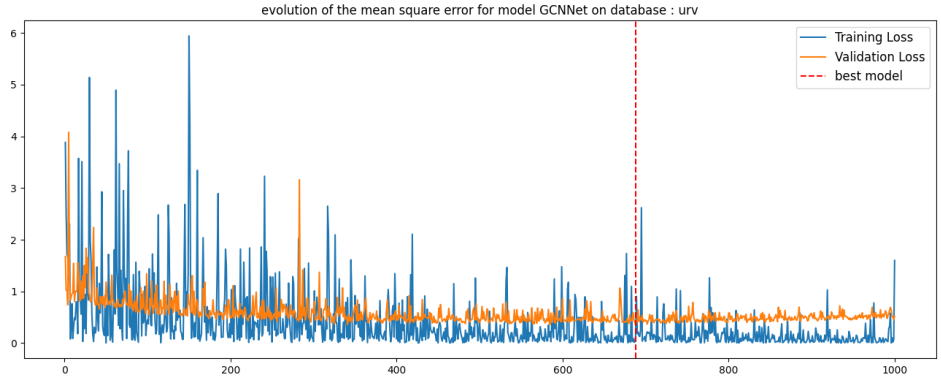
and the resulting predicted-actual curve is as follows:



9.3 for URV dataset

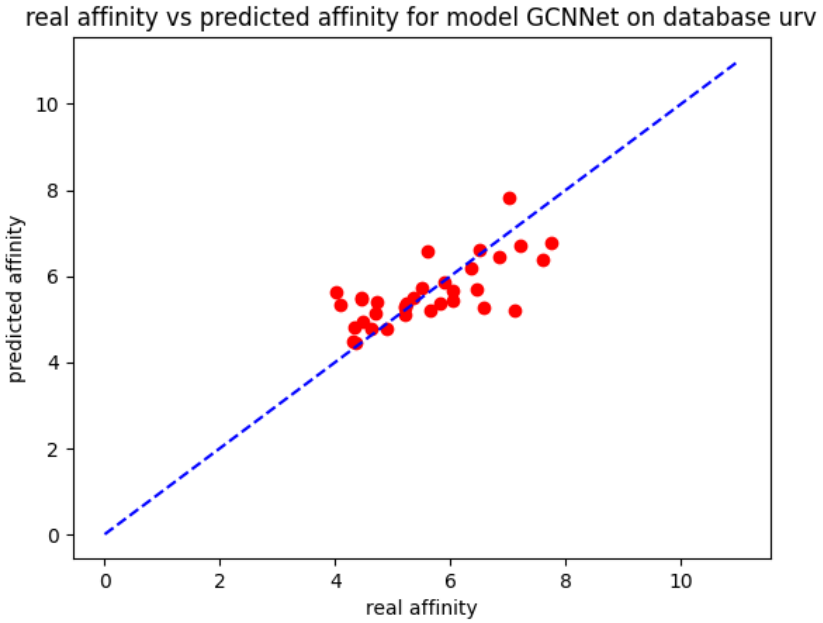
9.3.1 train and test GCN-based model

by training the network while tuning the parameters starting with those suggested by the paper shown we obtain the error loss curve for training and validation set



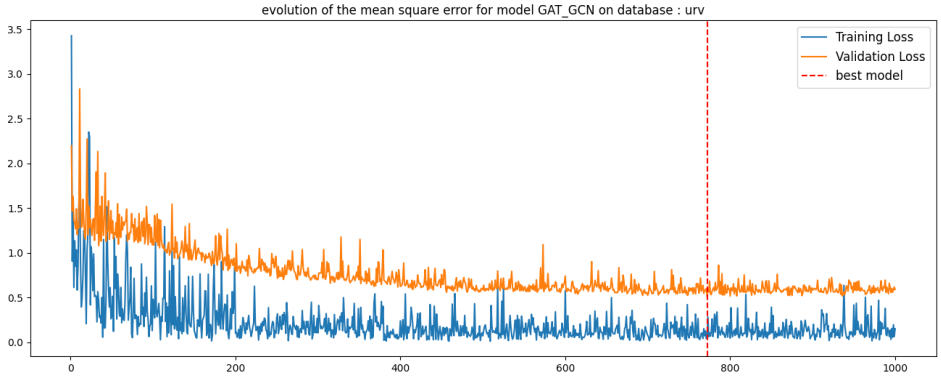
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	4
train size	238
validation size	60
validation percentage	20.0 %
MSE	0.35485

and the resulting predicted-actual curve is as follows:



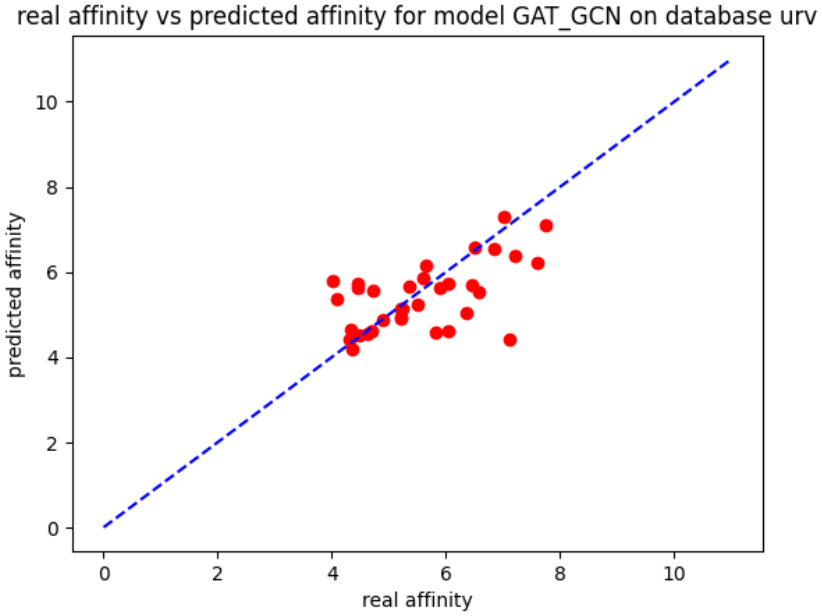
9.3.2 train and test GATGCN-based model

by training the network while tuning the parameters starting with those suggested by the paper shown we obtain the error loss curve for training and validation set



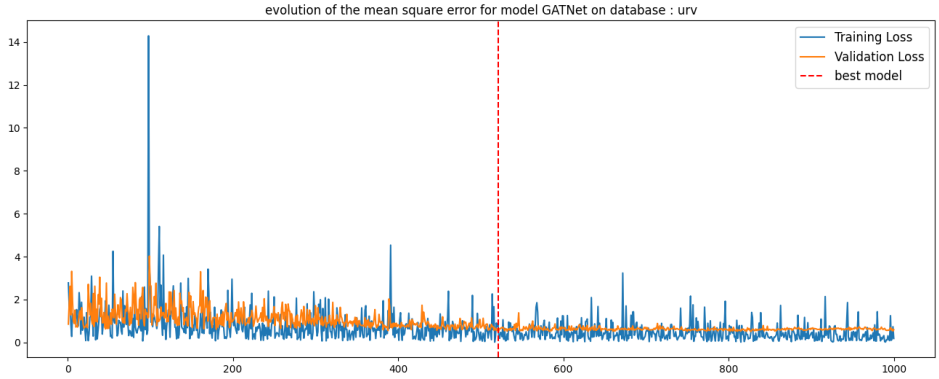
optimizer	ADAM
learning rate	0.0001
epochs	1000
train batch size	8
train size	238
validation size	60
validation percentage	20.0 %
MSE	0.51364166

and the resulting predicted-actual curve is as follows:



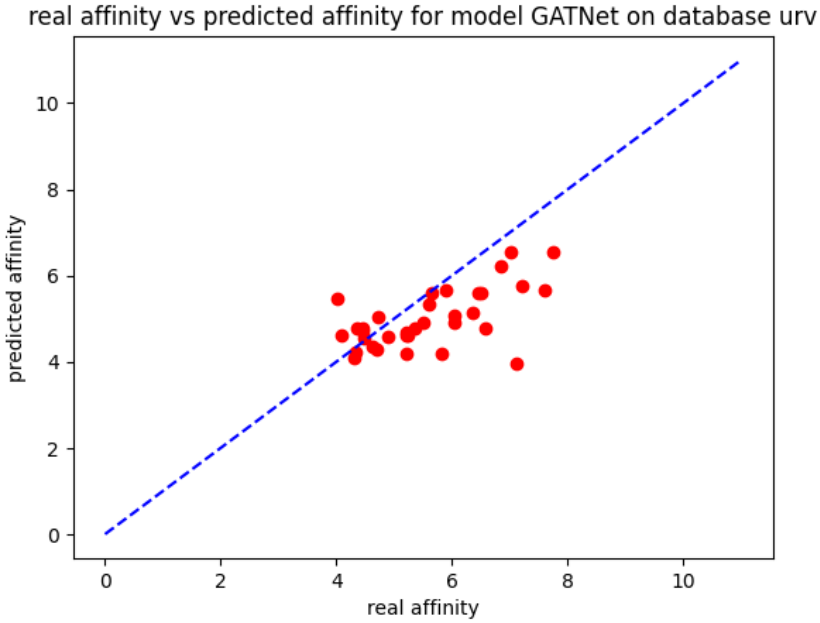
9.3.3 train and test GAT-based model

by training the network while tuning the parameters starting with those suggested by the paper shown we obtain the error loss curve for training and validation set



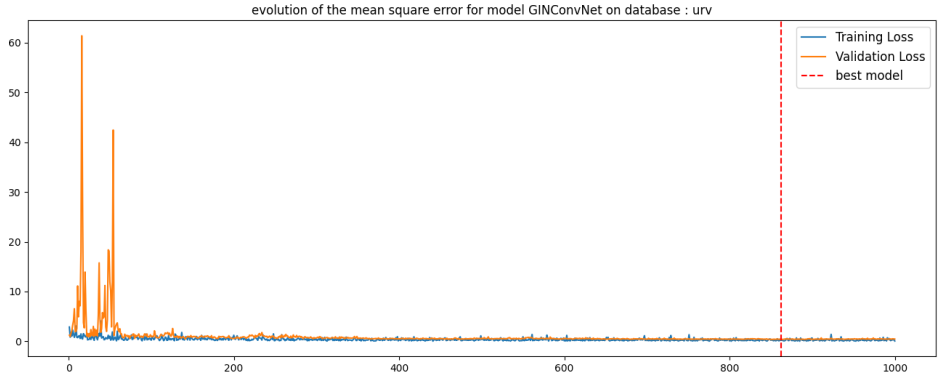
optimizer	ADAM
learning rate	0.0005
epochs	1000
train batch size	4
train size	208
validation size	90
validation percentage	30.0 %
MSE	0.50838196

and the resulting predicted-actual curve is as follows:



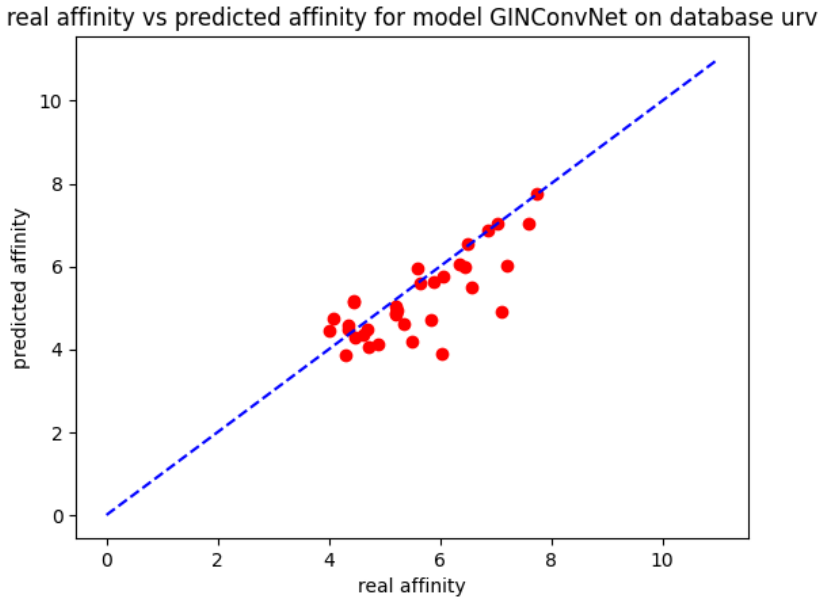
9.3.4 train and test GINCONV-based model

by training the network while tuning the parameters starting with those suggested by the paper shown we obtain the error loss curve for training and validation set



optimizer	ADAM
learning rate	0.0001
epochs	1000
train batch size	8
train size	238
validation size	60
validation percentage	20.0 %
MSE	0.3247614

and the resulting predicted-actual curve is as follows:



9.4 Summary and conclusion

the results are summarized in the following table

model	dataset	MSE in paper	MSE obtained
GCN-based	davis	0.254	0.25
GATGCN-based	davis	0.245	0.27
GAT-based	davis	0.232	0.25
GINCONV-based	davis	0.229	0.24
GCN-based	kiba	0.179	0.2
GATGCN-based	kiba	0.147	0.15
GAT-based	kiba	0.139	0.2
GINCONV-based	kiba	0.139	0.17
GCN-based	URV	-	0.35
GATGCN-based	URV	-	0.51
GAT-based	URV	-	0.51
GINCONV-based	URV	-	0.32

it was noticed that best MSE was obtained in results of kiba dataset then davis and finally the results of URV dataset - either in paper or in thesis - and this might be due to the following reasons:

1. the sample size of the data including train and test sets is largest in kiba then davis followed by URV and this helps the network learn more robust and generalizable patterns, thus make accurate predictions on unseen data..

2. kiba dataset has the best diversity specially for the target proteins as it integrates different bioactivity scores and this is crucial to train a model capable of generalizing to unseen drug-target pairs.

References

[1] [Thin et al.] Thin Nguyen; Hang Le; Thomas P. Quinn; Tri Nguyen; Thuc Duy Le; and Svetha Venkatesh, "GraphDTA: predicting drug–target binding affinity with graph neural networks", Journal Bioinformatics, Volume 37, Issue 8, March 2021, Pages 1140–1147 Available at https://academic.oup.com/bioinformatics/article/37/8/1140/5942970.

[2] Related data, pre-trained models and source code in [1] are publicly available at https://github.com/thinng/GraphDTA.

[3] the repository forked from [2] to perform experiments in this thesis available at https://github.com/YoussefEzz/GraphDTA_forked.

[4] PDB website: https://www.rcsb.org/.

[5] Bio.PDB Biopython module: https://biopython.org/wiki/The_Biopython_Structural_Bioinformatics_FAQ.

[6] chemspider website: https://www.chemspider.com/.

[7] rdkit library: https://www.rdkit.org/docs/source/rdkit.Chem.html.

[8] [He et al.] Tong He; Marten Heidemeyer; Fuqiang Ban; Artem Cherkasov and Martin Ester "SimBoost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines",

Journal of Cheminformatics, 9, Article number: 24 (2017)
Available at https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0209-z.

[9] [Ozturk et al.] Hakime Ozturk; Arzucan Ozgur and Elif Ozkirimli "DeepDTA: deep drug–target binding affinity prediction",
Journal of Bioinformatics, Volume 34, Issue 17, September 2018, Pages i821–i829,
Available at https://academic.oup.com/bioinformatics/article/34/17/i821/5093245.

[10] [Ozturk et al.] Hakime Ozturk; Arzucan Ozgur and Elif Ozkirimli "WIDEDTA: PREDICTION OF DRUG-TARGET BINDING AFFINITY",
Available at https://arxiv.org/abs/1902.04166.

[11] Berman, H.M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N., Weissig, H., Shindyalov, I.N., Bourne, P.E. (2000). The Protein Data Bank. Nucleic Acids Research, 28(1), 235-242. [doi:10.1093/nar/28.1.235]

[12] Zdrazil, B., Felix, E., Hunter, F., et al. (2024). The ChEMBL Database in 2023: a drug discovery platform spanning multiple bioactivity data types and time periods. Nucleic Acids Research, 52(D1), D1180–D1192. [doi:10.1093/nar/gkad1004]

[13] Gilson, M. K., Liu, T., Baitaluk, M., et al. (2016). BindingDB in 2015: A public database for medicinal chemistry, computational chemistry and systems pharmacology. Nucleic Acids Research, 44(D1), D1045–D1053. [doi:10.1093/nar/gkv1072]