Department of Computer Science
Selma Lagerløfs Vej 300
DK-9220 Aalborg Ø

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Representation learning with autoencoders in drug-target interaction prediction

Theme:
Drug Target Interaction (DTI) prediction, Deep Learning, Autoencoders

Project Period:
February 2021 - June 2021

Project Group:
dt101f21

Participant(s):
Alexandr Dyachenko
Andreas Laugård Hald
Christian Galasz Nielsen
Dominik Tabak

Supervisor(s):
Christian S. Jensen
Jilin Hu

Copies: 1

Number of Pages: 42

Date of Completion:
June 17, 2021

Abstract:

Interactions between molecules and proteins is an area that currently has a lot of interest. Developing methods that can predict these interactions would provide large cost savings as well as improve the speed at which new interactions can be found. One way that researchers have tried finding these interactions is by using autoencoders. We employ an architecture involving two autoencoder neural networks that feed into a prediction network. We compare to state-of-the-art methods across several metrics and find that the autoencoders we present score high in AUPR, but low in other metrics compared to the other methods.

A thank you note from the authors

We would like to take this opportunity to thank our supervisors Christian S. Jensen and Jilin Hu of Aalborg University for their endless patience, continuous knowledge sharing, committed guidance, and constant motivation throughout this project.

We would also like to thank our families and loved ones for their limitless support in pushing us towards this achievement. We hereby dedicate this thesis to them.

Summary

Recently there has been a big focus on being able to predict interactions between proteins and molecules. Finding these interactions the traditional way is both very costly, but also very expensive. Therefore, new methods are constantly being developed that attempt to help find these interactions. There are multiple different "groups" of algorithms that attempt to accomplish this. Due to data scarcity there is a need for deep learning frameworks that can learn representations with less labelled data. Therefore, we instead utilized autoencoders in our framework.

We try 3 different setups for encoding and decoding which are: CNN for encoding and CNN for decoding. CNN for encoding, and with DNN for decoding. And lastly, RNN for encoding, and DNN for decoding.

We then compared our 3 approaches with both older methods to find interactions, but also with state-of-the-art methods and found out that our 3 approaches underperformed on all metrics for both datasets that we tested them on, except for a better AUPR on the davis dataset. In addition to this, we also created our own base model which instead relied on featurization of the molecules and proteins into vectors before applying a basic neural network and found out that this basic model also performed better than our 3 approaches using autoencoders.

A strong point of autoencoders, however, is that they do not need to have their data labelled, so we created a dataset with reduced labels and compared our best variant (AUEN CNN-CNN) with DeepDTA which is the best performing model, but in this setting DeepDTA still performed better, which was not surprising because it also relies on autoencoders.

Lastly, we compared our best variant (CNN-CNN) with our own base model, with our own base model only relying on one-hot encoding and once again we used the data that had reduced labels. This time, however, our best variant performed better than the base model, especially in terms of MSE. From this, we concluded that our approach could potentially be used for binary classification as our AUPR metric was good, but that in general, the state-of-the-art methods achieve better scores in our metrics.

Contents

Chapter 1

Introduction

Drug discovery is a time consuming and expensive process of finding new medication by probing a drug-target pair for interaction, where the target is a protein, and the drug is a molecule that either inhibits or stimulates the protein's function [1]. Finding new medication is important to help humankind overcome diseases and illnesses.

Discovering drugs was originally done manually by finding which drugs had interactions with which targets, but machine learning has provided a more precise and focused approach, accelerating the speed of finding these interactions. Because the interactions are vital to drug discovery, increasing the efficiency and accuracy of finding them holds the potential to allow for faster drug design. For example each year, only about 50 new molecular entities are approved by the U.S. Food and Drug Administration [2], and 39 new active substances by the European Medicines Agency [3]. Molecular entities and active substances are similar terms and describe active ingredients, which are responsible for the activity of medicine.

Computer scientists have tried to reduce the weak points of the drug discovery process by using machine learning approaches for Drug-Target Interaction (DTI) prediction [4]. As drug discovery has advanced, new machine learning techniques have been explored and developed to predict interactions that matter for the production of new drugs. Popular methods include similarity or distance-based methods, deep learning methods and hybrid methods [5]. Each category has associated benefits and drawbacks, but as more and more data is becoming available, deep learning methods have risen in popularity, as one of the main issues with deep learning, data availability, though still a challenge, has become less impactful [5]. Moreover, deep learning approaches have proven efficient, while offering high accuracy in predicting interactions and a low error rate [6], resulting in fewer interactions being false positives and false negatives. These approaches also manage to reduce the loss of information when predicting DTIs, overcoming limitations of other approaches [5].

However, most approaches to predicting DTIs rely on an end-to-end learning style. Contrary to this our approach involves pre-training two autoencoders. Autoencoders are neural networks that compress the original input using an encoder and then reconstruct it through a decoder. The main idea is that the decoder output aims to be as similar to the input as possible, and as such, it is the encoder that is interesting in the drug discovery setting, because it learns a compressed encoding of the input while retaining the necessary information. This is called unsupervised learning meaning that no labelled data is required to train these networks. This, in turn, means that the frameworks that are comprised of multiple neural networks, some of which are autoencoders require less labelled data for training. And since labelled data is a scarce resource in DTI prediction, we find it fitting to use autoencoders to combat it.

It is the goal of the project to develop an advanced algorithm that can produce desirable results for DTI prediction. To achieve this, we propose a framework for drug-target interaction prediction using autoencoders to provide feature extraction and dimensionality reduction for both drugs and targets. This would allow the interaction prediction network to learn predicting interactions with less labelled data than usual.

This report is structured into six sections; this introduction (chapter 1), related work (chapter 2), preliminaries (chapter 3), framework (chapter 4), experiments (chapter 5), and conclusion (chapter 6). Related work goes through papers that have been relevant for this project, covering papers especially about DTI prediction involving binding affinity and autoencoders. Preliminaries contain introductory information and go into detail about the tools and methods used in our approach. The framework presents the specifics behind the design of our model. Experiments present our setup and metrics that enable comparison with this project to existing approaches. Finally, the conclusion offers some finalizing words on the project, including potential future work.

This project builds on a previous project carried out by the same team. The goal of the previous project was to build a framework for drug discovery, including a database, interface, and algorithm. The algorithm was a distance-based method tested on the framework to produce results for discussion and improvement. Inspired by the future work leftover from the previous project, this project dives deeper into drug discovery. The previous project and the accompanying software package are available here [7]

Chapter 2

Related Work

In this section, we detail several relevant papers that have influenced this project and led to the solution we propose in this project. These are various methods and approaches to drug discovery. With many approaches already existing, we look to existing papers to find out where we can improve something or where we can find a niche within this project to not just replicate existing results but add to the domain.

One approach to drug discovery is proposed by SimBoost [8]. In this, the goal was to provide interaction data that is non-binary and results that are not based on binarization thresholds. To do this, they use a Kronecker Regularized Least Squares method (KronRLS) [9] as a baseline for drug-target interaction. They take data from enzymes, ion channels, nuclear receptor and G Protein-Coupled Receptor datasets for featurization. They then create a matrix of the binding affinities of drugs and targets, which is used as a feature extraction step where they balance the binding affinities to prevent overfitting. They feed this matrix to a gradient boosting regression tree, which has been trained on their dataset. This tree generates a score based on their algorithm.

Another approach that is also non-binary is the DeepDTA [10] paper approach, which uses a deep learning-based convolutional neural network in two blocks. One block takes sequences, and the other block takes SMILES strings. Each sequence or string goes through three consecutive one-dimensional convolutional layers with filters that doubles and triples in the second and third layers respectively. Finally, it goes through a max-pool layer where a representation of a sequence or string is output and concatenated into a combined representation that is put through three fully connected layers where they employ a dropout regularization technique to avoid over-fitting.

A different non-binary approach is employed in the DeepGS paper [6], which tries to consider the local chemical context of amino acids and SMILES strings along with a method that considers the molecular structure. They consider both because the topological structure of molecules provides information as to how the atoms are connected, while the chemical context reveals the functionality of atoms. They obtain one drug embedding from the graph of the molecule as a fingerprint and the other is obtained directly from SMILES strings by using Smi2Vec which produces a 100-dimensional ma-

trix. Similarly, the protein embedding also produces a 100-dimensional matrix with Prot2Vec. They put this information through three different neural networks (Convolutional Neural Network, Graph Attention Network, and Bi-directional Gated Recurrent Unit) that outputs one protein and two drug representations. They concatenate this output and input it into a feed-forward neural network to make predictions based on binding affinities.

One issue with these non-binary approaches is that they introduce a need for interpretation of data which become a challenge of this approach. Especially the neural network part of the DeepDTA approach also shows poor performance for representation of proteins specifically. This indicates an additional challenge to get decent results. A problem with binary approaches is, however, that they do not include binding affinity data where it is relevant. This data is vital for providing information on the strength of a given interaction between a drug and a target. These approaches generate more realistic results, that uses binding affinity scores directly, negating the need for generation of synthetic negative samples [10]. Generally, this means that binary approaches have the benefit of less complexity, but non-binary approaches can achieve results that are better suited for real world applications.

Other models have been presented which utilize autoencoder neural networks to generate deep representations for datasets and for prediction of DTI [11][12][13][14].

In FRnet-DTI [11], two autoencoder neural networks are proposed. The first one, FRnet-Encode, handles feature generation and manipulation and the second one, FRnet-Predict, handles classification and prediction. This means that they use an autoencoder for both feature generation and another autoencoder for prediction. They employ an architectural structure from a successful classification network called GoogleNet [15] which also includes an inception module where convolution operations with various filtering sizes are done in parallel with the pooling operation. This allows them to not have to find proper filter sizes or operation types between the pooling and convolution part of their model. One issue with their double autoencoder approach is the amount of hyperparameters available for tuning is even higher, and while exhaustingly tuning the parameters would provide better results, it also comes with a heavy computational cost and more risks of overfitting.

Another autoencoder approach is seen in [12] where a multimodal deep autoencoder (MDA) is employed to fuse multiple topological similarity matrices of drugs and targets which automatically learns the low-dimensional features of drugs and targets. They then apply a deep neural network to predict the drug-target interactions. The autoencoder in this paper is used in the feature learning step only, FRnet-DTI used two autoencoders. The MDA autoencoder reduces the dimensionality of their dataset which speeds up the training of their model. Like other autoencoder approaches, it achieves significant results in the AUC and AUPR metrics. They do use a binary approach, however, which they plan to change with future work.

Chapter 3

Preliminaries

3.1 Drug discovery process

This section goes into detail about the process of drug discovery, starting from the original manual process and where machine learning can be applied to help and speed up the process. Following this, a short section will detail the general categories of machine learning approaches in drug discovery

3.1.1 Interactions and the process to achieve them

Firstly, to explain the general process, the following section goes into detail about interactions between molecules and proteins and how this is important for drug discovery. Secondly, the general process of drug discovery is explained.

The interactions of molecules and proteins are noted in DTI and refers to the binding of a drug to a target location which results in a change to its behavior or its function. The drugs mentioned here are defined as any chemical compound consisting of molecules that bring physiological change in a human body when injected, consumed, or absorbed. The targets in the definition of DTI refers to entities like proteins and the target, specifically, is a biological target which is any part of living organisms that the drug can bind to for the physiological changes [16].

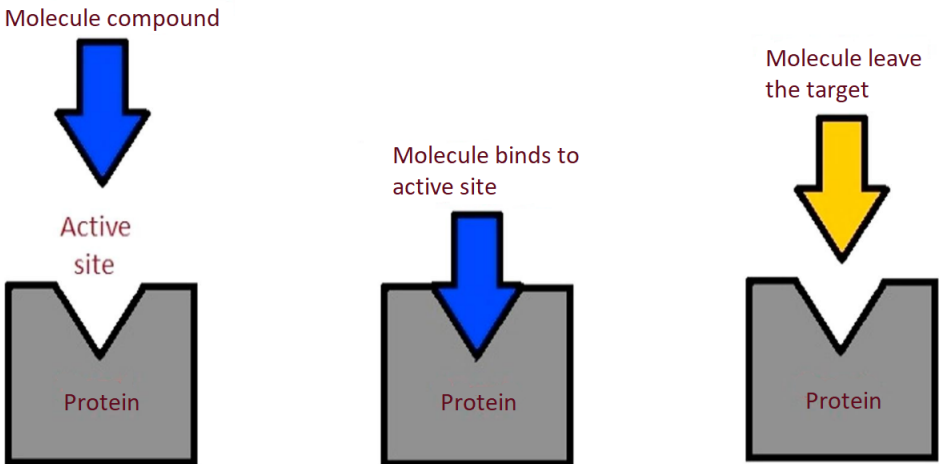

Figure 3.1: The interaction of a molecule with a protein [16]

The figure above shows how the chemical compound finds an active site in a target and then binds to the target temporarily. This forces a change physiologically, and the drug then leaves the biological target.

Prediction based on DTI can be approached as either a regression problem or a binary classification problem. Binary approaches look for a simple yes or no answer to the question "is there an interaction between this 'drug' and this 'target'?". Non-binary approaches look at the binding affinity to provide information on the interaction and can be used to see the strength of this interaction. Something which has proven to be more applicable in real world scenarios [17]. Through the papers in the related work section, it was investigated what the values of each approach are. We will go into detail on this later and what it means for our project.

The process behind DTI is presented in the following figure, which is a simplified drug discovery process. It is simplified to avoid a hugely detailed model, with a lot of parts that are less relevant to this project. Certain parts of the process have been left out like reproducibility (where experiments are reproduced to confirm results) and general testing of the drug, including clinical trials, that comes after discovery and development of the drug.

Figure 3.2: The drug discovery process simplified.

Drug discovery generally starts with target identification and validation [1]. It is here that the biological origin of a disease and its potential targets are identified. The molecule or protein is isolated and its role in the disease is determined. As part of this step, High-Throughput Screenings (HTS) are performed, where large libraries of chemicals are tested for their ability to modify the target [18]. This is a tedious process, which originally had a human screening large amount of data, involving a lot of information processing to estimate potential targets with desirable interactions. Today, robots handle this step, by screening faster than a human could. As part of this, pan-assay interference compounds are tested which are compounds that are generally observed to have common degrees of interaction with many targets. Through this part of the process, the molecular features that the pan-assay interference compounds have in common, while also interacting with the target, can be used to narrow potential targets. For validation purposes, structure-activity relationships (SAR) are determined from potential targets. SAR is also used to identify the lead compound in the next step.

In the lead identification step, a chemical lead is found. A lead is a synthetically stable, feasible, drug-like molecule. To identify this lead, SAR is used as preliminary evidence of target engagement. It is here that the lead is assessed. It should be able to bind to a specific target, but there is also other important profiling regarding absorption, distribution, metabolism, and excretion that are important to assess before continuing to the next step.

The next step is lead optimization, where a lead compound has been identified, and an iterative process starts where medical chemists can attempt to increase activity against chosen targets or reduce activity against other unrelated targets. This process requires many iterations that will (hopefully) result in a drug candidate being developed that is feasible for drug development [1].

3.1.2 Applying machine learning

There are multiple parts of the entire process where machine learning can be applied [19], but our focus is mainly around the second part, HTS, where large libraries of chemical compounds are investigated to find potential interactions. In part, machine learning could help the discovering process where molecule compounds that are likely to have a degree of interaction with the target protein are found. Furthermore, during the iteration where the desired chemical features are pinpointed, machine learning approaches can be used to more quickly find the chemical compounds that are sought after. This is also shown on the figure below.

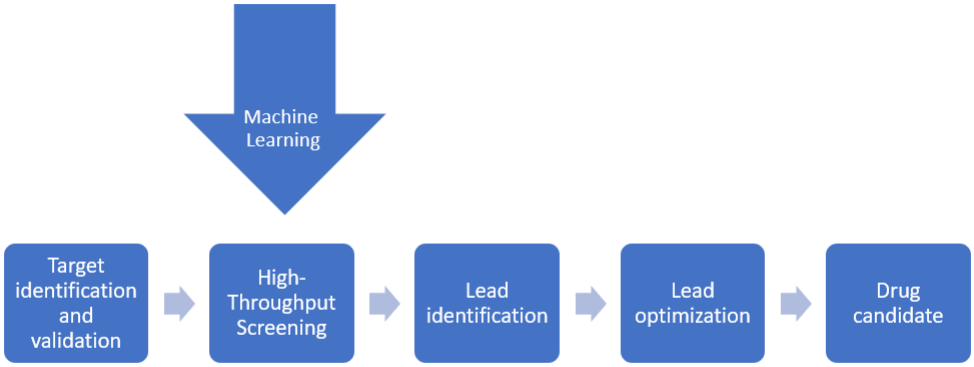

Figure 3.3: The drug discovery process with applied machine learning.

By providing the medical chemists with an automated source of information, including making predictions, the chemist will not have to rely on their own ability to go through the datasets, looking for the right compounds. Instead, they can receive a likely prediction (or several) and start from there, possibly skipping a lengthy process of going through data. It is here that machine learning approaches can aid drug discovery, and by predicting accurately potentially find more likely interactions and do so more effectively.

3.1.3 Representation of molecules

Molecules are represented in a format called SMILES (simplified molecular-input line-entry system). It represents the structure and chemical composition of a molecule using an ASCII string

3.1.4 Representation proteins

Proteins are represented in a format called FASTA (fast-all). It uses a string to represent the proteins using the amino acids single letter code. As an example, alanine would be represented using the capital letter A.

3.2 Datasets

Just like with every other machine learning project, a huge amount of data is needed to train and test the networks. The whole framework will be explained in detail in a chapter 4 but for now the important thing to note is that the system is made up of three networks, a molecule autoencoder, a protein autoencoder, and an interaction prediction network. These three networks operate on three different sets of data, and so to train and test all of them we used four publicly available and well-known datasets: DrugBank, UniProt, KIBA and Davis.

DrugBank is an online database that contains comprehensive information on drugs and targets [20]. It is constantly being updated with new information and the current version 5.1.8. which was released 03. January 2021. consists of 14,548 drug entries [21], of which we have extracted 10,763 in SMILES string format. We used these molecules to train our molecule autoencoder.

UniProt is also an online database but it contains sequences and annotations for more than 120 million proteins [22]. We have managed to extract 10,504 proteins in the FASTA string format. We used these proteins to train our protein autoencoder.

KIBA dataset was first introduced in the paper [23]. In the scope of this paper, the authors carried out a systematic evaluation of drug-target interaction and then compared their results to the data reported in the widely used databases, specifically ChEMBL and STITCH. The result of their work was an integrated drug-target bioactivity matrix across 52,498 drugs and 467 targets, including 246,088 KIBA scores, all of which has been made freely available. From this freely available dataset the paper [8] filtered it to contain only drugs and targets with at least 10 interactions yielding a total of 229 unique proteins and 2,111 unique drugs, producing 118,254 interactions pairs. These interactions are measured with KIBA scores, and while usually a lower the KIBA score suggests a stronger the interaction, the dataset available through SimBoost [8] was pre-processed so that the KIBA indication of binding affinity was reversed. This also means that the threshold for decision of whether an interaction exists was changed to be 12.1. This is needed when calculating AUPR metric explained in chapter 5.

Davis dataset was introduced in the paper Davis et. al. [24]. It has 442 proteins and 68 compounds, which yield 30,056 total interactions. This dataset used the Kd value (kinase dissociation constant) which basically is a measure of interaction strength between a drug and a protein. Following prior work we have transformed these values into log space using:

$$pK_d = -\log 10(\frac{K_d}{1e9})$$

This gives a more readable value range to us and is more comparable to KIBA scores. Now, to measure AUPR metric in chapter 5 the threshold value of 7 was used.

We used Davis and KIBA datasets to train and test our interaction prediction network, though we didn't combine the datasets, rather passed them through the network separately. We did this because the values used to measure binding affinity scores for the two datasets are not the same. Moreover,

DeepDTA and DeepGS which are projects that we want to compare our framework to did the same, experimenting and presenting findings for the datasets separately.

3.3 One hot encoding

Since neural networks can't function with anything but numerical data input, the molecules and proteins from all four datasets need to be "transformed" into a suitable representation. This is where one-hot encoding comes in. In this project it is an algorithm used to turn characters into arrays, comprised of bits where only one bit is set to 1 (hot bit) and the rest to 0 (inactive bits). The index at which this single hot bit resides in the array indicates the character the array is representing. The transformation of characters to indices is done through a dictionary. If one needs to encode a sequence of characters then a matrix is created where each row represents a character in the sequence and that row contains the array encoding. This is best visualised using an example:

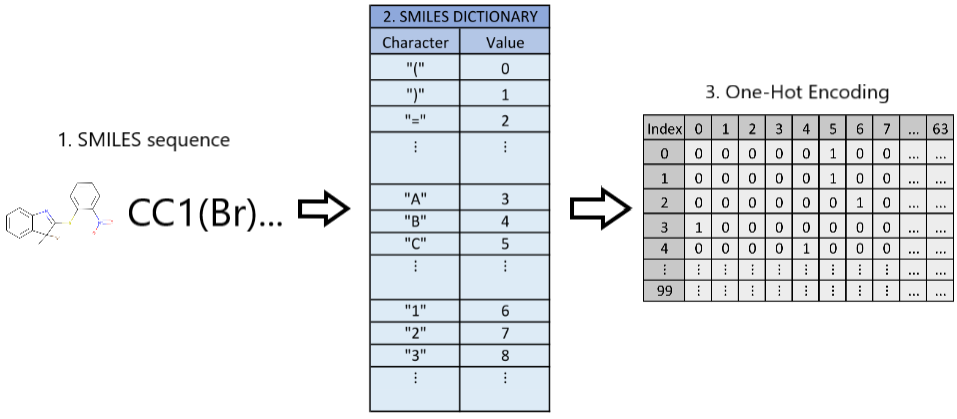

Figure 3.4: The process of encoding SMILES strings

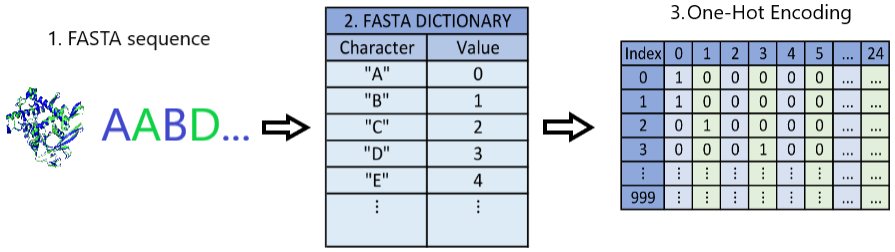

Figure 3.5: The process of encoding FASTA strings

Let's look at figures 3.4 and 3.5. On the first step we see a SMILES string representing a molecule and a FASTA string representing a protein, respectively. The process of creating a one-hot encoding for both a SMILES and a FASTA string is the same. The difference is in the dictionary used to achieve the encoding, because for FASTA strings in Davis and KIBA there are only 25 characters that

can appear in the string, while for SMILES strings there are 64 possible characters. The second step shows samples of these dictionaries, respectively.

On the last step the way that the encoding is achieved is by initializing a matrix, fully populated with zeros, of shape 100 by 64 and 1,000 by 25 for molecules and proteins, respectively. The reasoning behind the number of rows for both matrices is that those are the average lengths of SMILES and FASTA strings in KIBA and Davis datasets [10]. If the sequence is longer than the average it is cut to match the number of rows while if it's shorter there won't be a "hot" bit in the row. After matrix initialization the algorithm loops through each character in the respective sequence and notes it's position in the string which corresponds to a row number. Then, a lookup in the dictionary, for the observed character is performed, and the column number is noted. These two indexes specify which cell in the matrix will contain the "hot" bit. That way first two rows of one-hot encoding in figure 3.4 have "hot" bits on position 5 while first two rows of one-hot encoding in figure 3.5 have "hot" bits on position 0.

3.4 Autoencoders

Autoencoders are networks that are conceptually composed from two parts, an encoder and a decoder as can be seen in figure 3.6.

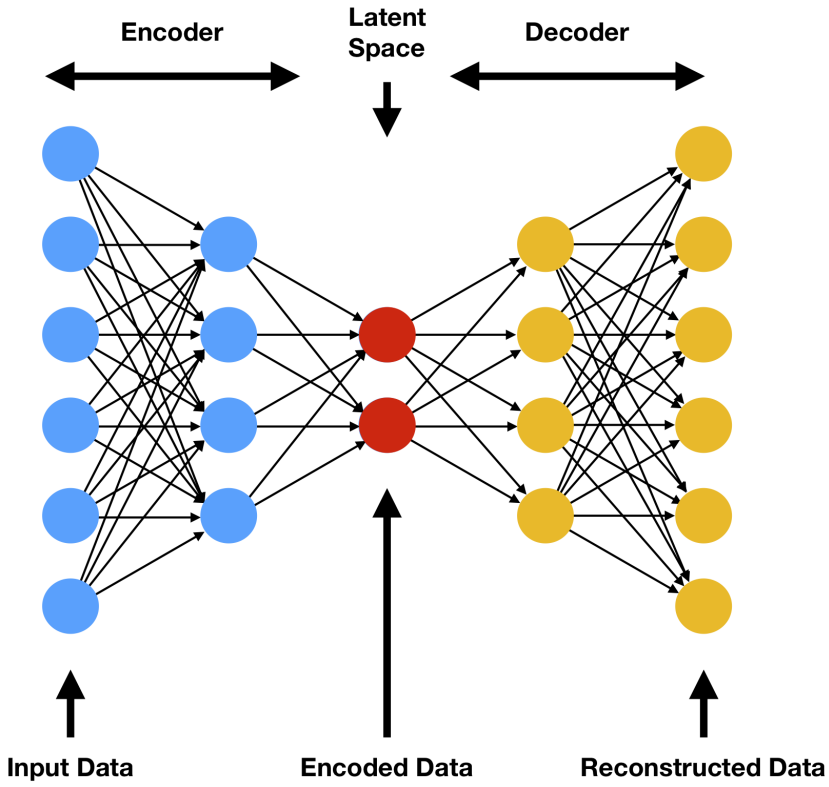

Figure 3.6: An example of a simple autoencoder [25].

The idea, at least behind the training of autoencoders, is to compress the input into latent space with the encoder, then, using the decoder, reconstruct the encoded data as similar to the input data as possible [26]. This is called unsupervised learning, because there aren't any labels to guide the learning process, which is the main advantage of using autoencoders in DTI since there is a scarcity of labelled data. Once the autoencoder is trained, depending on the use case the user would use just the encoder or both the encoder and decoder, together. The use cases are reducing the dimensionality of the input, feature extraction, denoising the input data, classification and even image generation.

3.5 Binding affinity or binary interactions

In this section, we briefly present the differences between treating DTI prediction as a regression problem and a classification problem.

3.5.1 Binary interactions

For binary interactions, interactions are defined by either happening, which is presented as a 1, or as not happening, which is presented as a 0. The datasets we use include the binding affinity of molecule-protein combinations.

The benefit of using binary interactions is that it reduces the complexity, but the downside is that it also reduces the expressiveness. This reduces the amount of information given by the network, and might be a problem for real-world applications, as explained in chapter 2.

3.5.2 Binding affinity

Unlike with binary interactions, it is also possible to use the values for binding affinities as is, without binarization thresholds. The benefit of doing this is that valuable information is preserved at the cost of increasing complexity.
Let us say that the binding affinity between a protein and a molecule is 10, while for another protein and molecule it is 15. The last protein and molecule couple would have a binding affinity value that is 50% greater than the first couple, which is valuable information, because if a protein-molecule interaction is very strong, then it is likely that a similar protein-molecule interaction would have at least a somewhat good binding affinity. When both of them are simply converted to a 1 as with binary interactions, this information is lost [10]. This is the main weakness of threshold binarization, which was also detailed in chapter 2.

The approach detailed here provides a valuable benefit, trading complexity for better real-world results. With the added complexity not being an issue, prediction using this approach is preferable for this project.

3.6 Neural networks in general

Neural Networks (NN) are machine learning approaches that are important to deep learning [27]. It gets its name because the network resembles the structure of neurons in the brain, sending signals between each other. The networks consist of input layers, one or more hidden layers and output layers. Each layer consists of neurons that are connected to form the network. The neurons, depicted in 3.7, can be considered a node which takes in a sum of multiple inputs each multiplied by an accompanying weight, adding to that sum a threshold value. Then this value is pushed through an activation function producing the output of the node.

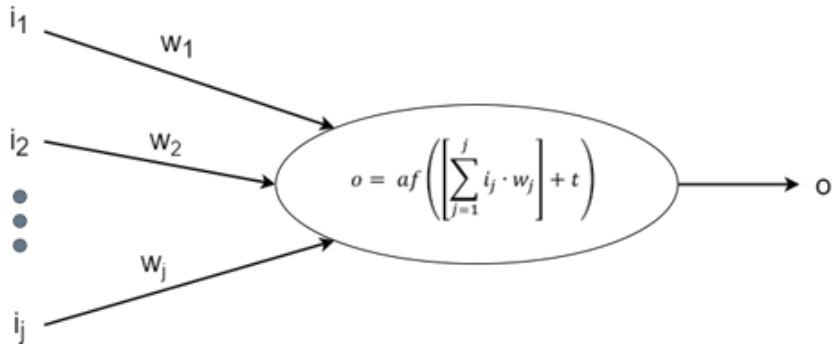

Figure 3.7: An artificial neuron

Training of the neural network is done in epochs, where all training data are used for one cycle. The number of epochs is set as a parameter before training is started.

To evaluate the network, loss functions can be used to measure the accuracy of the network and train it. The loss function calculates the error between the resulting outputs of a network and the target outputs. The goal is to minimize this loss, or rather, minimize the loss function by finding the appropriate values for the weights used in the network. This can be done by means of gradient descent. That is, we calculate the gradient of the loss function and move in the opposite direction of it which is the direction of the local minimum. We repeat this process until we calculate the gradient that is 0 meaning we found the local minimum. When the loss value has been calculated using the values from the output layer, we can see that to correct that loss, we would need to correct the activations of the neurons in the previous layer until the input layer is reached. This is due to the weights of the previous layer, that are influencing the activation of the neurons in the current layer. To deal with this, backpropagation can be used. Backpropagation is an algorithm for calculating the local error values of the hidden nodes and adjusting the corresponding weights until the input layer has been reached.

3.7 Convolutional neural networks

A convolutional neural network (CNN) is a class of deep neural networks. As with all neural networks, it consists of an input layer, one or more hidden layers, and an output layer. The input in a CNN is a tensor with a certain shape. After being encoded in the input layer, hidden layers are used to perform convolutions. The name "convolutional neural network" was given because the network employs a mathematical operation called convolution [28]. These convolutions are sets of filters. The goal of the convolutional layer is the extraction of high-level features from the input. An example of this process can be seen on figure 3.8. There we have a 6 by 6 matrix as an input with a 3 by 3 matrix as a filter. Stride here is 1 which means that each step window slides 1 cell to the right (or down) thus our output will be a 4 by 4 matrix. When we overlay the filter on the first step we see that it is over a 3 by 3 segment filled with 10s. We multiply each cell of the segment by the appropriate (overlaying) cell of the filter and then we sum the results. It our case it will be $(10*1)+(10*0)+(10*-1)+(10*1)+(10*0)+(10*-1)+(10*1)+(10*0)+(10*-1)=0$. Thus we place a 0 on the first cell of the first row of the output. Then we move to the next segment, the first two columns of which are 10s and the third column are 0s. We perform the same operation and get $(10*1)+(10*0)+(0*-1)+(10*1)+(10*0)+(0*-1)+(10*1)+(10*0)+(0*-1)=30$. That will be the second cell of the first row of the output. We keep repeating the same operation until we fill out the output matrix.

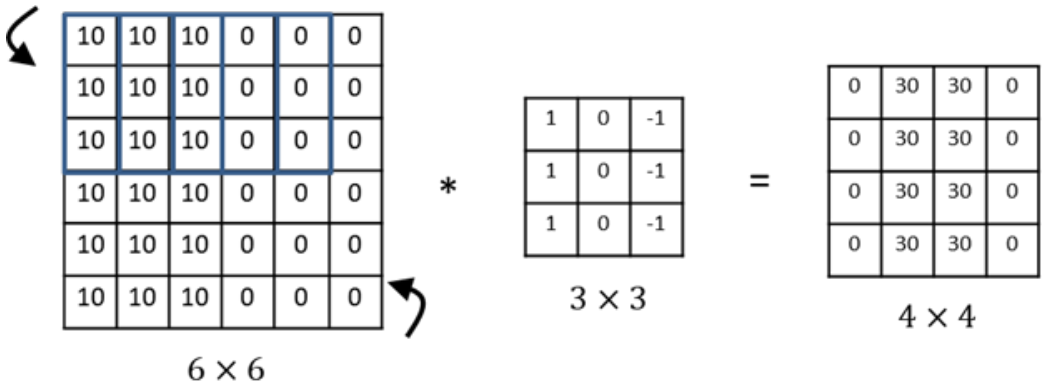

Figure 3.8: Visual representation of convolution [29].

Following convolutional layers, the results are sent to the next layer. The next layer may be a pooling layer which goes along with the convolutional layer and aims to reduce the dimensions of the data by combining the outputs of neurons in one layer into a single neuron in the next layer. Basically it reduces the number of parameters of the input tensor and thus helps to reduce overfitting as well as to extract representative features from the input tensor and to reduce the amount of computations (thus increasing efficiency). There are three main ways of pooling that extract different features. These are Max pooling, Min pooling and Avg pooling. For example, like we see on figure 3.9 the "2x2" Max pooling takes the largest value from the "2x2" squares while the Avg pooling takes the average. Pool sizes can also vary in order to reduce the dimensionality more drastically. Following this, the data is

flattened and then sent to the fully connected layers in which every neuron in one layer is connected to every neuron in another layer.

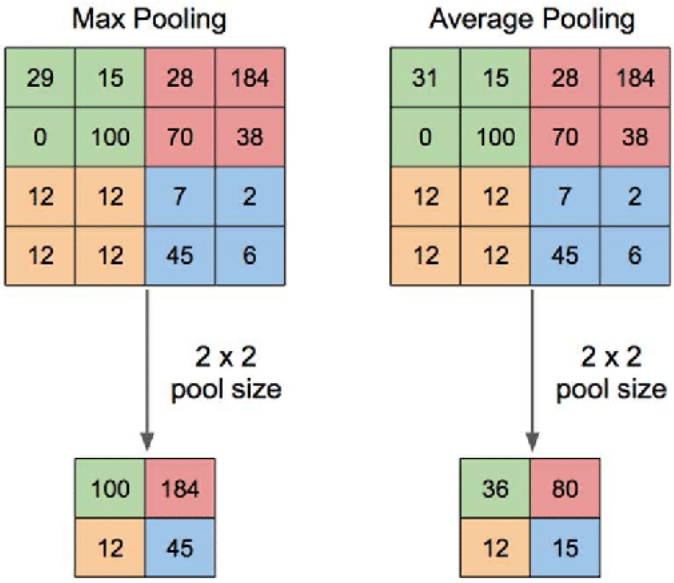

Figure 3.9: Visual representation of Max pooling and Avg pooling [30].

This fully-connected layer then takes in the result of the pooling layer and produces the result based on the features that were extracted during the previous steps.

Chapter 4

Framework

a This section will demonstrate and explain the structure of the AUEN framework. Since the framework supports different architectures, we present our three different variants of the framework and subsequently inspect them. They are:

1. AUEN CNN-CNN
2. AUEN CNN-DNN
3. AUEN RNN-DNN

The variants differ in the composition of the autoencoders, mainly the encoder and decoder, with each one being noted in the variant's name. Consequently, AUEN CNN-CNN means that both the encoder and decoder are a Convolutional Neural Network. AUEN CNN-DNN means that the encoder is a Convolutional Neural Network while the decoder is a Deep Neural Network (DNN). Finally, AUEN RNN-DNN means the encoder is a Recurrent Neural Network (RNN) and the decoder is a Deep Neural Network (DNN). To clarify, a DNN is a simple neural network (NN) without any modifications, and a RNN is a class of NN's that, as output, yield the prediction as well as their internal state, allowing them to operate on a sequence of data. All of these networks will be explained in more detail in the following sections where we present the variants that utilize them. The variants share the same interaction prediction network. Our main variant is the AUEN CNN-CNN variant and it is the focus of this report. The other two variants will be used for comparison with AUEN CNN-CNN and show the difference in performance.

4.1 Framework architecture

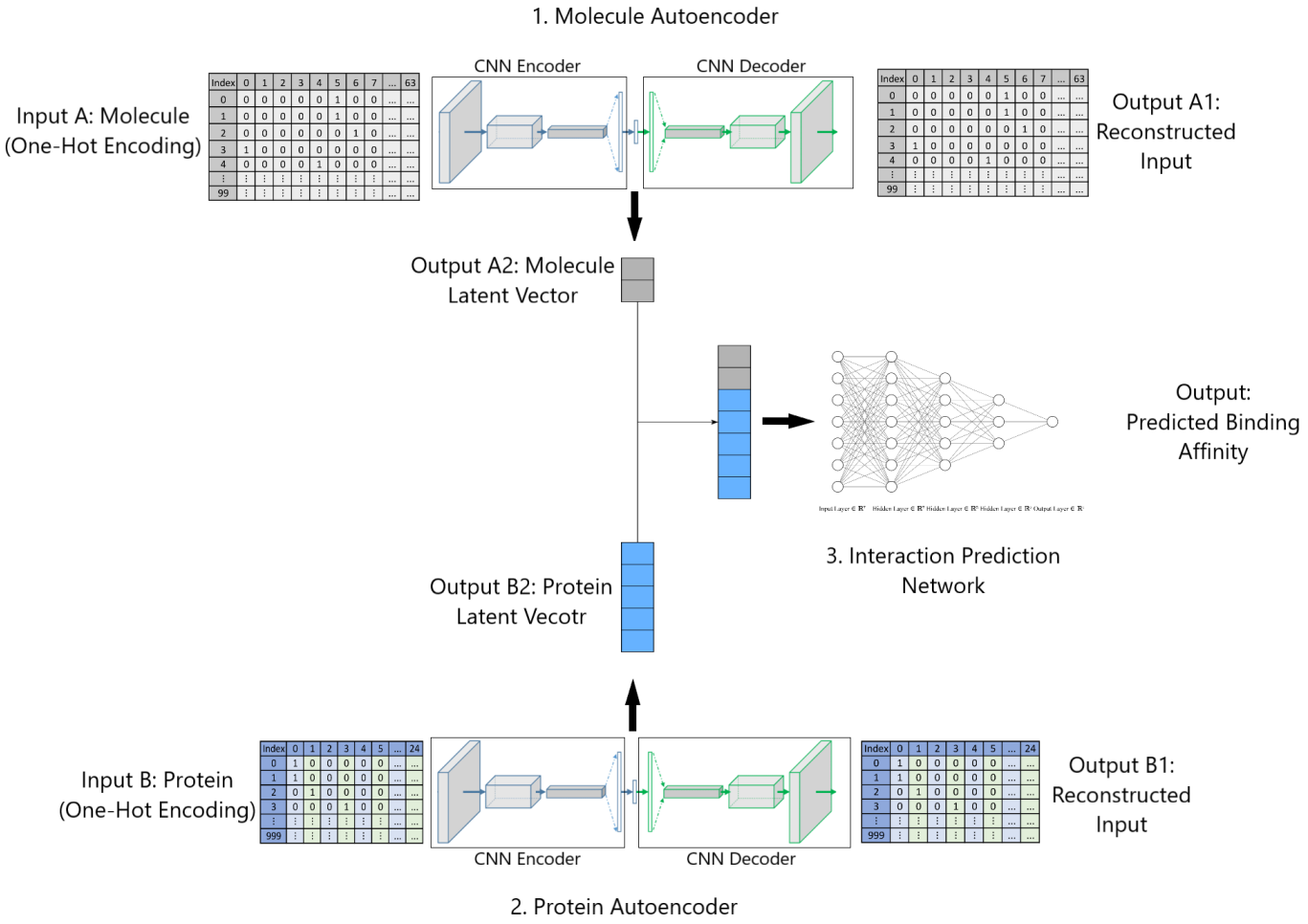

Figure 4.1: The architecture of AUEN framework with CNN-CNN

As can be seen in figure 4.1, the AUEN framework consists of three neural networks, the molecule autoencoder, the protein autoencoder, and the interaction prediction network. In the figure, the autoencoders are shown to have two outputs. This indicates that the framework has two operational settings: pre-training the autoencoders, and predicting binding affinity with the framework. To be able to do the latter, the former must be done in advance, hence the name pre-training.

4.1.1 Autoencoder pre-training

The molecules and proteins from DrugBank and UniProt datasets are first transformed into one-hot encodings. One thing to note is that one-hot encoding has two benefits. Firstly, this method ensures that machine learning does not assume that certain parts of FASTA and SMILES strings are more important than others. Secondly, due to how we use convolutional neural networks as the first part of our autoencoders, we expect them to extract information about the neighbours of "hot bits" in the encoding to discern patterns in SMILES and FASTA strings. To continue, the encodings are used as input to the molecule and protein autoencoders which are pre-trained in the same fashion as explained in chapter 3. This can be seen in figure 4.1 following the path from inputs A and B, through the encoder and decoder, to outputs A1 and B1. These outputs are the ones guiding this training process and if they are reconstructed as close to the original input as possible, the autoencoders in the framework learn to retain the information from the input while decreasing its dimensionality. This is the completion of the pre-training operational setting, and from now on the framework only uses the encoder segment of the autoencoders meaning there won't be A1 and B1 outputs anymore, rather outputs A2 and B2.

4.1.2 Predicting binding affinity with the framework

In this operational setting there are three steps the framework goes through.

1. Encode the symbols in SMILES or FASTA sequences into one-hot encodings.
2. Compress the encodings to vectors in latent space.
3. Predict the binding affinity based on the concatenated drug and target latent representations.

The first step is necessary because the autoencoders were trained on DrugBank and UniProt datasets, but now the KIBA and Davis datasets are used. For the second step the encoded molecules and proteins are then fed into their respective autoencoders. This action compresses the original input into latent vectors meaning that these vectors hold all the information from the input without large dimensional taxation. Due to this, the interaction prediction network has less work to overcome to converge on a minimum loss value. This allows our framework to be trained on less labelled data than usual and achieve good performance. The latent vectors representing the molecule and protein are then concatenated and fed to the interaction prediction network to predict binding affinity of the pair. The network is first trained on one part of a dataset and then tested on a part of the dataset it has not been trained on. The results of the tests are shown in chapter 5.

4.1.3 Interaction prediction network

Since all three variants share the same interaction prediction network we would like to explain it here. The network consists of 7 fully connected dense layers with 700, 500, 300, 100, 50, 25, and 1 as the number of neurons in each layer, from the front of the network to the back. The last layer needs to have a single neuron since this network needs to predict a single number (binding affinity). To avoid over-fitting, between each layer sits a dropout layer with a 0.1 rate which sets the activation of some neurons to 0. The activation function used throughout the network is Rectified Linear Unit (ReLU). Predicting a single continuous number as output also means the appropriate loss function is Mean Squared Error (MSE) with Adam optimizer and a default learning rate of 0.01.
In the next few sections we will showcase the different variants of the framework and their characteristics.

4.2 AUEN CNN-CNN variant

The AUEN CNN-CNN is our main variant and its architecture is displayed in figure 4.1 in the previous section. The molecule and protein autoencoders are made up of a convolutional neural network as an encoder and decoder.

Since CNNs were explained in chapter 3 we will directly dive into how they're used here.
A convolution can capture spatial features between molecule's atoms and protein's acids. We hoped to utilize this characteristic and created an encoder with with three one dimensional convolutional layers with double and triple the number of filters in each of the subsequent layers. Following each convolutional layer is a max-pooling layer, decreasing dimensionality by one half each time, and after the last one, a dense layer finishes the compression and produces the 50- and 250-dimensional vector for the molecule and protein, respectively.

We tried to make the decoder mirror the encoder as much as possible. So first, a dense layer ups the number of dimensions to 2,976 for the molecule and 864 for the protein. Then, a Reshaping layer reshapes the input to be two dimensional since the following convolutional layer only accepts that kind of input. The convolutional layers are again one dimensional and each is followed with an up-sampling layer which is the opposite of a max-pooling layer and doubles the number of dimensions except in the last one where it quadruples them. The final layer needed to be a dense one producing a vector with 6,400 and 25,000 dimensions for the molecule and protein, respectively. In the end a Reshape layer reshapes the output to match the input and a softmax layer finalizes the decompression operation by turning the numbers of each row in the matrix to probabilities that sum up to 1.

All convolutional and dense layers use Rectified Linear Unit (ReLU) as an activation function. Categorical Crossentropy was used as the loss function to calculate the error rate when training the model, and Adam optimizer with a default learning rate of 0.01 was used to minimize the loss.

4.3 AUEN CNN-DNN variant

The AUEN CNN-DNN is our second variant and its architecture very close to AUEN CNN-CNN described in the previous section. The only difference between them is that for both the molecule decoder and protein decoder the structure is not CNN but rather a DNN which is just a basic fully-connected neural network.

The idea behind that variant of the model was to check the difference the produced in the results of autoencoder training based on the structure of the decoder portion of the autoencoder. The encoder part for both molecules and proteins is identical to that of AUEN CNN-CNN. Only difference between two models is the decoder part for both molecules and proteins. For molecules autoencoder the decoder portion consists of 3 fully connected layers with dimensions of 1920, 2560 and 6400. The expected output of the decoder in our case is a 2 by 2 one-hot encoding matrix, thus we reshape the output of the last layer to 100 by 64 matrix (same as the input). For proteins the general structure looks the same, but the layer sizes are different, with 3 fully-connected layers having dimensions of 1000, 10000 and 25000. The expected output for the protein's decoder is 1000 by 25 matrix thus we also reshape the output of the last layer to fit these dimensions. In both models after the Reshape layer a softmax layer finalizes the decompression operation by turning the numbers of each row in the matrix to probabilities that sum up to 1.
First dense layers of the decoder for both molecules and proteins decoders use sigmoid activation function. The following to layers for both molecules and proteins decoders use ReLU activation function. Categorical Crossentropy was used as the loss function to calculate the error rate when training the model, and Adam optimizer with a default learning rate of 0.01 was used to minimized the loss.

4.4 AUEN RNN-DNN variant

The RNN-DNN framework builds on the general AUEN framework architecture. Where it differs is that this variant uses a Recurrent Neural Network (RNN) for encoding, and a Deep Neural Network (DNN) for decoding. This is the same for both autoencoders, one handling proteins (FASTA) and the other handling molecules (SMILES). A recurrent neural networks output depends on prior elements within the sequence. For a given string "MDSKGSSQKGSRL" the order is important, like with SMILES and FASTA strings. Inputting this into a recurrent neural network, means that the network needs to account for the order of the input on top of the given input. The figure below shows a simple recurrent neural network with three input layers at the bottom, three hidden layers in the middle and three output layers at the top. The order (i.e. the prior input) would be preserved in hidden states between layers.

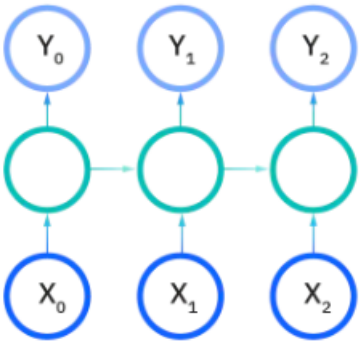

Figure 4.2: A simple RNN [31].

Furthermore, recurrent neural networks also share parameters across the layers of the network. They also utilize Backpropagation Through Time (BPTT) to determine gradients. This, in simple terms, means applying a backpropagation algorithm to train the recurrent neural network for each time step in a sequence.

For the decoder, a DNN was used, which is a basic form of a neural network.

The AUEN RNN-DNN variant has multiple layers. The RNN uses two bidirectional consecutive Long Short-Term Memory (LSTM) layers with 64 and 32 output neurons and two dense layers of shape 128 and 50 for encoding of molecules. For proteins, the variant has two consecutive LSTM layers with 64 and 32 output neurons, and a dense layer which takes a shape of 250. The LSTM layers are a unique layer variant to the RNN, which have three gates; input, output and forget. These gates control the flow of information to address a problem of long-term dependency. The DNN uses three dense layers of shapes 1000, 3200 and 6400 for molecules, and 10000 and 25000 for proteins and a reshape layer down to a 100, 64 shape to match the input for decoding. We also add a softmax layer as the last activation function, which generalizes a function to multiple dimensions. The two latent vector outputs are then fed to the prediction network, as they are in all AUEN variants.

4.5 Base model

In addition to AUEN framework we created a base model with the intention of checking if more complex networks were even needed in order to predict molecule protein interactions. This model utilized the same network architecture as the interaction prediction network of the framework. The same KIBA and Davis datasets were used but with two encoding styles, one being one-hot encoding and the other Mol2Vec and ProtVec featuirzers. The results for both versions of the base model can be found in chapter 5.

4.5.1 Featurizers

The reason for using a featurizer is to convert information about the molecules and proteins into something that is more easily digested by a neural network and to improve performance. Featurizers work in different ways but, usually, they either featurize the string representation of the molecule or the protein, or they featurize the 3D structure. Some featurizers also only look for specific things that relate to the binding of the molecules and proteins while omitting other parts entirely.

Mol2Vec

Mol2Vec [32] is a molecule featurizer. It takes the SMILES string of a molecule and creates a vector from it. Mol2vec is an unsupervised model. The user can either train it themselves or use the pre-trained model, which is what we have chosen to do. Mol2Vec is inspired by word2Vec [33] which is a model that can process natural languages.

ProtVec

ProtVec [34] featurizes protein input in the form of a FASTA string and outputs vectors. ProtVec is similar to Mol2Vec [32] in the sense that they are both based on Word2Vec [33], which makes it possible to concatenate the resulting vectors.

Chapter 5

Experiments

5.1 Methodology

We will now go into the methodology before going into our results. We will start out by going through the hyperparameter settings that were used for getting our results. For the most part, the variants have been trained with 100 epochs, as well as the base models, with the exception of subset testing which was trained with 200 epochs. The batch size was always set to 256.
We use 84% of the dataset for training and the remaining 16% for testing
To run the tests we used a local desktop with the following hardware:

- OS: Windows 10 home build: 19042.1052
- CPU: i5-6600K, 4 core / 4 threads, running at 4.1GHz
- GPU: Nvidia gtx 1070, founder's edition
- RAM: 16 GB running at 2400 MHz in dual channel
- Motherboard: Z170
- Storage: NVMe ADATA SX8200PNP

The runtime for doing all tests in a row is about 18 hours with the biggest limiting factor being the GPU.

5.2 Metrics

To measure the performance of our system we have used several widely accepted metrics. Same metrics were used in papers DeepDTA and DeepGS, which means we can easily compare our results to their results.

The First measure is the loss function, MSE, which is mean square error sometimes also called mean square deviation. MSE is the average squared difference between the estimated values (predicted values) and the actual value (ground truth). This metric is defined by a following formula:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

Where $\hat{y}_i$ is the predicted value, y_i is the ground-truth value, and N represents the number of drug-target pairs. Main feature of mean squared error is that it is always non-negative, and values closer to zero are better. Zero MSE would mean that there is no difference between prediction and the ground truth.

Concordance index is a metric to evaluate the predictions made by an algorithm. It is defined as the proportion of concordant pairs divided by the total number of possible evaluation pairs. CI is defined by a following formula:

$$CI = \frac{1}{Z}\sum_{y_i<y_j}\zeta(f_i - f_j)$$

$\zeta(x)$ is a step function defined like this: $\zeta(x) = \{1, x > 0; 0.5, x = 0; 0, x < 0\}$. "$Z$ is a normalization constant that equals the number of drug-target pairs with different binding affinity values. More specifically, when $y_i > y_j$, a positive score is given if and only if the predicted $f_i > f_j$" [6]. In other words the CI measures whether the predicted binding affinity values rank the corresponding drug-target interactions in the same order as the ground-truth does.

"The metric r2m is used to evaluate the external prediction performance of QSAR (Quantitative Structure-Activity Relationship) models" [6]. It is defined by the following formula

$$r_m^2 = r^2(1 - \sqrt{r^2 - r_0^2})$$

"and a model is acceptable if and only if $r_m^2 \geq 0.5$. r^2 and r_0^2 represent the squared correlation coefficient values between the observed and predicted values with and without intercept, respectively" [6]. The correlation coefficient always takes a value between -1 and 1, with -1 or 1 indicating perfect correlation (all points would lie along a straight line in this case). A positive correlation indicates a positive association between the variables (increasing values in one variable corresponds to increasing values in the other variable), while a negative correlation indicates a negative association between the variables (increasing values in one variable corresponds to decreasing values in the other variable). A correlation value close to 0 indicates no association between the variables. The square of the correlation coefficient, r^2 represents the fraction of the variation in one variable that may be explained by the other variable. The intercept of the fitted line is a value such that the line passes through the center

of mass (x, y) of the data points.

AUPR or area under the precision-recall curve (sometimes also referred to as AUCPR). AUPR is defined as an actual area under the curve and thus is not easy to define with a formula. The precision-recall curve shows the trade-off between precision and recall for different thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall). Precision-recall (PR) curves, like the closely related receiver operating characteristic (ROC) curves, are an evaluation tool for binary classification that allows the visualization of performance at a range of thresholds. PR curves are increasingly used in the machine learning community, particularly for imbalanced data sets where one class is observed more frequently than the other class

5.3 Results

5.3.1 Comparison to other models

To determine whether our framework is performing well or not we have found several other models for binding affinity prediction. KronRLS and SimBoost are older systems that are frequently used as a baseline. DeepCPI [35] is a model that uses a graph neural network for molecules and a convolutional neural network to produce embeddings for the proteins. It also is an older model but more successful than KronRLS and SimBoost. DeepDTA and DeepGS are current state-of-the-art systems that produce the best results and the ones we took inspiration from thus we mainly are going to compare the results of our framework to their results. As can be seen from tables 5.1 and 5.2 none of our variants actually perform better than the baseline. On CI, MSE and r2m metrics, our framework performs worse, however our AUPR score is much better. That means that while our variants do not produce precise binding affinity scores, they are very good at actually predicting interaction or lack of interaction between the molecules and the proteins in a binary fashion.

Model name	Molecules	Proteins	CI	MSE	r2m	AUPR
KronRLS	Pubchem	S-W	0,782	0,411	0,342	0,635
SimBoost	Pubchem	S-W	0,836	0,222	0,629	0,76
DeepCPI	GNN	CNN(Embedding)	0,852	0,211	0,657	0,782
DeepDTA	CNN	CNN(One-hot)	0,863	0,194	0,673	0,788
DeepGS	GAT+Smi2Vec	CNN(Prot2Vec)	0,86	0,193	0,684	0,801
Base model	Mol2Vec	ProtVec	0.8	0.332	0.49	0.782
AUEN CNN-CNN	CNN	CNN	0.633	0.629	0.107	0.760
AUEN CNN-DNN	CNN	DNN	0.717	0.508	0.245	0.749
AUEN RNN-DNN	RNN	DNN	0.533	0.661	0.03	0.759

Table 5.1: Metrics for our variants compared to the baseline models for the KIBA dataset

Model name	Molecules	Proteins	CI	MSE	r2m	AUPR
KronRLS	Pubchem	S-W	0,871	0,379	0,407	0,661
SimBoost	Pubchem	S-W	0,872	0,282	0,644	0,709
DeepCPI	GNN	CNN(Embedding)	0,867	0,293	0,607	0,705
DeepDTA	CNN	CNN(One-hot)	0,878	0,261	0,63	0,714
DeepGS	GAT+Smi2Vec	CNN(Prot2Vec)	0,882	0,252	0,686	0,763
Base model	Mol2Vec	ProtVec	0.846	0.333	0.498	0.929
AUEN CNN-CNN	CNN	CNN	0.723	0.588	0.261	0.905
AUEN CNN-DNN	CNN	DNN	0.686	0.663	0.146	0.897
AUEN RNN-DNN	RNN	DNN	0.608	0.745	0.036	0.897

Table 5.2: Metrics for our variants compared to the baseline models for the Davis dataset

5.3.2 KIBA and Davis Subset Testing

After all the experiments on full datasets were finished and the conclusion was drawn from the results, we decided to attempt to leverage another strength of the autoencoders. The fact that they do not need labelled data to train which is great when labeled data is sparse or unavailable. Due to the way autoencoders are built into our framework we can train them separately on lots of unlabelled data to refine the representations they produce and then hopefully obtain better results than the baseline models on reduced KIBA and Davis datasets.

The DeepDTA paper [10] also employs a similar technique, where they partition their dataset into six subsets and one part is chosen as an independent test set. They do this to learn a generalized model. We employ a similar technique with our framework to utilize a strength of autoencoders. We took ten thousand interactions from both KIBA and Davis and trained the DeepDTA model on those interactions. For our framework, we pre-trained the autoencoders on 10,504 proteins and 10,763 molecules and then at the second stage passed the same ten thousand interactions from each dataset. As can be concluded from the results in tables 5.3 and 5.4 our framework is still beaten by DeepDTA by a sizable margin. The lower performance hints at an issue with the representations that our autoencoders produced and suggests that they need improvement to reach better scores in the metrics we use.

Model name	Molecules	Proteins	CI	MSE	r2m	AUPR
AUEN CNN-CNN	CNN	CNN	0.696	0.567	0.189	0.754
DeepDTA Reduced Dataset	CNN	CNN(One-hot)	0.772	0.371	N/A	N/A

Table 5.3: Metrics for our framework compared to the DeepDTA trained on reduced KIBA dataset

Model name	Molecules	Proteins	CI	MSE	r2m	AUPR
AUEN CNN-CNN	CNN	CNN	0.722	0.637	0.215	0.902
DeepDTA Reduced Dataset	CNN	CNN(One-hot)	0.772	0.371	N/A	N/A

Table 5.4: Metrics for our framework compared to DeepDTA. Both trained on reduced Davis dataset

As mentioned in the methodology section of this chapter, this reduced version of our AUEN CNN-CNN variant was trained using 200 epochs instead of 100 used for the full set. The reason behind this was that the loss for especially molecules had not converged yet and was still dropping at a good rate at 100 epochs. This can also be seen in figure 5.1 and figure 5.2

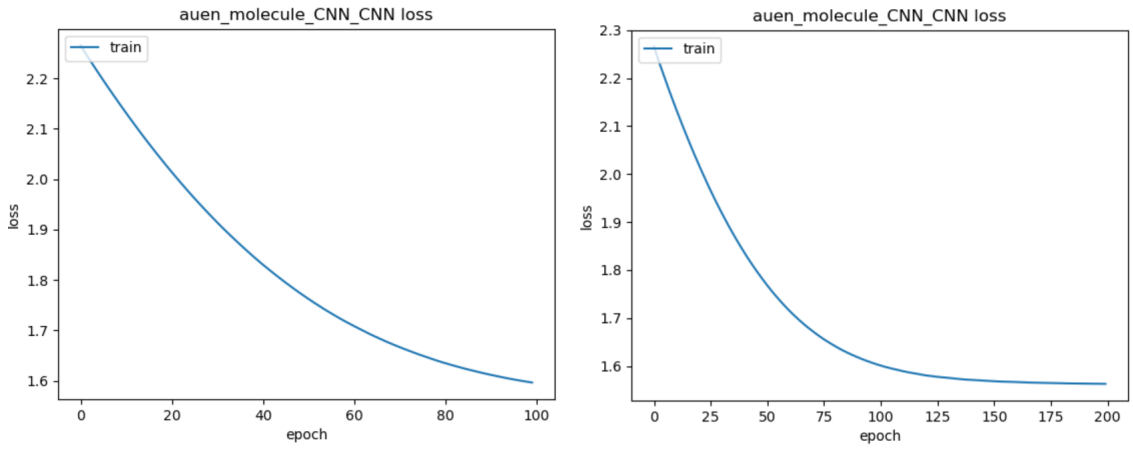

Figure 5.1: Comparison of the loss for molecules with 100 epochs in the figure to the left and 200 epochs in the figure to the right

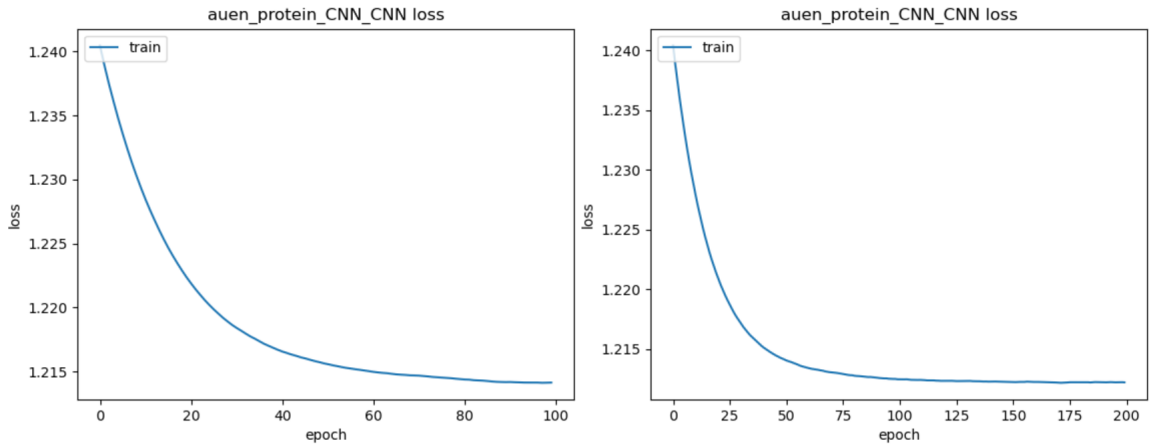

Figure 5.2: Comparison of the loss for proteins with 100 epochs in the figure to the left and 200 epochs in the figure to the right

In addition to this, going from 100 to 200 epochs also proved beneficial to the CI score and loss of the interaction model. Especially for the kiba dataset as can be seen in figure 5.3 and figure 5.4

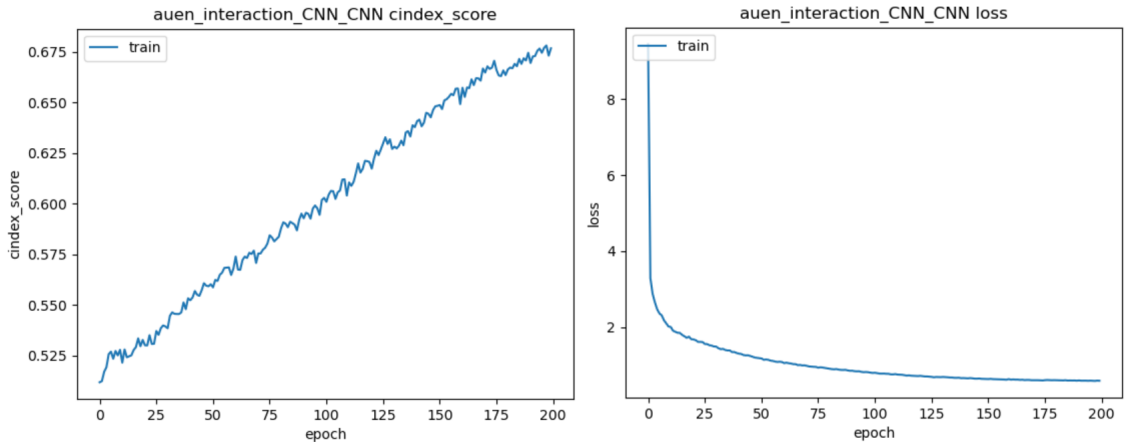

Figure 5.3: Cindex and loss of the interaction model with 200 epochs on the KIBA dataset

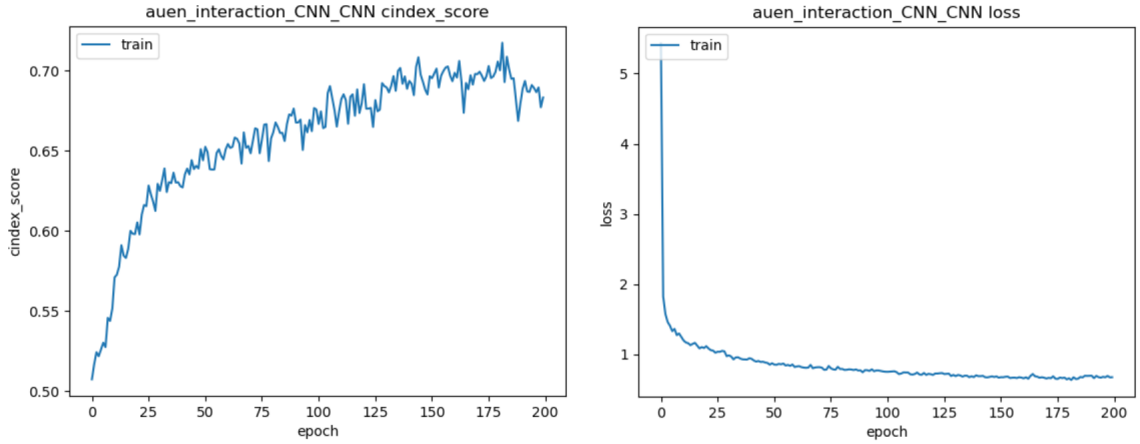

Figure 5.4: Cindex and loss of the interaction model with 200 epochs on the Davis dataset

5.3.3 Base model testing

From here, we decided to investigate the reasons behind low performance of the system and we started with checking how good the representations produced by the autoencoders actually are. We did this by comparing our AUEN CNN-CNN variant with our baseline model with both of them using the same data with reduced labels, the results of which can be seen in table 5.5 and table 5.6. From these tables, we can see that using fewer labels has a large impact on the performance of the base model, while the negative impact on the AUEN CNN-CNN variant is not nearly as bad. This is especially noticeable when comparing the MSE between the AUEN CNN-CNN variant and our baseline model.

Model name	Molecules	Proteins	CI	MSE	r2m	AUPR
AUEN CNN-CNN	CNN	CNN	0.696	0.567	0.189	0.754
Base Model	One-Hot Encoding	One-Hot Encoding	0.670	5.956	0.181	0.756

Table 5.5: Table comparing AUEN CNN-CNN to base model on KIBA dataset

Model name	Molecules	Proteins	CI	MSE	r2m	AUPR
AUEN CNN-CNN	CNN	CNN	0.722	0.637	0.215	0.902
Base Model	One-Hot Encoding	One-Hot Encoding	0.724	1.726	0.175	0.897

Table 5.6: Table comparing AUEN CNN-CNN to base model on Davis dataset

5.4 Discussion

Overall, we can draw several conclusions from the results of our experiments. Firstly, the autoencoders show feature quality comparable to plain one-hot encoding which means that the representations they produce are not very useful. Secondly, their features are not good for prediction of continuous value (in our case the binding affinity) as with their features as input our framework produces high mean squared error. That means they are not good for regression tasks. However, when less labelled data is used our framework starts outperforming the plain one-hot encoding meaning that auto-encoders could prove useful in the case where labelled data is not available as could also been seen by DeepDTA still having good performance even with less labels. Lastly, an unlikely occurrence is that the AUPR score of our framework is very good, occasionally surpassing the DeepDTA and DeepGS models. That means that the features extracted by the autoencoders work well for classification tasks due to the binary nature of the problem. Another thing to note is that the hardware requirements for this type of modelling are rather high. As an example, we looked at implementing BERT [36], however, upon doing a bit of research we found out that the GPU were are using would not be powerful enough as it simply did not have enough VRAM, and even if that hadn't been an issue it would have been running very slowly. While it was not quite as severe for our framework, we still used up almost all the VRAM that our GPU had to offer. One potential solution to this problem, however, could be cloud computing, although that requires more funding.

Chapter 6

Conclusion

We have proposed a drug discovery framework, that utilizes autoencoder neural networks and a prediction neural network to predict binding affinity for molecules and proteins. By predicting binding affinity, our predictions provide more information about the bindings, which brings better real-world application. We have implemented and tested multiple classes of autoencoders, including a CNN encoder with a CNN decoder, CNN encoder with a DNN decoder and RNN encoder with a DNN decoder. We also utilized Mol2Vec and ProtVec to assist with featurizing the molecules and proteins, respectively in one of our base models. Kiba and davis datasets were used for our project. We compare this framework to existing state-of-the-art approaches in drug discovery, including DeepDTA, DeepGS, DeepCPI, SimBoost and KronRLS. We have found that our model does not produce results that were comparable or better than the models that we compared it to. Generally, with the autoencoder approach, we found that autoencoders did not provide a boost in general performance across the metrics for this implementation. However, the AUPR score of the base model proved to be higher than the competition in general with scores 0.897 and 0.902 for our model, specifically for the Davis dataset, proving that the proposed framework did achieve good precision for binary classification.

In contribution to the field, our project shows an implementation of an autoencoder neural network in drug discovery, that provided good AUPR scores, but otherwise was unable to compete with existing state-of-the-art models.

6.1 Future work

For future work we leave further experimentation with the whole framework, by trying out different compression ratios, and fine-tuning of all hyper parameters to improve the results, implement "early stopping" to dynamically stop the training when the system reaches an optimal number of epochs. We also need to investigate the exact reason for all of our variants scoring poorly on some metrics. In particular, the MSE is rather high (compared to the models presented in related works) for all of our framework variants. To address that we would like to try different internal designs of the autoencoders. One other possible area where we can improve is encoding. Instead of using one-hot encoding for molecules and proteins there is an option of utilizing specialised dictionaries, in the same way as DeepGS did [6], to produce different encoding which might affect feature extraction by the autoencoders. Lastly we would like to try using transformers instead of autoencoders, specifically Bidirectional Encoder Representations from Transformers (BERT). BERT is a new "state of the art" Natural Language Processing (NLP) model that showed great results surpassing the competition [36]. We hypothesise that if BERT is implemented properly it might give good results for representation learning.

Bibliography

[1] Amol Deore, Jayprabha Dhumane, Rushikesh Wagh, and Rushikesh Sonawane. "The Stages of Drug Discovery and Development Process". In: *Asian Journal of Pharmaceutical Research and Development* 7 (Dec. 2019), pp. 62–67. DOI: 10.22270/ajprd.v7i6.616.

[2] *FDA*. visited 21-5-2021. https://www.fda.gov/drugs/new-drugs-fda-cders-new-molecular-entities-and-new-therapeutic-biological-products/novel-drug-approvals-2020.

[3] *EMA*. visited 21-5-2021. https://www.ema.europa.eu/en/about-us/what-we-do/authorisation-medicines/medicine-evaluation-figures.

[4] Hailin Chen and Zuping Zhang. "A Semi-Supervised Method for Drug-Target Interaction Prediction with Consistency in Networks". In: *PloS one* 8 (May 2013). DOI: 10.1371/journal.pone.0062975.

[5] Maryam Bagherian, Elyas Sabeti, Kai Wang, Maureen A Sartor, Zaneta Nikolovska-Coleska, and Kayvan Najarian. "Machine learning approaches and databases for prediction of drug–target interaction: a survey paper". In: *Briefings in Bioinformatics* 22.1 (Jan. 2020).

[6] Xuan Lin, Kaiqi Zhao, Tong Xiao, Z. Quan, Z. Wang, and Philip S. Yu. "DeepGS: Deep Representation Learning of Graphs and Sequences for Drug-Target Binding Affinity Prediction". In: *ArXiv* (2020).

[7] Andreas Laugård Hald, Christian Galasz Nielsen, Alexandr Dyachenko, and Dominik Tabak. *Drug Discovery: Framework for querying drug candidates based on similarity*. 2021.

[8] 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". In: *Journal of Cheminformatics* (Apr. 2017).

[9] André C. A. Nascimento, Ricardo B. C. Prudêncio, and Ivan G. Costa. "A multiple kernel learning algorithm for drug-target interaction prediction". In: *BMC Bioinformatics* 17 (Jan. 2016).

[10] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. "DeepDTA: deep drug–target binding affinity prediction". In: *Bioinformatics* 34 (Sept. 2018).

[11] Farshid Rayhan, Sajid Ahmed, Zaynab Mousavian, Dewan Md Farid, and Swakkhar Shatabda. "FRnet-DTI: Deep convolutional neural network for drug-target interaction prediction". In: *Heliyon* 6.3 (2020), e03444. DOI: https://doi.org/10.1016/j.heliyon.2020.e03444. URL: https://www.sciencedirect.com/science/article/pii/S2405844020302899.

[12] Huiqing Wang, Jingjing Wang, Chunlin Dong, Yuanyuan Lian, Dan Liu, and Zhiliang Yan. "A Novel Approach for Drug-Target Interactions Prediction Based on Multimodal Deep Autoencoder". In: *Frontiers in Pharmacology* 10 (2020), p. 1592. DOI: 10.3389/fphar.2019.01592.

[13] Zhixian Liu, Qingfeng Chen, Wei Lan, Haiming Pan, Xinkun Hao, and Shirui Pan. "GADTI: Graph Autoencoder Approach for DTI Prediction From Heterogeneous Network". In: *Frontiers in Genetics* 12 (2021), p. 490. DOI: 10.3389/fgene.2021.650821. URL: https://www.frontiersin.org/article/10.3389/fgene.2021.650821.

[14] Meriem Bahi and Mohamed Batouche. "Deep semi-supervised learning for DTI prediction using large datasets and H2O-spark platform". In: (2018), pp. 1–7. DOI: 10.1109/ISACV.2018.8354081.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper With Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2015.

[16] Kanica Sachdev and Manoj Kumar Gupta. "A comprehensive review of feature based methods for drug target interaction prediction". In: *Journal of Biomedical Informatics* 93 (2019), p. 103159. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2019.103159. URL: https://www.sciencedirect.com/science/article/pii/S1532046419300772.

[17] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. "Toward more realistic drug–target interaction predictions". In: *Briefings in Bioinformatics* 16.2 (Apr. 2014), pp. 325–337. ISSN: 1467-5463. DOI: 10.1093/bib/bbu010. URL: https://doi.org/10.1093/bib/bbu010.

[18] Jayme L Dahlin and Michael A Walters. "The essential roles of chemistry in high-throughput screening triage." In: *Future Med Chem* 6, 11 (2015), pp. 1265–1290. DOI: 10.4155/fmc.14.60.

[19] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, et al. "Applications of machine learning in drug discovery and development". In: *Nature Reviews Drug Discovery* 18 (2019), pp. 463–477. DOI: https://doi-org.zorac.aub.aau.dk/10.1038/s41573-019-0024-5.

[20] David S Wishart, Yannick D Feunang, An C Guo, et al. "DrugBank 5.0: a major update to the DrugBank database for 2018". In: *Nucleic Acids Research* 46.D1 (Nov. 2017), pp. D1074–D1082. DOI: 10.1093/nar/gkx1037.

[21] *DrugBank*. visited 15-6-2021. https://go.drugbank.com/about.

[22] The UniProt Consortium. "UniProt: a worldwide hub of protein knowledge". In: *Nucleic Acids Research* 47.D1 (Nov. 2018), pp. D506–D515. DOI: 10.1093/nar/gky1049.

[23] Jing Tang, Agnieszka Szwajda, Sushil Shakyawar, Tao Xu, Petteri Hintsanen, Krister Wennerberg, and Tero Aittokallio. "Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis". In: *J Chem Inf Model.* (2014).

[24] Mindy Davis, Jeremy Hunt, Sanna Herrgard, Pietro Ciceri, Lisa Wodicka, Gabriel Pallares, Michael Hocker, Daniel Treiber, and Patrick Zarrinkar. "Comprehensive analysis of kinase inhibitor selectivity". In: *Nat Biotechnol* 29 (2011), pp. 1046–1051.

[25] Pier Paolo Ippolito. *Feature Extraction Techniques*. visited 17-6-2021. https://towardsdatascience.com/feature-extraction-techniques-d619b56e31be.

[26] Noam Koenigstein Dor Bank and Raja Giryes. "Autoencoders". In: (2020). arXiv: 2003.05991 [cs.LG].

[27] *Neural Networks by IBM cloud education*. visited 17-6-2021. https://www.ibm.com/cloud/learn/neural-networks.

[28] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network". In: (2017), pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[29] *Machine Learning - Max Average Pooling*. visited 17-6-2021. https://dev.to/sandeepbalachandran/machine-learning-max-average-pooling-1366.

[30] Muhamad Yani, S Irawan, and M.T. S.T. "Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail". In: *Journal of Physics: Conference Series* 1201 (May 2019), p. 012052. DOI: 10.1088/1742-6596/1201/1/012052.

[31] *Recurrent Neural Networks by IBM cloud education*. visited 17-6-2021. https://www.ibm.com/cloud/learn/recurrent-neural-networks.

[32] Sabrina Jaeger, Simone Fulle, and Samo Turk. "Mol2vec: Unsupervised Machine Learning Approach with Chemical Intuition". In: *Journal of Chemical Information and Modeling* 58 (2018), pp. 27–35.

[33] Greg Corrado Tomas Mikolov Kai Chen and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: (2013). arXiv: 1301.3781. URL: https://arxiv.org/pdf/1301.3781.pdf.

[34] Asgari Ehsaneddin and Mofrad M.R.K. "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics". In: *PLoS ONE 10(11)* (2015).

[35] Masashi Tsubaki, Kentaro Tomii, and Jun Sese. "Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences". In: *Bioinformatics* 35.2 (July 2018), pp. 309–318. DOI: 10.1093/bioinformatics/bty535.

[36] Kenton Lee Jacob Devlin Ming-Wei Chang and Kristina Toutanova.
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".
In: (Oct. 2018). arXiv: 1810.04805. URL: https://arxiv.org/abs/1810.04805.