jul-22

MACHINE LEARNING METHODS FOR DRUG REPURPOSING

Machine Learning Methods for Drug Repurposing

PROYECTO

presentado para optar

al Título de Grado en Ingeniería en Sistemas de Telecomunicación

por

Mikel Casals Baena

bajo la supervisión de

Idoia Ochoa

Donostia-San Sebastián, julio de 2022

Mikel Casals Baena

Tecnun
Universidad de Navarra
ESCUELA DE INGENIERÍA
INGENIARITZA ESKOLA
SCHOOL OF ENGINEERING

Tecnun
Universidad
de Navarra
ESCUELA DE INGENIERÍA
INGENIARITZA ESKOLA
SCHOOL OF ENGINEERING

Proyecto Fin de Grado

INGENIERIA EN SISTEMAS DE TELECOMUNICACIÓN

MACHINE LEARNING METHODS FOR DRUG REPURPOSING

Mikel Casals Baena

Donostia-San Sebastián, julio de 2022

Pº Manuel Lardizabal, 13. 20018 Donostia-San Sebastián, Gipuzkoa
Tel. 943 219 877 · Fax 943 311 442 · www.tecnun.es

Acknowledgments

I would like to dedicate this section to all the people that have contributed directly or indirectly to this project.

Firstly, I want to thank Idoia Ochoa and Mikel Hernaez for giving me the opportunity to work on their team. I also want to thank them for all the personal advice they have given me whenever I had doubts about my future. During the project I have worked alongside very talented people from different academic backgrounds, and it has been a very enriching experience. I would like to thank all of them and specially Jesús de la Fuente, for helping me whenever I had a problem or I did not understand something.

I am also thankful to all the people that have been part of my journey at Tecnun during these 4 years. Thanks to all the friends that I have met at university that have helped me become the person I am now. To my counsellor, Héctor Solar, for mentoring me whenever I felt lost. Also, thanks to my classmates, Bruno, Carolina and Viktor. I would not have finished the degree if it had not been for their support.

To my parents and my sister Leire, thank you for your unconditional support, for always believing in me and for making this experience possible.

A special mention to all the friends I met during my stay in Montreal. It was a life changing experience for me, and it would not have been the same without them. Thank a lot to all of them.

Thanks as well to the rest of my family and friends that I have not been able to mention.

Contents

List of Figures

List of Tables

Abstract

Drug Repurposing consists on using already approved drugs to treat other diseases. This is done by identifying new targets that the drug may have. To accelerate this long and costly process, computational methods have been developed to predict drug-target interactions (DTIs).

Recently, Machine Learning has had a tremendous impact on many scientific fields including DTI prediction. In this project, two state-of-the-art methods named MolTrans and HyperAttentionDTI are described and compared on 8 different datasets. Moreover, each dataset is divided according to 3 different splits, so that the generalization of the methods with respect to drugs and proteins is tested.

Graphs are a type of data structures that have nodes and edges connecting them. They can model complex systems accurately such as DTI networks. In recent years, Graph Machine Learning methods have been developed to improve on conventional models. Node embedding techniques and Graph Neural Networks are introduced as ways to transform nodes into vector embeddings to be able to make predictions on them.

While the two analyzed methods offer competitive results, Graph Machine Learning can take advantage of the expressiveness that graphs have in order to make more accurate predictions.

Chapter 1

Introduction

Drug discovery is the process of finding a new drug molecule that is able to inhibit or activate a target to have a therapeutic effect in a disease state [1]. A target is a biological entity, such as proteins, genes or RNA.

The drug discovery process starts with the identification of a target. In this step, research is done to develop a first hypothesis on which target is more likely to be inhibited. Further validation may be required after identifying the target using either *in vitro* (outside a living organism) or *in vivo* (within a living organism) tools.

Once the target has been validated, the next step is to develop compound screening assays. In these assays, a number of possible drug compounds are screened against the target to identify molecules that interact with it as desired. This search is part of the lead discovery phase in which a drug-like small molecule is found. This will then lead to preclinical and clinical development. If successful, the discovered molecule will become a medicine to be sold to the market.

Drug discovery is a long and costly process. For instance, the average cost for a pharmaceutical company to develop a new drug molecule is estimated to be over $1-2 billion, and it can take over 10-15 years on average from the target identification phase until the drug is ready to be sold [2]. In recent years, many pharmaceutical companies have shifted to new strategies such as drug repurposing to overcome the aforementioned limitations.

1.1 Drug Repurposing

Traditionally, the focus of drug discovery was based on the "one target, one drug" model [3]. Drug designers used to produce drugs that only interacted with a single target, and tried to avoid drugs that interacted with multiple targets. However, this approach has resulted in a decrease in productivity by pharmaceutical companies. With the increased understanding of

complex diseases in recent years, the "multi-target, multi-drug" model is becoming widely accepted. This is known as polypharmacology. Therefore, drug repurposing has received increasing attention recently.

Drug repurposing is based on using already approved drugs as new therapies for other diseases [4]. There are already numerous known interactions between drugs and targets available in databases. However, the number of unseen pairs is proportionately bigger. Trying to prove experimentally that a drug molecule interacts with a target is also an expensive and time-consuming process.

Therefore, drug-target interaction (DTI) prediction plays a key role in accelerating this process by screening new possible pairs effectively and reducing the costs of the drug discovery process.

1.2 Computational methods for DTI prediction

There are three main types of computational approaches that are currently used for DTI prediction, namely ligand-based, docking simulation and chemogenomic approaches [5].

Ligand-based approaches are based on the chemical similarity principle [6]. This principle states that molecules that are similar usually have alike physico-chemical properties and thus, bind to analogous targets. Therefore, these methods predict which targets a drug may have by comparing it to already proven DTIs. An example of a ligand-based method is Quantitative Structure Activity Relationship (QSAR) [7]. The drawback that these approaches have is that they do not generalize well when the number of known pairs is limited.

Docking simulation approaches use the three-dimensional structure of drugs and targets to predict their interaction [8]. Therefore, these methods rely on the availability of drug and protein structures. Moreover, docking simulation can be inefficient in time if the structures are complex.

Lastly, chemogenomic approaches use chemical structure data from drugs and genomic sequence data from proteins to perform DTI prediction [9]. Machine Learning-based methods are a type of these approaches.

1.3 Objective and structure of the project

The main objective of the project is to evaluate 2 state-of-the-art machine learning methods for DTI prediction to provide a fair comparison between them. Eight different datasets will be used in order to perform this comparison. Moreover, an brief introduction to graph machine learning methods will be provided to see how they would compare with conventional machine learning models.

The project is structured as follows. In Chapter 2, an introduction to some fundamental concepts in machine learning is done. These concepts are useful to understand two state-of-the-art methods that are explained in Chapter 3. In Chapter 4, the results obtained in each of the methods for the eight datasets are analyzed and compared. Chapter 5 introduces graphs and explains node embedding techniques and Graph Neural Networks as a way to improve on conventional machine learning methods. Finally, in Chapter 6, an overall conclusion is made on the whole project.

Chapter 2

Preliminaries

To understand the machine learning methods that are going to be analyzed in this project, a brief introduction to some fundamental concepts is needed.

2.1 Machine Learning

A machine learning algorithm is an algorithm that is able to learn from data [10]. There are many different tasks in machine learning such as regression, classification or machine translation.

Most machine learning algorithms are based on the concept of optimization, in which the objective is to minimize a function $f(\mathbf{x})$, by finding the optimal $\mathbf{x}$. The function to be minimized is normally referred to as the cost function.

In order to minimize the cost function, a technique called gradient descent is used. Fundamentally, gradient descent uses derivatives to change $\mathbf{x}$ in the correct direction so as to find the global minimum of the function.

One of the most important challenges when designing a machine learning algorithm is that it should be able to make good predictions to new input data on which it has not been trained. This is also known as generalization.

DTI prediction is mainly modeled as a classification task. This means that the machine learning algorithm has to predict which class the introduced input data belongs to. For instance, given drug and protein input data, the algorithm is supposed to predict whether the drug and protein interact or not. This is an example of a binary classification task.

2.2 Feedforward Neural networks

A feedforward neural network, also known as multilayer perceptrons (MLPs) is the fundamental model of deep learning. The term feedforward comes from the fact that given an input vector x, this model flows the information in one direction until the prediction is done. An example of a neural network can be seen in Figure 1.

These models can have many hidden layers before the output layer. Each layer is formed by a specific number of nodes. Each node has weights associated to them and the objective is to transform the input data by multiplying it with the node's weights. Then, a non-linear function is applied to it such as sigmoid.The number of layers and the number of nodes in each layer are both hyperparameters of the model and need to be optimized to the task that is being solved.

As any machine learning models, the weights of the layers' nodes are trained using a training data set.

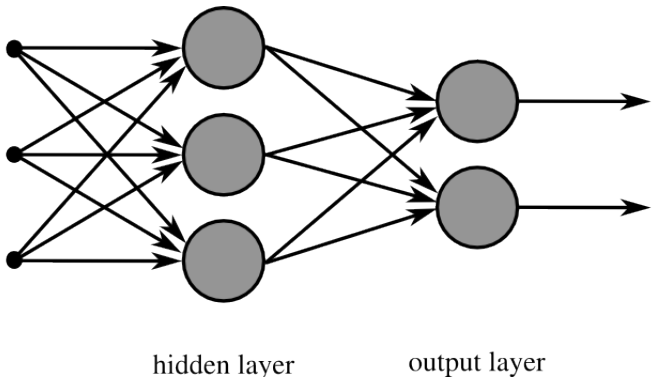

Figure 1: Feedforward neural network.

2.3 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a specific type of neural network that has competitive predictive performance in various task related to images, but it can also be applied to any data that has a grid-like structure. An example of a CNN can be visualized in Figure 2.

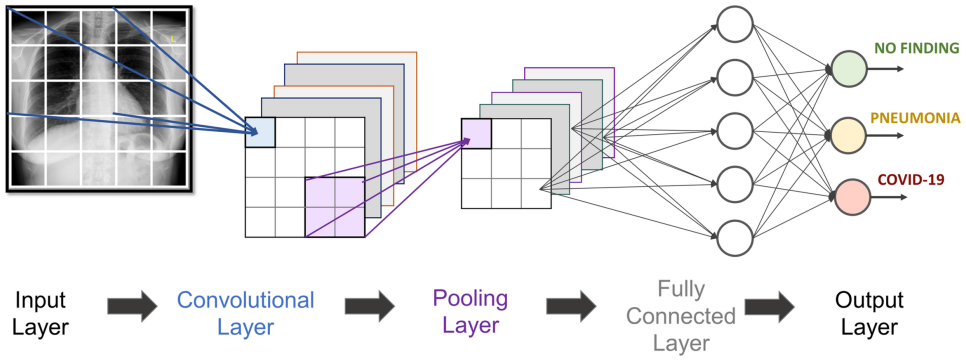

Figure 2: Convolutional Neural Network [11].

These type of networks have the following layers:

- **Convolutional layer:** This layer is mainly based on the convolution operation. Every convolutional layer is composed of a set of filters that have relatively small width and height. These filters are slided through the input data and compute the dot product at each point. Then, a nonlinear activation function such as ReLU is applied, producing a so-called feature map. The feature map can be interpreted as the responses of the filter at every spatial position of the data.
- **Pooling layer:** This layer is used after each convolutional layer in order to reduce the dimension of the feature map. The most common type is max pooling, where the feature map is taken in small grids and the maximum value in each grid is taken.

CNN can have many concatenated convolutional and pooling layers. In the end, a feedforward neural network is typically applied by flattening the output of the last pooling layer, in order to make predictions.

2.4 Transformers

Transformers have demonstrated high performance in many natural language processing tasks [12]. They are based on the Sequence-to-Sequence (Seq2Seq) model [13], which is basically a neural network that transforms an input sequence into another sequence.

Seq2Seq models are composed by an Encoder and a Decoder. An input sequence is fed into the Encoder and maps it to a n-dimensional vector. This vector is taken by the Decoder and turns it into an output sequence. For example, if the input sequence is a sentence in English, the Encoder transforms it into a vector, and the Decoder outputs the sentence translated to another language. Like any other machine learning model, these models need data in order to be trained.

Moreover, transformers make use of attention. An attention mechanism models, for each of

the elements of a sequence, the importance that the rest of the elements have on it.

Although the details of how transformer work are not necessary for this project, in Figure 3, the model architecture can be seen. It is divided into 2 parts, the left one being the encoder and the right one the decoder. Simply put, the encoder and decoder are a stack of attention mechanisms and feedforward neural networks, that given an input produce an output embedding.

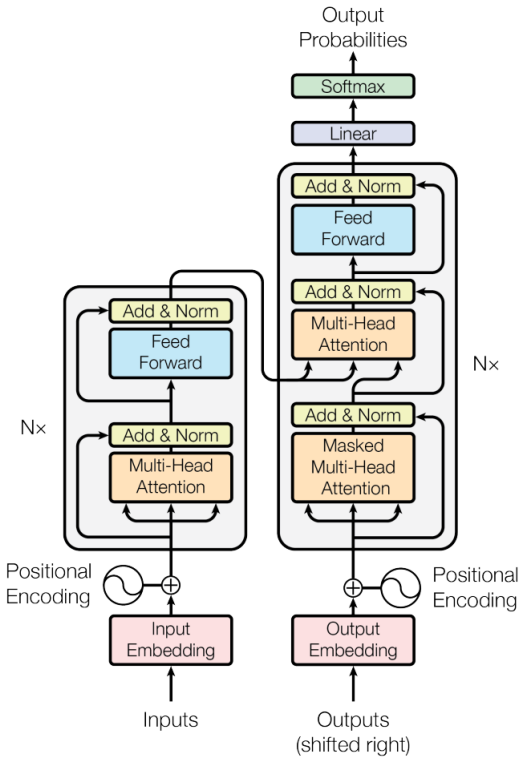

Figure 3: Transformer model architecture [12].

Chapter 3

Evaluation of state-of-the-art methods

3.1 MolTrans

MolTrans [14] is a Molecular Interaction Transformer that is used for DTI prediction. The two key points of the model are the following.

- Previous machine learning models use the whole molecular structure of drugs and proteins in order to make predictions. However, it is known that the interactions between drugs and proteins are sub-structural [15]. MolTrans decomposes drugs and proteins into sub-structures to make more accurate predictions. Moreover, it creates a map of interactions to understand which sub-structures interact between each other.
- Although there are relatively few confirmed drug-target pairs, there is a lot of drug and target data available from different sources that is not labeled. MolTrans uses this data to create high-quality sub-structures with an algorithm called Frequent Consecutive Sub-sequence (FCS) mining. Moreover, it uses transformers to capture the complex signals among the sequential sub-structures of drugs and proteins.

The model is summarized in Figure 4.

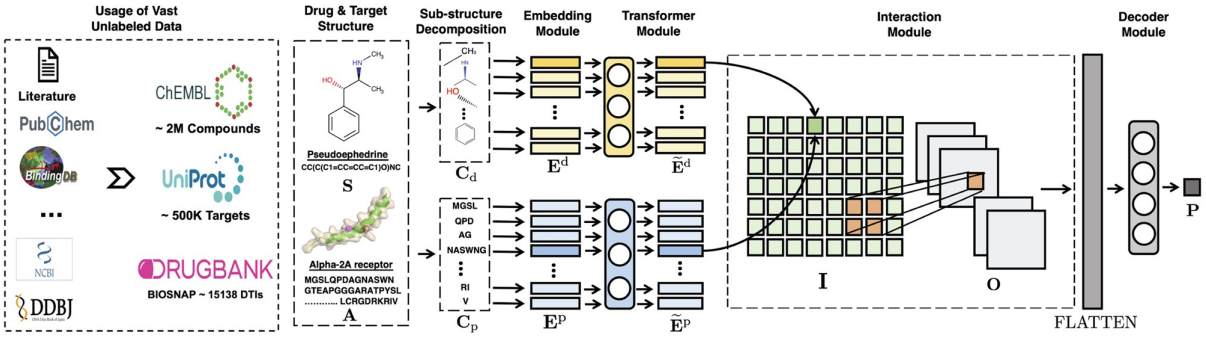

Figure 4: MolTrans model structure [14].

The input of the model is the following:

- Drugs are represented by the Simplified Molecular Input Line Entry Specification (SMILES) string.
- Target proteins are represented by a sequence composed of any of the 23 aminoacids.

The model first takes the input drug and target data, and passes it through the FCS mining module. This module is first trained on unlabeled data from drugs and proteins to identify the most common sub-structures. Then, the input drug and target data is decomposed into a sequence of sub-structures $\mathbf{C}_\mathrm{d}$ and $\mathbf{C}_\mathrm{p}$, respectively.

The next module in the model is the augmented transformer embedding module. The objective of this part is to get an embedding of the drug and the protein based on $\mathbf{C}_\mathrm{d}$ and $\mathbf{C}_\mathrm{p}$. This module is separated into two submodules: the embedding module and the transformer module.

The first step in the embedding module is to transform $\mathrm{C_d}$ and $\mathrm{C_p}$ into $\mathbf{M}^\mathrm{d} \in \mathbb{R}^{k \times \Theta_d}$ and $\mathbf{M}^\mathrm{p} \in \mathbb{R}^{k \times \Theta_p}$. l/k is the total number of possible drug/protein sub-structures obtained from the FCS algorithm. Θ_d/Θ_p is the maximum length that the sequence of sub-structures can have for drugs/proteins, respectively. Each column $\mathbf{M}^\mathrm{d}_\mathrm{i}$ and $\mathbf{M}^\mathrm{p}_\mathrm{j}$ is a vector that contains a 1 in the corresponding sub-structure index for the ith sub-structure of the drug sequence and the jth sub-structure of the protein sequence, respectively.

Two different embeddings are created and then aggregated, namely, the content embedding and the positional embedding. The content embedding is denoted as $\mathbf{E}^\mathrm{d}_{\mathrm{cont_i}}$ for a drug's ith sub-structure and $\mathbf{E}^\mathrm{p}_{\mathrm{cont_j}}$ for a protein's jth sub-structure. They are generated as

$$\mathbf{E}^\mathrm{d}_{\mathrm{cont_i}} = \mathbf{W}^\mathrm{d}_\mathrm{cont}\mathbf{M}^\mathrm{d}_\mathrm{i} \tag{1}$$

$$\mathbf{E}^\mathrm{p}_{\mathrm{cont_j}} = \mathbf{W}^\mathrm{p}_\mathrm{cont}\mathbf{M}^\mathrm{p}_\mathrm{j} \tag{2}$$

$\mathbf{W}^{\mathrm{d}}_{\mathrm{cont}} \in \mathbb{R}^{\vartheta \times l}$ and $\mathbf{W}^{\mathrm{p}}_{\mathrm{cont}} \in \mathbb{R}^{\vartheta \times k}$ are dictionary lookup matrices that are used to generate the content embeddings for drugs and proteins, respectively. These matrices are learned during the training process of the model. ϑ is the size of the embedding.

The positional embedding $\mathbf{E}^{\mathrm{d}}_{\mathrm{pos_i}}$, $\mathbf{E}^{\mathrm{p}}_{\mathrm{pos_j}}$ is obtained as

$$\mathbf{E}^{\mathrm{d}}_{\mathrm{pos_i}} = \mathbf{W}^{\mathrm{d}}_{\mathrm{pos}} \mathbb{I}^{\mathrm{d}}_{\mathrm{i}} \tag{3}$$

$$\mathbf{E}^{\mathrm{p}}_{\mathrm{pos_j}} = \mathbf{W}^{\mathrm{p}}_{\mathrm{pos}} \mathbb{I}^{\mathrm{p}}_{\mathrm{j}} \tag{4}$$

$\mathbf{W}^{\mathrm{d}}_{\mathrm{pos}} \in \mathbb{R}^{\vartheta \times \Theta_d}$ and $\mathbf{W}^{\mathrm{p}}_{\mathrm{pos}} \in \mathbb{R}^{\vartheta \times \Theta_p}$ are dictionary lookup matrices and they are learned during the training process. $\mathbb{I}^{\mathrm{d}}_{\mathrm{i}} \in \mathbb{R}^{\Theta_d}$ and $\mathbb{I}^{\mathrm{p}}_{\mathrm{i}} \in \mathbb{R}^{\Theta_p}$ contain a one in the ith and jth position respectively.

Once the content embedding and the positional embedding are obtained, the final embedding $\mathbf{E}^{\mathrm{d}}_{\mathrm{i}}$, $\mathbf{E}^{\mathrm{p}}_{\mathrm{j}}$ is calculated as

$$\mathbf{E}^{\mathrm{d}}_{\mathrm{i}} = \mathbf{E}^{\mathrm{d}}_{\mathrm{cont_i}} + \mathbf{E}^{\mathrm{d}}_{\mathrm{pos_i}} \tag{5}$$

$$\mathbf{E}^{\mathrm{p}}_{\mathrm{j}} = \mathbf{E}^{\mathrm{p}}_{\mathrm{cont_j}} + \mathbf{E}^{\mathrm{p}}_{\mathrm{pos_j}} \tag{6}$$

Up to this part, the obtained embeddings are independent for each sub-structure of the drug and the protein. The role of the transformer module is to capture the chemical relationships between the sub-structures to augment the embedding, by using transformer encoder layers:

$$\tilde{\mathbf{E}}^{\mathrm{d}} = \mathrm{Transformer}_{\mathrm{Drug}}(\mathbf{E}^{\mathrm{d}}) \tag{7}$$

$$\tilde{\mathbf{E}}^{\mathrm{p}} = \mathrm{Transformer}_{\mathrm{Protein}}(\mathbf{E}^{\mathrm{p}}) \tag{8}$$

Once the embedding is achieved, the interaction prediction module comes next. It is divided into the pairwise interaction layer and the neighborhood interaction layer.

The pairwise interaction is modeled for each sub-sequence i of the drug and j for the protein as follows.

$$\mathbf{I}_{\mathrm{i,j}} = \mathrm{F}(\tilde{\mathbf{E}}^{\mathrm{d}}_{\mathrm{i}}, \tilde{\mathbf{E}}^{\mathrm{p}}_{\mathrm{j}}) \tag{9}$$

F can be any function that measures the interaction between the pair of sub-structures. MolTrans uses the dot product as this function and therefore, the output is a two-dimensional interaction map $\mathbf{I} \in \mathbb{R}^{\Theta_d \times \Theta_p}$.

The neighborhood interaction layer is important because it models the interactions that the nearby sub-structures may have on each sub-substructure. A CNN layer is used to model these interactions. The output representation $\mathbf{O}$ is obtained as

$$\mathbf{O} = \mathrm{CNN}(\mathbf{I}) \tag{10}$$

The last part of the model is the decoder module. The objective of this modules is to output a probability of interaction between the input drug and protein. To do that, $\mathbf{O}$ is first flattened into a vector. This vector is fed into a linear layer to output the probability of interaction $\mathbf{P}$. The linear layer is parameterized by a weight matrix $\mathbf{W}_o$ and a bias vector $\mathbf{b}_o$ and $\mathbf{P}$ is calculated as

$$\mathbf{P} = \sigma(\mathbf{W}_o \text{FLATTEN}(\mathbf{O}) + \mathbf{b}_o) \tag{11}$$

σ is the sigmoid function defined as $\sigma(a) = \frac{1}{1+\exp(-a)}$.

All the model's parameters are optimized using the binary classification loss $\mathbf{L}$:

$$\mathbf{L} = \mathbf{Y}\log(\mathbf{P}) + (1 - \mathbf{Y})\log(1 - \mathbf{P}) \tag{12}$$

$\mathbf{Y}$ is the ground truth label.

3.2 HyperAttentionDTI

HyperAttentionDTI [16] proposes an end-to-end method to predict DTIs that is bio-inspired. The model is based on CNN and the attention mechanism. Unlike other methods, HyperAttentionDTI makes use of the attention mechanism to incorporates several non-covalent interaction types, such as hydrophobic interactions and hydrogen bonds.

The model is summarized in Figure 5.

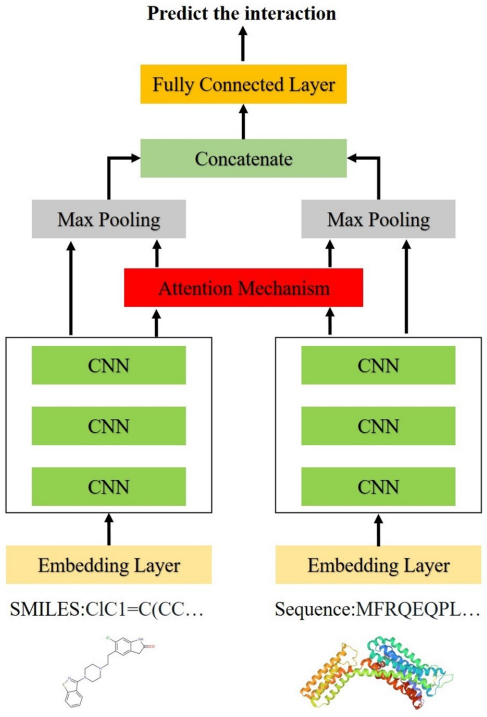

Figure 5: HyperAttentionDTI model structure [16].

The input of the model is the following:

- Drugs are represented by the SMILES string.
- Target proteins are represented by a sequence composed of any of the 23 aminoacids.

The first part of the model is the Embedding Layer. It transforms each character in the SMILES or aminoacid sequence string to an embedding vector. After this layer, the matrix $D_e \in \mathbb{R}^{M \times ed}$ for each drug $P_e \in \mathbb{R}^{N \times ep}$ for each protein are obtained. M/N is the length of the string of the drug/protein and ed/ep is the size of the embedding for the drug/protein.

Once the embedding is done, the next part of the model is the CNN block. There is a specific block for drugs and another one for proteins. Each CNN block has three one-dimensional CNN layers. The objective of these layers is to extract semantic information from the sequences. When the embedding matrices D_e and P_e are passed through their corresponding CNN blocks, the latent feature matrices $D_{cnn} \in \mathbb{R}^{M \times f}$ for the drug and $P_{cnn} \in \mathbb{R}^{N \times f}$ for the protein are generated. f corresponds to the number of filters that the last CNN layer has.

The Attention block comes after the CNN block. This module models the semantic interdependencies both in spatial and channel dimensions between the drug and protein subsequences. The input of this block is $D_{cnn} = \{d_1, d_2, ..., d_M\}$ and $P_{cnn} = \{p_1, p_2, ..., p_N\}$ and the output is an attention matrix $A \in \mathbb{R}^{N \times M \times f}$ that captures the aforementioned interactions in spatial and channel dimensions.

The steps to obtain A are the following. First, the attention vectors da_i and pa_j are obtained from d_i and p_j as

$$da_i = F(W_d \cdot d_i + b) \tag{13}$$

$$pa_j = F(W_p \cdot p_j + b) \tag{14}$$

F is a non-linear activation function such as ReLU. $W_d \in \mathbb{R}^{f \times f}$ and $W_p \in \mathbb{R}^{f \times f}$ are the weight matrices, and b is the bias vector. These last three parameters are learned during the training process.

The attention vector $A_{i,j} \in \mathbb{R}^f$ is generated as

$$A_{i,j} = F(W_a \cdot (da_i + pa_j) + b) \tag{15}$$

$W_a \in \mathbb{R}^{2f \times f}$ is the trainable weight matrix.

Therefore, the attention matrix A is directly obtained from the attention vectors. To get the attention matrix for drugs $A_d \in \mathbb{R}^{M \times f}$ and for proteins $A_p \in \mathbb{R}^{N \times f}$, the following operations are done.

$$A_d = \sigma(MEAN(A, 2)) \tag{16}$$

$$A_p = \sigma(MEAN(A, 1)) \tag{17}$$

$MEAN(a, b)$ returns the mean value of each row of the matrix a along the dimension b. σ is the sigmoid function.

The next step is to update the latent feature matrices to D_a and P_a as follows.

$$D_a = D_{cnn} \cdot 0.5 + D_{cnn} \odot A_d \tag{18}$$

$$P_a = P_{cnn} \cdot 0.5 + P_{cnn} \odot A_p \tag{19}$$

$\odot$ means element-wise multiplication.

Once D_a and P_a are obtained, a global-max pooling operation is applied to them in order to get the feature vectors v_{drug} and $v_{protein}$. These two vectors are then concatenated.

Lastly, the concatenated vector is passed through the output block. This block is composed of multilayer fully connected neural networks with Leaky ReLU as activation function and dropout layers between each neural network layer, in order to avoid overfitting. the output of the block is the probability of interaction $\hat{y}$ between the input drug and protein.

Given the ground truth label y, the model is trained based on the binary cross entropy loss defined as follows.

$$\mathrm{L} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})] \tag{20}$$

3.3 Evaluation methodology

Every model has been designed to be trained in a specific way. Therefore, to compare both methods equally, the original methodology in each model should be applied.

MolTrans uses a training, validation and test scheme. This means that the dataset is divided into 3 sets. For instance, 70% of the dataset becomes the training set, 10% becomes the validation set and 20% the test set. The validation set is useful to optimize the hyperparameters of the model.

HyperAttentionDTI is evaluated on a 5-fold cross validation scheme. In other words, the dataset is divided into 5 parts. Each time, one of the parts is taken as the test set and the rest as the training set. The results are then averaged. Moreover, it takes 20% of the training set as the validation set.

To compare both methods, the models are evaluated on 3 different splits:

- Sp split: Every drug and target in the dataset is part of at least one interaction both in the training and test set.
- Sd split: Some drugs only appear in interactions on the test set, so that the generalization of the model with respect to drugs can be evaluated.
- St split: Some targets only appear in interactions on the test set, so that the generalization of the model with respect to targets can be evaluated

The dataset is divided using 5 different random seeds and the mean of the results is performed for each split.

Chapter 4

Results and discussion

4.1 Datasets

The datasets that have been used to compare the methods are the following:

- BindingDB [17]: It consists of 10665 drugs and 1413 proteins and it contains DTIs with Kd (dissociation constant) affinity values obtained from wet lab assays. For this project, Kd values below 30 units are considered positive interactions.
- BIOSNAP [18]: DTI network that contains drugs that are available in the U.S. market.
- Davis [19]:It consists of 68 drugs and 379 proteins and contains DTIs with Kd affinity values obtained from wet lab assays. For this project, Kd values below 30 units are considered positive interactions.
- DrugBank [20]: It contains drugs that are approved or soon to be approved by the U.S. Food and Drug Administration (FDA). It has information on drugs, DTIs, drug action and drug interaction.
- Yamanishi [21]: There are 4 different datasets depending on the target proteins classes. For instance, enzymes (Yamanishi E), G protein-coupled receptors (GPCR), ion-channels (IC) and nuclear receptors (NR).

The summary of the relevant features of each dataset after the preprocessing can be found in Table 1.

Dataset	Drugs	Proteins	Positive DTI pairs
BindingDB	3085	719	5942
BIOSNAP	4643	2229	14486
Davis	65	314	1048
DrugBank	7434	4975	26419
Yamanishi E	445	663	2924
Yamanishi GPCR	223	95	635
Yamanishi IC	210	204	1476
Yamanishi NR	54	26	90

Table 1: Summary of dataset features.

Since the datasets only contain positive DTI pairs, subsampling is performed to balance them with negative DTI pairs. The subsampling is performed by counting the number of positive pairs that each drug and protein has, and generating the same number of negative pairs, randomly.

4.2 Metrics

The metrics that have been used to compare both methods are the Area Under the Receiver Operating Characteristic (ROC) curve (AUC-ROC) and the Area Under the Precision Recall (PR) curve (AUC-PR).

To explain what each of the metrics measures, the confusion matrix needs to be defined first. The task that is being solved is a binary classifier where a drug-target pair can interact or not. Therefore, when a drug-target pair is passed as input to the models four outcomes can happen, based on the ground-truth:

- A true positive is correctly predicted as positive.
- A true positive is incorrectly predicted as negative. Therefore, it is a false negative.
- A true negative is incorrectly predicted as positive. Therefore, it is a false positive.
- A true negative is correctly predicted as negative.

Therefore, the confusion matrix is defined as in Table 2.

	Truth = 0	**Truth = 1**
Prediction = 0	True negatives (TN)	False negatives (FN)
Prediction = 1	False positives (FP)	True positives (TP)

Table 2: Confusion matrix.

Given a test set, the objective is to complete the confusion matrix with the corresponding values. Then, the Precision (P) True Positive Rate (TPR) or Recall and the False Positive Rate (FPR) can be calculated as in Equations 22 and 23.

$$P = \frac{TP}{TP + FP} \tag{21}$$

$$TPR = \frac{TP}{TP + FN} \tag{22}$$

$$FPR = \frac{FP}{FP + TN} \tag{23}$$

The ROC curve is the plot of the TPR versus the FPR for different thresholds in the classifier. The PR curve is the plot of the precision versus the recall for different thresholds in the classifier. An example of both plots can be found in Figure 6.

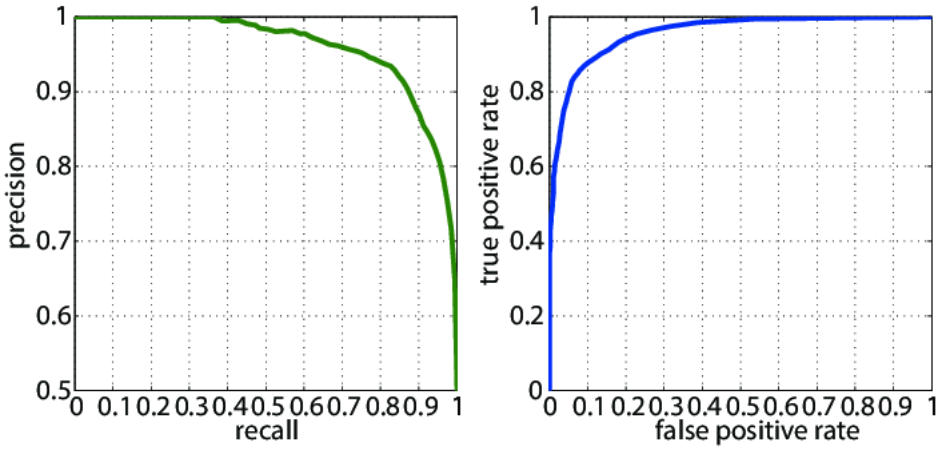

Figure 6: PR curve (left) and ROC curve (right) [22].

The AUC-ROC and AUC-PR are simply the area under their corresponding curve, which is a number between 0 and 1. The closer to 1 the metrics are, the better the model has performed.

4.3 Results

After running MolTrans and HyperAttentionDTI on the 8 datasets and with 3 different splits, the results obtained can be seen in Tables 3, 4, 5.

Datasets	**ROC-AUC**		**PR-AUC**	
	MolTrans	**HyperAttentionDTI**	**MolTrans**	**HyperAttentionDTI**
BindingDB	0.7834	0.9165	0.6525	0.8593
BIOSNAP	0.6947	0.7905	0.5230	0.6814
Davis	0.5255	0.5542	0.3330	0.3953
DrugBank	0.7152	0.7914	0.5533	0.6845
Yamanishi E	0.5753	0.7003	0.4107	0.6113
Yamanishi GPCR	0.6623	0.7456	0.4735	0.5912
Yamanishi IC	0.6270	0.7264	0.4373	0.6374
Yamanishi NR	0.5442	0.4981	0.4211	0.3432

Table 3: Sd split results.

Datasets	**ROC-AUC**		**PR-AUC**	
	MolTrans	**HyperAttentionDTI**	**MolTrans**	**HyperAttentionDTI**
BindingDB	0.8962	0.9540	0.8026	0.9102
BIOSNAP	0.7896	0.8609	0.6298	0.7702
Davis	0.6928	0.7383	0.5294	0.5758
DrugBank	0.7917	0.8621	0.6444	0.7759
Yamanishi E	0.8834	0.9530	0.8003	0.9221
Yamanishi GPCR	0.7368	0.8042	0.5301	0.6479
Yamanishi IC	0.8752	0.9492	0.7719	0.9172
Yamanishi NR	0.5436	0.4568	0.4206	0.3264

Table 4: Sp split results.

Datasets	**ROC-AUC**		**PR-AUC**	
	MolTrans	**HyperAttentionDTI**	**MolTrans**	**HyperAttentionDTI**
BindingDB	0.7260	0.7449	0.5613	0.6174
BIOSNAP	0.7843	0.8592	0.6179	0.7737
Davis	0.6766	0.7129	0.4769	0.5478
DrugBank	0.7948	0.8632	0.6481	0.7769
Yamanishi E	0.7103	0.8449	0.6044	0.7954
Yamanishi GPCR	0.6615	0.7280	0.4883	0.5653
Yamanishi IC	0.7165	0.8688	0.5509	0.8267
Yamanishi NR	0.4064	0.3424	0.2664	0.2938

Table 5: St split results.

If the results are analyzed, it can clearly be observed that HyperAttentionDTI outperforms MolTrans on most cases. Therefore, it can be said that HyperAttentionDTI makes better

predictions compared to MolTrans.

However, MolTrans performs better in the Yamanishi NR dataset. As it was seen in Table 1, Yamanishi NR has relatively fewer data compared to the rest of the datasets. It can be concluded that HyperAttentionDTI does not predict DTIs as well as MolTrans when the dataset is small. MolTrans advantage may come from the fact that it uses a lot of unlabeled data to divide drugs and proteins into sub-structures.

Furthermore, both models make better predictions when the datasets are split in the Sp mode. This makes sense because, in Sp, the models are able to see all the drugs and proteins during the training phase.

Interestingly, MolTrans generalizes targets better because it obtains better results in St splits than Sd. On the other hand, HyperAttentionDTI makes a better generalization of drugs, getting better results in Sd splits than St. In other words, MolTrans is more capable of predicting DTIs with new targets and HyperAttentionDTI with new drugs.

Lastly, another important conclusion that can be obtained is that the size of the datasets are not directly proportional to the performances of the models. DrugBank contains the most number of interactions but there are smaller datasets such as, Yamanishi E where the models perform better in testing.

Chapter 5

Graph Machine Learning

The Machine Learning models that have been described up to now work well on vector or matrix data. However, they cannot be directly applied to graphs, since they are more complex data structures. Graphs have arbitrary size and their topological structure is complex. Moreover, they can change with time. Being able to make predictions with machine learning on these data structures would outperform conventional models due to their capability to model complex systems more accurately.

In this chapter, a brief introduction to graphs is going to be made, as well as a description of some fundamental Graph Machine Learning methods.

5.1 Graphs

A graph is a data structure that is useful to describe complex systems [23]. For example, the interactions between proteins could be represented as a graph by having proteins as nodes and their interactions as edges, as in Figure 7.

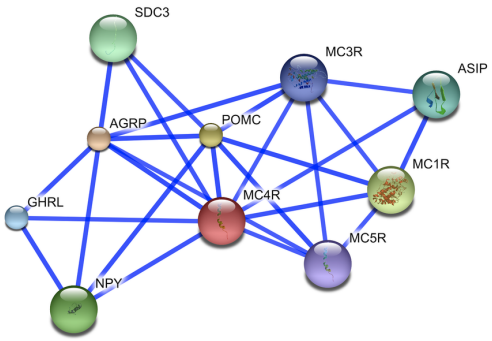

Figure 7: Human melanocortin 4 receptor (MC4R) protein-protein interaction graph [24].

Formally, a graph is formally defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of nodes and $\mathcal{E}$ is a set

of edges between the nodes. An edge going from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$ is denoted as $(u, v) \in \mathcal{E}$.

A graph can be represented through an adjacency matrix $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. Every node is indexed in a particular row and column of the matrix. An edge is represented as an entry in this matrix as follows:

$$A[u, v] = \begin{cases} 1, & \text{if } (u, v) \in \mathcal{E} \\ 0, & \text{otherwise} \end{cases}$$

Graphs can be classified as directed and undirected graphs. Moreover, graphs can also have weighted edges. This means that instead of the adjacency matrix having binary values, it can contain any real values, modeling the importance of each edge compared to the rest.

Multi-relational graphs are a type of graphs that can have different edge types. In these cases, the graph can be represented as an adjacency matrix specific to each type of edge. If a multi-relational graph also has nodes of different types, it is called an heterogeneous or knowledge graph.

5.2 Graph statistics

Graphs can be analyzed and compared using various statistics. In this section, the Davis and BindingDB datasets are going to be analyzed from a graph point of view. For this, the NetworkX [25] Python library has been used. This library contains data structures to represent graphs, as well as different algorithms to analyze and visualize them.

In Table 6, a summary of some graph statistics can be seen for Davis and BindingDB datasets. The density d of an undirected graph is defined as

$$d = \frac{2m}{n(n-1)} \times 100$$

where n is the total number of nodes and m is the total number of edges in the graph. The sparsity is the opposite of the density, representing the percentage of edges that do not exist over the total number of possible edges. A connected component within a graph is a sub-graph where every pair of nodes is connected through a path.

	Davis	BindingDB
Number of drugs	65	3085
Number of proteins	314	719
Total number of nodes	379	3804
Total number of edges	1048	5938
Number of connected components	1	232
Density	1.46%	0.08%
Sparsity	98.54%	99.92%

Table 6: Graph statistics for Davis and BindingDB datasets.

In Figures 8 and 9, a visualization of the Davis and BindingDB graphs can be found. It can be seen that in the BindingDB dataset there are a lot of drug-protein pairs that interact just between them, whereas in Davis, drugs and targets are more connected.

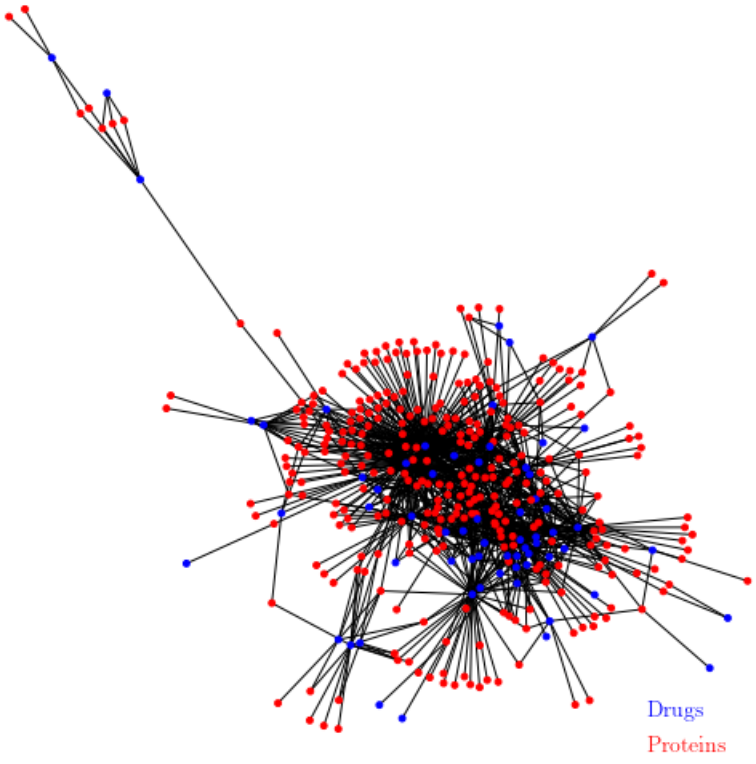

Figure 8: Davis graph.

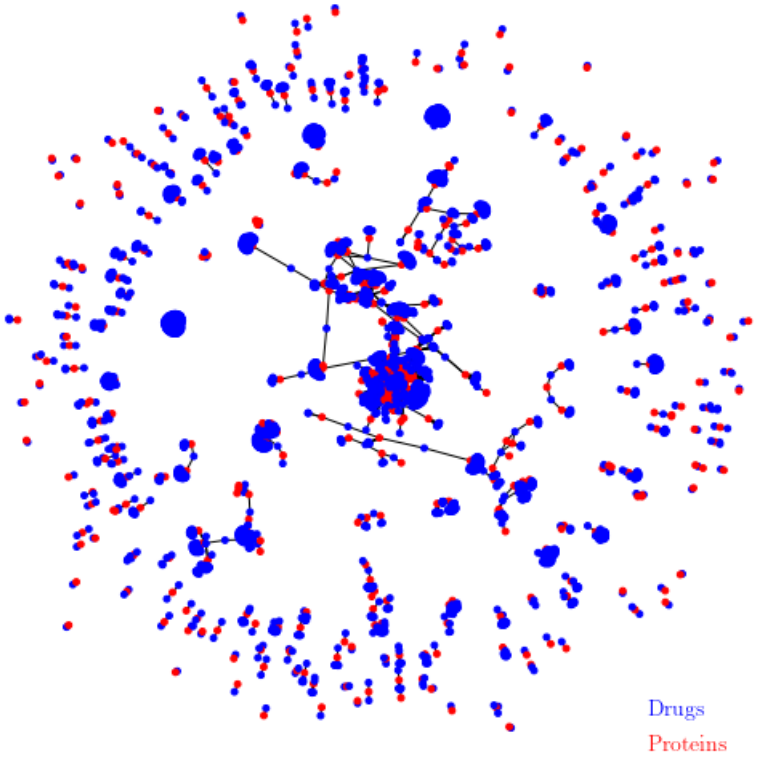

Figure 9: BindingDB graph.

5.3 Node embedding

The main objective of node embedding techniques is to use low-dimensional vectors to encode nodes. These vectors should capture the position of the nodes within the graph as well as their local neighborhood structure. By creating vector embeddings of nodes, conventional machine learning models could be applied to them in order to make predictions.

To create node embeddings an encoder-decoder framework is used. The steps to follow are summarized below.

1. The encoder maps node v to an embedding $\mathbf{z}_v \in \mathbb{R}^d$.

$$\text{ENC}(v) = \mathbf{z}_v$$

2. A node similarity function $\mathbf{S}[u, v]$ is defined to be able to measure the similarity between any pair of nodes u, v in the graph.

3. The decoder should map from the embeddings created by the encoder to the results obtained from the similarity function by using a certain function such as the dot product.

$$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v$$

4. The parameters of the encoder are optimized so that

$$\mathbf{S}[u, v] \approx \text{DEC}(\mathbf{z}_u, \mathbf{z}_v)$$

The most straightforward encoder is an embedding-lookup matrix.

$$\text{ENC}(v) = \mathbf{Z} \cdot v = \mathbf{z}_v$$

$\mathbf{Z} \in \mathbb{R}^{d \times |V|}$ is a matrix where each column represents a node embedding and $v \in \mathbb{I}^{|V|}$ is a one-hot vector representing the position of node v. $|V|$ is the total number of nodes in the graph.

The optimization of the encoder-decoder is done like a typical machine learning model, by defining a loss function $\mathcal{L}$ over a set of node pairs $\mathcal{D}$ used for training.

$$\mathcal{L} = \sum_{(u,v) \in \mathcal{D}} \ell(\text{DEC}(\mathbf{z}_u \mathbf{z}_v), \mathbf{S}[u, v])$$

where ℓ is a loss function that measures the difference between the two input values.

The machine learning algorithms that use this scheme to produce embeddings are called shallow embedding methods. There are many different node embedding methods that use approaches such as factorization [26, 27]. However, more recent random walk based node embedding methods will be explained.

Given a graph and a starting node, a random walk is a sequence of nodes that are chosen at random based on the structure of the graph. In random walk based methods, node embeddings are optimized based on the fact that two nodes will have similar node embeddings if the probability that they co-occur on a random walk is high.

Two important random walk based methods are DeepWalk [28] and node2vec [29].

The decoder in both cases is the following.

$$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) = \frac{\mathrm{e}^{\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_v}}{\sum_{v_k \in \mathcal{V}} \mathrm{e}^{\mathbf{z}_u^{\mathrm{T}} \mathbf{z}_{v_k}}}$$

The loss function is defined as

$$\mathcal{L} = \sum_{(u,v) \in \mathcal{D}} -\log(\text{DEC}(\mathbf{z}_u, \mathbf{z}_v))$$

$\mathcal{D}$ is the random walk training set.

The node similarity function is defined as probability of visiting a node v on a random walk of length T starting at node u. That is, $p_{\mathcal{G},T}(v|u)$. Therefore, the objective is to learn node embeddings so that

$$\text{DEC}(\mathbf{z}_u, \mathbf{z}_v) \approx p_{\mathcal{G},T}(v|u)$$

The main difference between the two methods is that DeepWalk defines $p_{\mathcal{G},T}(v|u)$ with uniform random walks. Meanwhile, node2vec makes use of hyperparameters in order to perform biased random walks and interpolate between breadth-first search (local graph description) and depth-first search (global graph description).

5.4 Limitations of shallow node embedding methods

Shallow node embedding methods have allowed many advances in the graph machine learning field. However, they have several drawbacks. The main problem with shallow node embeddings is that they are transductive. These methods can only generate embeddings for nodes that are seen in the training phase. If the embeddings of new nodes were needed, these methods would need to retrain the whole model again. Moreover, the described node embedding techniques cannot make use of feature data that each node may have.

Given the possibility that graphs can be dynamic over time, inductive methods that are able to generalize to previously unseen nodes are preferred. The most popular example of inductive models are graph neural networks.

5.5 Graph Neural Networks

The main objective of graph neural networks (GNN) is to generate embeddings of nodes based on the structure of the graph and feature information that each node may have. In other words, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of node features $\mathbf{X} \in \mathbb{R}^{d \times |\mathcal{V}|}$, GNNs generate node embeddings $\mathbf{z}_u, \forall u \in \mathcal{V}$.

GNNs are based on the neural message passing framework. That is, messages in the form of vectors are passed between nodes and they are transformed and aggregated in order to produce embeddings.

The key idea is that each node has its own computation graph based on their neighborhood structure. In other words, each node defines its own neural network. Messages from each node's neighbors are aggregated, transformed and passed to the node in order to generate the corresponding embedding vector.

In Figure 10, an example of a 2-layer GNN can be seen. If a GNN has K layers, it means

that it is capturing messages in the form of vectors from nodes that are up to K hops away from the objective node. A layer-0 GNN simply takes the feature vector of the node and transforms it to produce the embedding.

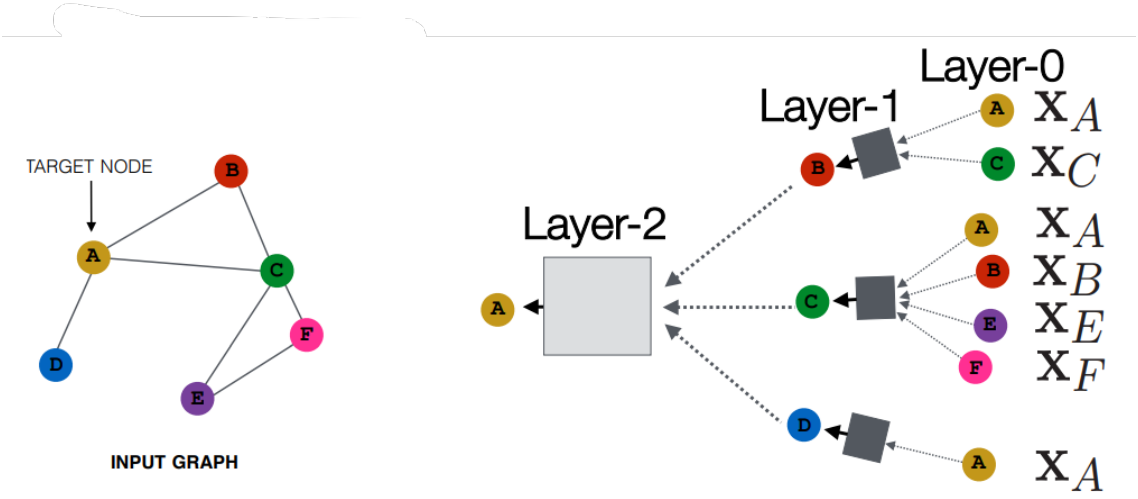

Figure 10: Computation graph of node A [30].

In a GNN layer two steps happen. First, the message from each node is transformed and then, all incoming messages are aggregated, as in Figure 11.

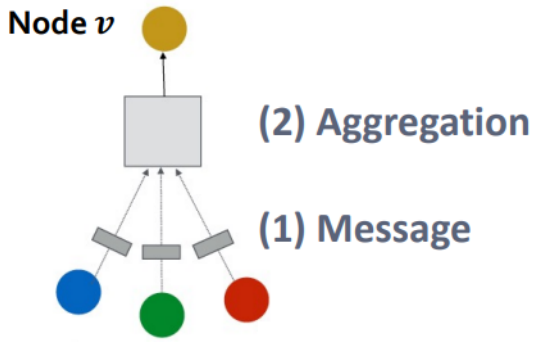

Figure 11: Steps in a GNN layer [30].

If $\mathbf{m}_u^{(l)}$ is the message from node u at layer l, it is computed as

$$\mathbf{m}_u^{(l)} = \text{MSG}^{(l)}(h_u^{(l-1)})$$

where $h_u^{(l-1)}$ is the embedding of node u at layer $l-1$ and $\text{MSG}^{(l)}$ is the message function at layer l. The message function can be of many types, such as a linear layer or a feedforward neural network.

Once the messages from each neighboring node are obtained, the embedding of node v at

layer l, $\mathbf{h}_v^{(l)}$, is obtained after the aggregation step as

$$\mathbf{h}_v^{(l)} = \text{CONCAT}(\text{AGG}(\{\mathbf{m}_u^{(l)}, u \in N(v)\}), \mathbf{m}_v^{(l)})$$

The aggregation operation AGG needs to be order invariant since nodes in a graph have arbitrary ordering. For instance, the sum or the mean of the messages. $N(v)$ is the set of v's neighboring nodes. The CONCAT operation is not mandatory but it guarantees that the information from node v is not lost.

Non-linear functions such as σ or ReLU could be applied at the message or aggregation step so that expressiveness is added.

Three fundamental GNN layers are:

- Graph Convolutional Networks (GCN) [31]: In Equation 24, the node embedding result of node v at layer l can be found. The message consists on transforming the embedding of each neighbor node at the previous layer by a weight matrix $\mathbf{W}^{(l)}$ and normalizing it with respect to the total number of neighbors. The aggregation part is simply a summation over all the messages. Lastly, the non-linear σ function is applied.

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \mathbf{W}^{(l)} \frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}\right) \tag{24}$$

- GraphSAGE [32]: In Equation 25, it can be seen that the message passing and aggregation happen within the AGG. Then, the information of the node itself is aggregated through a concatenation operation $CONCAT$.

$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_{\text{v}}^{(\text{l}-1)}, \text{AGG}\left(\left\{\mathbf{h}_{\text{v}}^{(\text{l}-1)}, \forall \text{u} \in \text{N(v)}\right\}\right)\right)\right) \tag{25}$$

- Graph Attention Networks (GAT) [33]: The particularity of GATs is the attention weights α_{vu}, as it can be seen in Equation (26). These weights model the importance that node u's message has on node v. They are computed for every connected node pair and they are trained simultaneously as the parameter weight matrix $\mathbf{W}^{(l)}$.

$$\mathbf{h}_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{vu} \mathbf{W}^{(l)} \mathbf{h}_u^{(l-1)}\right) \tag{26}$$

One key advantage of GNNs is their efficiency, since the parameter matrices $W^{(l)}$ are shared across all nodes and they are unique for each layer. Therefore, the number of parameters is significantly decreased compared to shallow embedding methods where the encoder optimizes the embedding for each node. Moreover, they can be trained on a graph and be applied to a previously unseen graph, thanks to its inductive capability.

The challenge lies in choosing the most accurate GNN architectures since there are many choices to be made such as the depth of the model, the aggregation function, the message transformation function, etc.

Chapter 6

Conclusions

To summarize, in this project MolTrans and HyperAttentionDTI have been evaluated on 8 different datasets and 3 different splits to test their generalization capabilities with respect to drugs and proteins. It can be concluded that HyperAttention performs better than MolTrans

Even though, both models offer competitive performance, after analyzing fundamental concepts of Graph Machine Learning, I believe that transforming the datasets into graphs and applying machine learning methods to them could improve substantially on current state-of-the-art methods.

Graph Machine Learning is still an emerging field of research, so the true potential that these methods could have is still not entirely known. From this project, the most important outcome that can be taken is that shallow node embeddings methods are not going to be useful for DTI prediction, due to their inherent transductivity. However, Graph Neural Networks have the necessary characteristics to revolutionize the drug repurposing field.

Apart from this, the two analyzed models only use sequence of strings to represent drugs and proteins. Graphs allow to make use of more features for their nodes. Therefore, apart from developing new graph-based machine learning models, new data from drugs and proteins should be considered and incorporated when performing DTI prediction. For instance, the recently developed model AlphaFold [34] has allowed to obtain the three-dimensional structure of most proteins very accurately. Moreover, different types of edges could be used for the graph representation, such as protein-protein interactions or drug side effects. From my perspective, this kind of data could be exploited to develop a more robust model.

All in all, it is clear that from this project, future work includes research on possible new drug and protein data and the development of an innovative machine learning model based on GNNs that outperforms MolTrans and HyperAttentionDTI.

Budget

In this section, the total budget needed for the development of the project is going to be explained. It is divided into:

- Equipment: it includes all the cost related to already acquired machinery. The calculation of the cost is done taking in consideration the amortization time and usage needed for the project.
- Consumable material: it includes all the material that is consumed during the development process of the project.
- Human resources: it includes the necessary human costs in each task of the project, based on the hours spent.

In Tables 7, 8, 9, the costs of equipment, consumable materials and human resources can be found, respectively.

The total cost of the project is 16224 €, as it can be seen in Table 10.

Equipment	**Adquisition price (€)**	**Amortization time (years)**	**Monthly amortization cost (€)**	**Usage time (months)**	**Amortization (€)**
Dell Precision 7280	16,000	4	333.33	6	1999,98
Quadro RTX 4000 GPU	1152,88	4	288,22	2	576,44
		Total			2576,42

Table 7: Equipment budget.

Description	**Quantity (kWh)**	**Price (€)**	
		Unitary	**Total**
GPU power consumption	53.76	0.246	13.22
Total			13.22

Table 8: Consumable material budget.

Task	**Duration (hours)**	**Price (€)**	
		Unitary	**Total**
1. Machine Learning preliminaries	30	30	900
2. Understanding and analyzing MolTrans	10	30	300
3. Understanding and analyzing HyperAttentionDTI	10	30	300
4. Dataset collection	25	30	750
5. Dataset preprocessing	100	30	3000
6. Graph Machine Learning analysis	40	30	1200
7. Programming scripts for simulations	20	30	600
8. Redaction of project documentation	85	30	2550
Total	320		9600

Table 9: Human resources budget.

Type	Price (€)
Equipment	2576.42
Consumable material	13.22
Human resources	9600
Indirect costs (10%)	1218.97
Total without taxes	13408.61
Total with taxes (21%)	16225

Table 10: Total budget.

Bibliography

[1] James P Hughes et al. "Principles of early drug discovery". In: *British journal of pharmacology* 162.6 (2011), pp. 1239–1249.

[2] Duxin Sun et al. "Why 90% of clinical drug development fails and how to improve it?" In: *Acta Pharmaceutica Sinica B* (2022).

[3] Ali Masoudi-Nejad, Zaynab Mousavian, and Joseph H Bozorgmehr. "Drug-target and disease networks: polypharmacology in the post-genomic era". In: *In silico pharmacology* 1.1 (2013), pp. 1–4.

[4] Vineela Parvathaneni et al. "Drug repurposing: a promising tool to accelerate the drug discovery process". In: *Drug discovery today* 24.10 (2019), pp. 2076–2085.

[5] Ruolan Chen et al. "Machine learning for drug-target interaction prediction". In: *Molecules* 23.9 (2018), p. 2208.

[6] Aurélien FA Moumbock et al. "Current computational methods for predicting protein interactions of natural products". In: *Computational and Structural Biotechnology Journal* 17 (2019), pp. 1367–1376.

[7] Dr Singh et al. "QSAR and its role in target-ligand interaction". In: *The Open Bioinformatics Journal* 7.1 (2013).

[8] Gerard Pujadas et al. "Protein-ligand docking: A review of recent advances and future perspectives". In: *Current Pharmaceutical Analysis* 4.1 (2008), pp. 1–19.

[9] Zaynab Mousavian and Ali Masoudi-Nejad. "Drug–target interaction prediction via chemogenomic space: learning-based methods". In: *Expert opinion on drug metabolism & toxicology* 10.9 (2014), pp. 1273–1287.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] Caio BS Maior et al. "Convolutional neural network model based on radiological images to support COVID-19 diagnosis: Evaluating database biases". In: *Plos one* 16.3 (2021), e0247839.

[12] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[14] Kexin Huang et al. "MolTrans: Molecular Interaction Transformer for drug–target interaction prediction". In: *Bioinformatics* 37.6 (2021), pp. 830–836.

[15] Jia Jia et al. "Mechanisms of drug combinations: interaction and network perspectives". In: *Nature reviews Drug discovery* 8.2 (2009), pp. 111–128.

[16] Qichang Zhao et al. "HyperAttentionDTI: improving drug–protein interaction prediction by sequence-based deep learning with attention mechanism". In: *Bioinformatics* 38.3 (2022), pp. 655–662.

[17] Tiqing Liu et al. "BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities". In: *Nucleic acids research* 35.suppl_1 (2007), pp. D198–D201.

[18] Marinka Zitnik, Rok Sosic, and Jure Leskovec. "BioSNAP Datasets: Stanford biomedical network dataset collection". In: *Note: http://snap. stanford. edu/biodata Cited by* 5.1 (2018).

[19] Mindy I Davis et al. "Comprehensive analysis of kinase inhibitor selectivity". In: *Nature biotechnology* 29.11 (2011), pp. 1046–1051.

[20] David S Wishart et al. "DrugBank 5.0: a major update to the DrugBank database for 2018". In: *Nucleic acids research* 46.D1 (2018), pp. D1074–D1082.

[21] Yoshihiro Yamanishi et al. "Prediction of drug–target interaction networks from the integration of chemical and genomic spaces". In: *Bioinformatics* 24.13 (2008), pp. i232–i240.

[22] Giorgio Roffo et al. "Reading between the turns: Statistical modeling for identity recognition and verification in chats". In: *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*. IEEE. 2013, pp. 99–104.

[23] William L Hamilton. "Graph representation learning". In: *Synthesis Lectures on Artifical Intelligence and Machine Learning* 14.3 (2020), pp. 1–159.

[24] Yana Bromberg. "Chapter 15: Disease Gene Prioritization". In: *PLoS computational biology* 9 (Apr. 2013), e1002902. DOI: 10.1371/journal.pcbi.1002902.

[25] Aric Hagberg, Pieter Swart, and Daniel S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[26] Amr Ahmed et al. "Distributed large-scale natural graph factorization". In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 37–48.

[27] Shaosheng Cao, Wei Lu, and Qiongkai Xu. "Grarep: Learning graph representations with global structural information". In: *Proceedings of the 24th ACM international on conference on information and knowledge management*. 2015, pp. 891–900.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

[29] Aditya Grover and Jure Leskovec. "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[30] Jure Leskovec. *Lecture Notes from CS224W Machine Learning with Graphs*. Stanford University, 2021.

[31] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[32] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[33] Petar Veličković et al. "Graph attention networks". In: *arXiv preprint arXiv:1710.10903* (2017).

[34] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.