Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
2021 | LIU-IDA/STAT-A--21/026--SE

Multi-task regression QSAR/ QSPR prediction utilizing text-based Transformer Neural Network and single-task using feature-based models

Spyridon Dimitriadis

Supervisor : Dr. Sanjiv Dwivedi
Examiner : Dr. Oleg Sysoev

External supervisor : Dr. Esben Jannik Bjerrum

LINKÖPING UNIVERSITY

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

With the recent advantages of machine learning in cheminformatics, the drug discovery process has been accelerated; providing a high impact in the field of medicine and public health. Molecular property and activity prediction are key elements in the early stages of drug discovery by helping prioritize the experiments and reduce the experimental work. In this thesis, a novel approach for multi-task regression using a text-based Transformer model is introduced and thoroughly explored for training on a number of properties or activities simultaneously. This multi-task regression with Transformer based model is inspired by the field of Natural Language Processing (NLP) which uses prefix tokens to distinguish between each task. In order to investigate our architecture two data categories are used; 133 biological activities from the ExCAPE database and three physical chemistry properties from MoleculeNet benchmark datasets.

The Transformer model consists of the embedding layer with positional encoding, a number of encoder layers, and a Feedforward Neural Network (FNN) to turn it into a regression problem. The molecules are represented as a string of characters using the Simplified Molecular-Input Line-Entry System (SMILES) which is a 'chemistry language' with its own syntax. In addition, the effect of Transfer Learning is explored by experimenting with two pretrained Transformer models, pretrained on 1.5 million and on 100 million molecules. The text-base Transformer models are compared with a feature-based Support Vector Regression (SVR) with the Tanimoto kernel where the input molecules are encoded as Extended Connectivity Fingerprint (ECFP), which are calculated features.

The results have shown that Transfer Learning is crucial for improving the performance on both property and activity predictions. On bioactivity tasks, the larger pretrained Transformer on 100 million molecules achieved comparable performance to the feature-based SVR model; however, overall SVR performed better on the majority of the bioactivity tasks. On the other hand, on physicochemistry property tasks, the larger pretrained Transformer outperformed SVR on all three tasks. Concluding, the multi-task regression architecture with the prefix token had comparable performance with the traditional feature-based approach on predicting different molecular properties or activities. Lastly, using the larger pretrained models trained on a wide chemical space can play a key role in improving the performance of Transformer models on these tasks.
Keywords: multi-task regression, QSAR, QSPR, attention based models, deep learning, transfer learning.

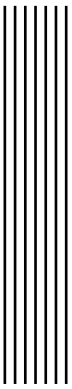
Acknowledgments

I would like to thank my external supervisor at AstraZeneca Esben Jannik Bjerrum for his continuous support, trust and guidance throughout my thesis work and for coming up with the original idea for this project. Also, I would like to thank Ross Irwin from AstraZeneca who provide me with the powerful pretrained Transformer models. I would like to thank my examiner Oleg Sysoev and my supervisor Sanjiv Dwivedi for their valuable comments of this work. Lastly, I am thankful to my family who supported me to pursue my goals.

Contents

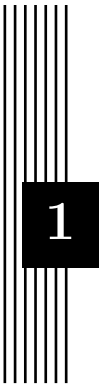
List of Figures

List of Tables



List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
BART	Bidirectional and Auto-RegressiveTransformers
BERT	Bidirectional Encoder Representations from Transformers
DNN	Deep Neural Network
ECFP	Extended Connectivity Fingerprint
ESOL	Estimating aqueous Solubility
ExCAPE	Exascale Compound Activity Prediction Engine
FNN	Feedforward Neural Network
GP	Gaussian Process
GPT3	Generative Pre-trained Transformer 3
LM	Language Model
MLP	Multilayer Perceptron
MSE	Mean Square Error
NLP	Natural Language Processing
QSAR	Quantitative Structure-Activity Relationship
QSPR	Quantitative Structure–Property Relationship
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
SMILES	Simplified Molecular-Input Line-Entry System
SVM	Support Vector Machine
SVR	Support Vector Regression



1 Introduction

1.1 Background

The scientific field of cheminformatics combines aspects of fields as mathematics, informatics and machine learning to address problems in chemistry. With the rapid growth of available chemical data and the successes of machine learning; the focus of solving chemical problems has turned into AI-driven approaches [16]. In basic terminology of chemistry, a molecule consists of two or more elements (atoms) which could be the same or not, and a chemical compound consists of two or more different elements. Molecules are composed of atoms connected with chemical bonds. Cheminformatics tries to address different problems based on molecules.

Using computational tools drug discovery process can be accelerated and the number of trials can be reduced. This would lead to saving time and money on this process, having a high impact in the field of medicine and in public health. Some chemical problems related to drug discovery are property and activity prediction [47, 49], organic synthesis [10], de novo generation and others. Property prediction focuses on estimating a molecular property (usually a continuous value) from its structure, for example how much a molecule dissolves in water. In organic synthesis, a complex molecule is generated from more simple compounds or vice versa (synthesis, retrosynthesis) [10]. The de novo design methods generate molecules similar to known; giving the ability to create new drug-like structures [41].

The molecular chemical formulas need to be represented in a specific format to be used along with machine learning models. In the literature there exist different ways to represent a molecule to make them compatible with machine learning methods [22]. One way to represent molecules is to create handcrafted features from the molecular structure, also known as descriptors, which require professional knowledge. Examples of these descriptors could be the molecular weight, the number of rotatable bonds and others [25] or extended-connectivity fingerprints (ECFPs) [65]. ECFPs are explained on Section 2.5. Such features are compatible with most traditional machine learning algorithms. Another way to represent a chemical structure is graphs [19], using graph-based models, such as graph variational autoencoders [42] or graph convolutional neural networks [19]. A third approach is to represent molecules in a linear notation, more specifically as a string of characters. Simplified molecular-input line-entry system (SMILES) [85] is a line notation for representing the structure of chemical species, a more detailed overview of SMILES is given in Section 2.1. Using SMILES it is possible to

take advantage of the natural language processing (NLP) models to address cheminformatics problems.

In this thesis some technical terms will be mentioned repeatedly, thus at this point they will be briefly explained and for more details, someone could look to the references that are cited. In bioactivity prediction content, a protein array consists of multiple proteins immobilized on a solid support [12] and is used to identify the protein-molecule interactions. Each family of protein target arrays is represented by a gene symbol, for instance OPRD1, KCNH2, HTR7, and each gene symbol corresponds to a regression task. Moreover, whenever the term task is mentioned, it refers to a regression task. Each molecule is a chemical compound that consists of atoms. Each gene symbol is a regression task and has many molecules with their activity values. Regarding the physical chemistry properties content, each property is a regression task, with the molecules as inputs and the property values as output.

1.2 Literature review

In an early study of Svetnik V. et al. [78], they tackled different quantitative structure-activity relationship (QSAR) tasks to predict the biological activity of a molecular structure. They address four QSAR tasks as classification (blood-brain barrier, estrogen receptor binding, P-glycoprotein transport activity, and multidrug resistance reversal activity) and two as regression problems (dopamine receptor binding affinity, COX-2 inhibition). As attributes, several descriptors are created for each of the tasks. The machine learning models that the authors chose are decision trees, random forest (RF), partial least squares, linear regression, support vector machines (SVM), and artificial neural networks (ANN). After comparisons, the random forest typically ranks among the highest performance models, concluding that "off-the-shelf" (without an extensive hyperparameter tuning) is a suitable choice. In the paper "Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships" [47], the authors selected 15 QSAR regression datasets to explore the performance of deep neural networks and random forest, using a number of descriptors to represent a molecule. The DNNs have a large number of adjusted hyperparameters and they trained over 50 DNNs with different hyperparameters. Their results have shown that in 11 out of 15 datasets DNNs on average outperform RF. These findings gave the incentive to explore more the deep neural network architectures on QSAR tasks.

The QSAR task of toxicity in DeepTox paper [50] is approached using a multi-label deep neural network by introducing a multi-task model for QSAR modeling. The dataset that was used is Tox21 to predict the toxicity of chemical compounds on the 12 toxic effects. The input consists of ECFPs descriptors and the output of the network has twelve neurons of the binary classification, corresponding to the twelve toxic effects. Since the twelve different tasks are correlated, the multi-task training could improve performance. The network combined the information from different tasks in its weights, taking advantage of the additional knowledge from all toxic effects which is not possible when tackling each task individually. The results have shown that it achieved competitive results and outperform the traditional machine learning methods in 10 out of 12 assays, showing promising results utilizing multi-task learning.

The work of Sturm Noé et al. [76], uses the ExCAPE-ML with 526 targets to predict the binary classification of molecules' bioactivity in presence of a target protein, then they test on an external in-house test set to evaluating on industry-oriented compounds. The activities were assigned into two classes (i.e. inactive, active) by setting a threshold on activity values (recorded as pXC50) where pXC50>=6 indicates that the compound-target is active. All models used ECFP descriptors to represent compounds. More specifically, a DNN trained on multi-task binary prediction on all target-proteins, the gradient boost model XGBoost were trained on each protein individually for binary classification, and a Bayesian matrix factorization; approaching the tasks as regression and then considering a compound as active if it had pXC50>=6. The conclusion of this work show that the DNN outperforms the other

two models on most of the tasks but not all, confirming the previous literature observations that deep learning can be useful in multi-task learning.

Li X.and Fourches D. with the "MolPMoFiT" [45] model approached the quantitative structure activity/property relationship QSAR/QSPR tasks like natural language processing (NLP) problem. As input, the SMILES string of characters representation was used on a recurrent neural network architecture, called ULMFiT [36] model. Moreover, transfer learning was utilized where they trained the language model (LM) to predict the next character given the previous characters. In this way the model understands the syntax of SMILES strings and then fine-tune these parameters on a single task at a time. They tested the performance of ULMFiT model on four benchmark datasets of molecular activities/properties and shown that it performed better than the state-of-the-art results reported in the literature for all four benchmark datasets. Their findings encourage the potential of NLP approach for QSAR problems.

In the recent work "Molecular representation learning with language models and domain-relevant auxiliary tasks" [26], the authors used the state of the art neural network in NLP the Transformer called Bidirectional Encoder Representations from Transformers (BERT) [24] on QSAR tasks. The main idea is as described previously, pre-train the BERT model in a self-supervised way and then fine-tune it on downstream tasks as QSPR, where they achieved remarkable results on QSPR benchmarks, indicating the potentials of Transformer architecture on these problems.

1.3 Objectives and Research Questions

This thesis aims to experiment with a novel multi-task regression approach to predict the biological activity and physicochemical properties of a molecular structure. QSAR/QSPR prediction is one of the key elements of early drug development. Moreover, effectively predicting the activity of chemical compounds could help prioritize the experiments during the drug discovery process, thus could reduce the time on conductive experiments.

The main research questions this thesis aims to answer:

1. Can a text-based model, with the same parameters and architecture, predict multiple molecular activities/properties and if yes, to which extent?

2. Can transfer learning improve the performance of the text-based model, by giving prior knowledge on the model about the syntax of chemical compounds?

3. How does the text-based model perform compare to a feature-based traditional machine learning algorithm?

The following chapter Data2 will present the datasets that are used in this thesis.

2 Data

The bioactivity data that is used in this work is from the publicly available database ExCAPE [77], containing SMILES representations of molecules and their activity on a target protein. The benchmark dataset webpage MoleculeNet [87] contain physicochemical properties with their molecules as SMILES strings.

2.1 Simplified Molecular Input Line Entry System

Simplified Molecular Input Line Entry System (SMILES) [86] is a way to represent chemical compounds as text on principles of molecular graph theory. SMILES is a proper language, with a simple vocabulary (atom and bond symbols) and only a few grammar rules. Atoms are denoted by their atomic symbol and are enclosed in square brackets, except of the elements in the "organic subset" 'B, C, N, O, P, S, F, Cl, Br, and I' which may be written without brackets in some cases. For instance, phoshine in chemistry is written as PH3 and P as SMILES. It shows that hydrogen is omitted when the atom is not in square brackets while in the case of the atom in square brackets it is not, e.g. hydroxyl anion [OH-]. The lower case letters specify the atoms in aromatic rings, for example aliphatic carbon is represented by the capital letter C while aromatic carbon by lower case c. The bonds in SMILES are denoted with special symbols -, =, #, and :, represent single, double, triple, and aromatic bonds, where single and aromatic bonds are usually omitted. For instance, "CC" is ethane (CH3CH3), "O=CO" is formic acid (HCOOH). Branches in SMILES representation are encapsulated in parenthesis, e.g. isobutyric acid is written as "CC(C)C(=O)O". In cyclic structures one single (or aromatic) bond should be broken to have a non-cyclic graph and be written as line notation. The disconnected compounds are denoted as separate structures that are concatenated by a period, e.g. sodium phenoxide is written as "[Na+].[O-]c1ccccc1".

In Figure 2.1 we can see how the chemical compound Tryptophan is converted from a molecular graph to line notation of SMILES. First all cycles are broken and replaced by matching digits, then some traversals are selected and the branches are colored to indicate an order to be written. Finally following the main backbone the SMILES string is constructed, where the beginning and end of each traversal are denoted as open and close parenthesis. A molecular graph can be read starting from different edges or following different trajectories.

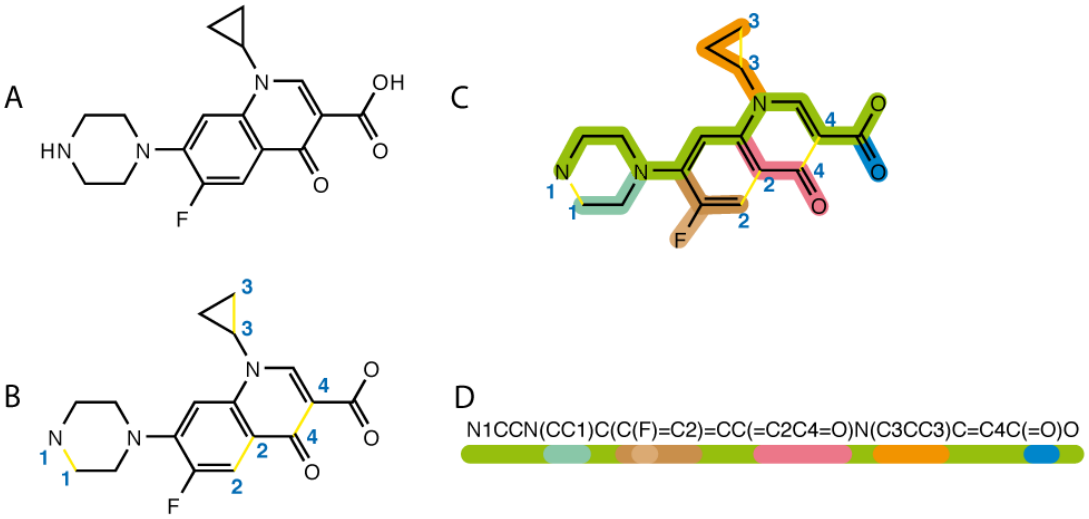


Figure 2.1: Transition from graph to line notation using SMILES generation algorithm. The molecular graph (A). All cyclic structures are broken and matching digits are written indicating their connection, constructing a spanning tree (B). Coloring the traversals to create paths (C). Reading the graph following the colors (D).
source: Original by Fdardel, slight edit by DMacks, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=2556784

2.2 Exascale Compound Activity Prediction Engine database

Exascale Compound Activity Prediction Engine abbreviated as ExCAPE is a publicly available database combining active and inactive compounds from two large open databases PubChem [39] and ChEMBL [51]. It contains over 70 million data points with about one million compounds and 1667 targets. Each data point consists of the activity of a chemical compound on an array of protein target, including the standardized compound structures, the official gene symbols and the standardized, log-transformed activity values (pXC50 values). This thesis is focused on predicting the activity values of around 310000 molecules and 133 protein targets that were of interest to investigate. In each protein target at least 1200 molecules are included in order to have comparable results for single task predictions. Each task is represented by the gene symbol or protein target, so every task has its chemical compounds (data points) with their activity values (target values) on that specific task.

Table 2.1 shows ten of the one hundred and thirty tasks denoted by the gene symbol, with the number of chemical compounds that they have, and the corresponding range and median of the activity values in pXC50. The gene symbol with the most and the least entries are included in this table. The regression task of 'OPRM1' gene symbol contains the most molecules 5830, and the 'GHSR' gene symbol the least, 1241 molecules. All pXc50 values has minimum value 5 and the upper bound differs between genes, as well as the median differ among tasks. The full table with all 133 gene symbols can be found in the Appendix.

The divergence between the target values of different tasks is noticed in Figure2.2, where x-axis denotes the different gene symbols and y-axis denotes the pXC50 values. Some tasks (GSK3B, FGFR1, AKT2) have the pXC50 values concentrated around 5 and are a bit skewed on the high values of pXC50, while other tasks (HSD11B1, P2RX7, HTR7) have median around 7 and high variance of the activity values. The TACR1 gene symbol is spread from pXC50 value 6 to 10, and has some extreme values over 11 than the other displayed tasks do not have. Thus, it is shown that some tasks have different distribution of the target values, while others have similar.

Gene Symbol	Count	pXC50 range	pXC50 median
OPRM1	5830	[5.0, 13.38]	7.30
KCNH2	5275	[5.0, 9.85]	5.63
HRH3	4662	[5.0, 11.22]	7.89
ADORA3	3810	[5.0, 15.0]	6.91
CA2	3652	[5.0, 10.0]	7.44
HPGD	3061	[5.0, 8.25]	5.35
SIGMAR1	2886	[5.0, 11.6]	7.48
ROCH1	1629	[5.0, 9.55]	5.60
PRKCD	1525	[5.0, 10.0]	5.60
GHSR	1241	[5.0, 11.0]	7.56

Table 2.1: A random sample of ten gene symbols with the number of chemical compounds that are associate with, the range and median of their activity values.

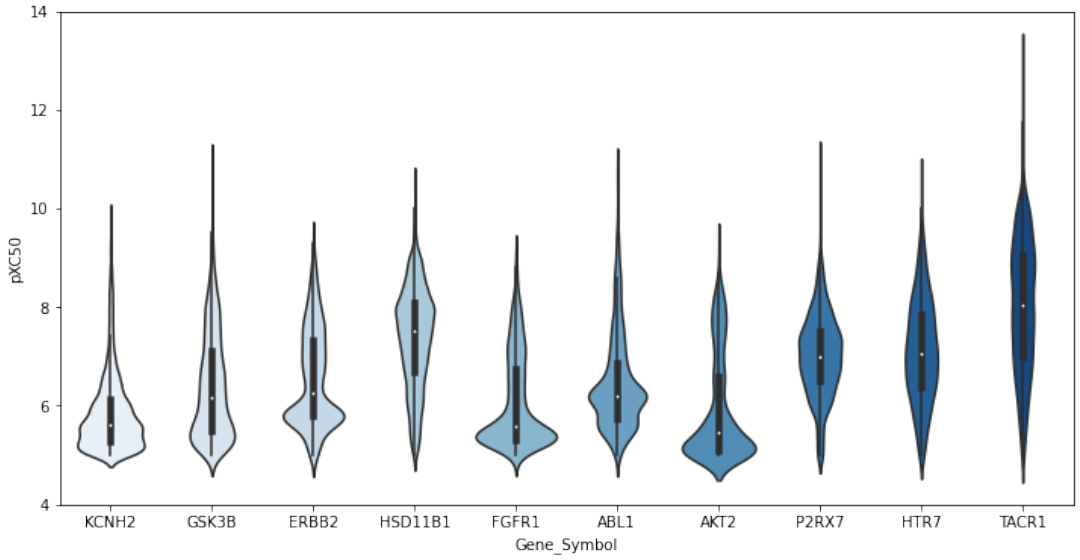


Figure 2.2: The violin plots show the divergence of the activity value distributions from a random sample of ten different tasks.

A sample of the whole dataset is shown in Table2.2, where all tasks are concatenated in one dataset for the multi-task approach. The first column indicate from which database is the information of the entry extracted, the second column shows the task (gene symbol). The third column is the SMILES stings that represent the molecules and the last column is the activity values that will be predicted by some model.

2.3 MoleculeNet benchmark datasets

From the publicly available benchmark datasets on webpage MoleculeNet [87], three physical chemistry property datasets were downloaded; ESOL, FreeSolv and Lipophilicity, with number of molecules 1128, 642, 4200 respectively. Estimating Aqueous Solubility (ESOL) [23] is the property that measueres how much a molecule disolves in water. Free Solvation (FreeSolv) [52] dataset contains the experimental hydration free energy of small molecules in water. Lipophilicity is an important molecular property for the drug discovery process, it shows how easy a molecule disolves in oil, fat, and it is provided as experimental results of octanol/water

Database Entry ID	Gene Symbol	SMILES	pXC50
PUBCHEM53320866	HTR7	ClC1=CC=C(CN2C3=C(CN...4)C=C1	6.12
CHEMBL553	CDK2	C=12C(=NC=NC1C=C(C(=...)C=CC3	5.1
CHEMBL2372200	OPRD1	N1[C@H](C(NCCCC[C@@H...=CC=C3	7.35
PUBCHEM73353231	OPRD1	O=C1NCC(=O)N[C@@H](C...O)C=C3	7.47
CHEMBL1408587	HPGD	O1C2=C(C(CC1=O)=COC(...C2)C)C	5.6
CHEMBL1080117	KCNH2	C1C2(C(CN1CCCSC3=NN=...=C6)Cl	6.7
CHEMBL1642761	OPRD1	C1CCN(C=2C=CC=CC12)C...CCCCC4	5.4
CHEMBL19876	MMP13	C=1(N(C=C(N1)C(O)NO)...C=C2)O	6.67
CHEMBL410234	KCNH2	C1(N2N=C(C=C2C)C)=CC...C)C)=O	6.19
PUBCHEM53388960	GNRHR	S1C=2N(CC3=C(F)C=CC=...=CC=C6	10.15

Table 2.2: A sample of the mixed gene symbols dataset with the molecular representation as SMILES string and the activity value as pXC50 value. Note: the SMILES strings are truncated in order to fit in the table.

distribution coefficient (logD at pH 7.4) of each compound. In all datasets the molecules are represented as SMILES strings. As seen in Figure2.3 the psysical chemistry property values are from totally different distributions and value ranges. This would lead to problems in backpropagating the gradients in the multi-task models4.2, thus each task was scaled by subtracting the mean and dividing by the standard deviation from each task's train set.

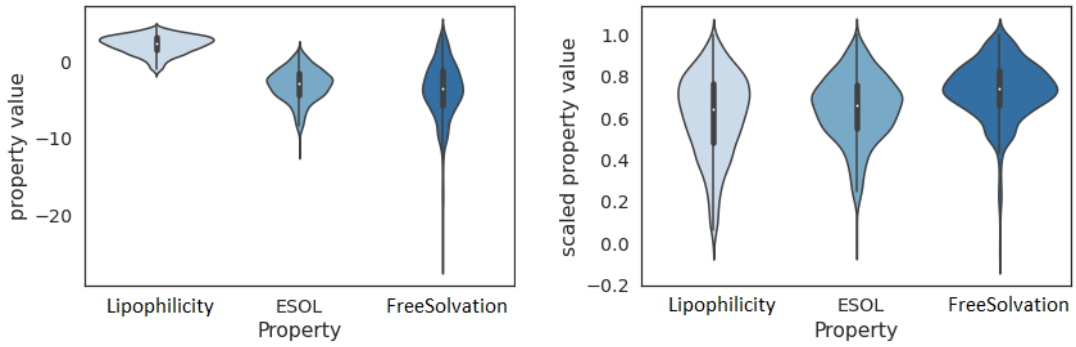


Figure 2.3: The distributions of the three physicochemical properties (Lipophilicity, ESOL, FreeSolvation) on the left and the scaled property values on right.

The length of SMILES strings in each property, which is measured by the number of characters that each string has, varies as shown in Figure 2.4. Lipophilicity has median of SMILES lengths around 50 characters which is more than twice the median of ESOL and FreeSolvation, around 20 characters. All three properties have upper thin tails with max length values 267 for Lipophilicity, 98 for ESOL and 82 for FreeSolvation.

2.4 Preprocessing for text-based models

As shown in Section 2.1, SMILES strings is a textual representation of molecules. These SMILES strings must be transformed into numerical representations, in order to be processed by machine learning models. This section will explain how SMILES string are processed to feed in a text-base model using tokenization and in a feature-base model using fingerprints (i.e. molecular features). The datasets are split into train and test sets. From each task 25% of the data kept for test set and the other 75% for training. In both approaches, text-base and feature-base, the split train and test sets contain the same observation, in that way we can

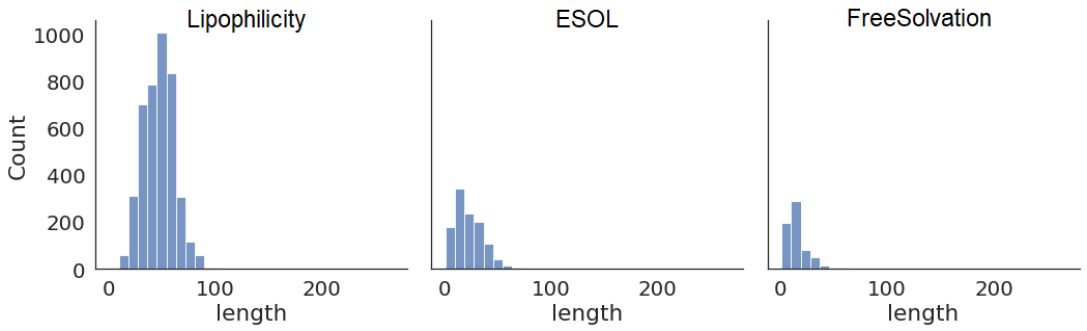


Figure 2.4: The histograms of SMILES strings' lengths in Lipophilicity, ESOL, FreeSolvation datasets. All three properties have upper thin tails with max length values 267 for Lipophilicity, 98 for ESOL and 82 for FreeSolvation.

compare the two approaches. Finally, for the text base approach 10% of the train set kept for validation set and for feature-base approach 5-fold cross-validation is used.

2.4.1 Tokenization

Tokenization is an early step in most natural language processing models. Tokenization technique divides the input text into characters or chunks of characters, called tokens. For example, in English language one of the most common way to tokenize a text is by words, where each word is a token [84]. Similarly in cheminformatics, SMILES strings should be divided into tokens, there exist different ways to tokenize SMILES strings. Using atom-level tokenization, each SMILES string is divided into characters with some exceptions. The atoms with two characters are extracted as tokens such as 'Cl', and 'Br'. Special cases encoded in brackets construct a token, for instance '[Na-]' is a token. For example, D-alanine SMILES representation 'N[C@H](C)C(=O)O' is tokenized as 'N','[C@H]','(','C',')','C','(','=','O',')','O'. All unique tokens of the training set construct the 'vocabulary' of the dataset. In this vocabulary some other special tokens are added, such as the start of SMILES token [SOS], the end of SMILES [EOS], the padding token [PAD] which pads the sequences in each mini-batch (mini-batch is defined in the end of Section 3.1) to have the same length, and the unknown token [UNK] which represents a token that does not exist in the train set but it appears in the test set. Additionally, each gene symbol will be represented as a token, meaning that for 133 gene symbols there exist 133 unique tokens, and the same for the three physical chemistry properties there exist three tokens. Finally, every token of the vocabulary corresponds to an index (integer number), so each input sequence can be transformed from text representation to numerical representation. After the input sequence is transformed into sequence of integers then it can be fed into the Embedding layer which is described in Section 3.2.1. In this thesis, the atom-level tokenization of SMILES will be used, following the literature [70, 46], which is the most common way to tokenize SMILES strings because it preserves valuable information about the molecules.

2.4.2 Data augmentation

Deep learning models need a lot of data to show their predictive power, thus in different applications data augmentation techniques have been developed [27, 37]. Esben Bjerrum introduced a technique to augment SMILES srings[9]. Each molecule can be represented as SMILES string as shown in Section 2.1, where having different starting atom from the same molecule can lead different SMILES representation [5].

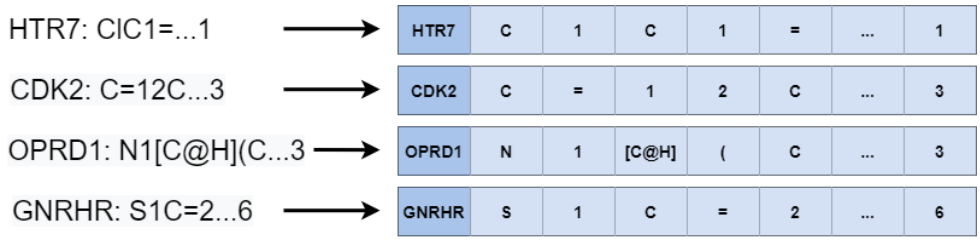


Figure 2.5: The sequence input to tokens. on the left side are the input sequences, i.e. the concatenation of the gene symbol and the SMILES string and in the blue boxes is the sequence tokenized.

SMILES augmentation in this thesis is performed using the open-source package for cheminformatics in python called RDKit [43], version 2020.09.1[1]. Data augmentation is possible only in the text-base models where it is performed during the training, meaning that in each batch the SMILES are augmented on the fly. Using this approach the model receives different SMILES stings for the same molecule and makes it generalize better.

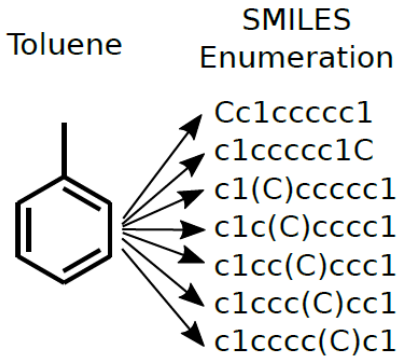


Figure 2.6: The illustration of data augmentation. The molecule toluene represented as different SMILES strings having different starting points of the 2D representation.
Reprinted with permission: Esben Jannik Bjerrum, SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules, arXiv, 2017 [9]

2.5 Molecular Fingerprints

In cheminformatics exists different ways to create molecular fingerprints (i.e. features) for representing a molecule. Following the models' performance on different molecular representation from literature [63, 22]; the most commonly used is the extended-connectivity fingerprints (ECFPs) [65]. ECFPs are binary arrays that encode physiochemical and structural properties of molecules. ECFPs represent molecular structures by means of circular atom neighborhoods, getting into account only the information in a specific radius of each atom. They can quickly be calculated from SMILES strings using the RDKit [43] python package. The radius and the number of bits can be specified when calculating fingerprints from SMILES strings.

In this work, radius=2 and bits number=2048 is used, which has shown competitive results in the literature with SVR [64].

[1] https://rdkit.org/

3 Theory

This chapter introduces the different theoretical frameworks that are used in this thesis to tackle the research questions. First, all the pieces to implement a Transformer Neural Network will be explained. Then the Support Vector Regression model and the 'kernel trick' are explained. Finally, Bayesian Hyperparameter Optimization is introduced, which is an efficient way to find optimal hyperparameters for a machine learning model.

The mathematical notation that will be used throughout this thesis will denote with lower case letters the scalar or scalar-valued functions, and with **bold** lower case letters the vectors or vector-valued functions. While the matrices or higher order matrices (called tensors in programming frameworks) with be denoted as capital letters.

3.1 Feedforward Neural Network

The idea of artificial neural networks is highly inspired by the human brain. Imagine a situation that a person is thirsty, the human feels the sense of being thirsty, so this input goes to the

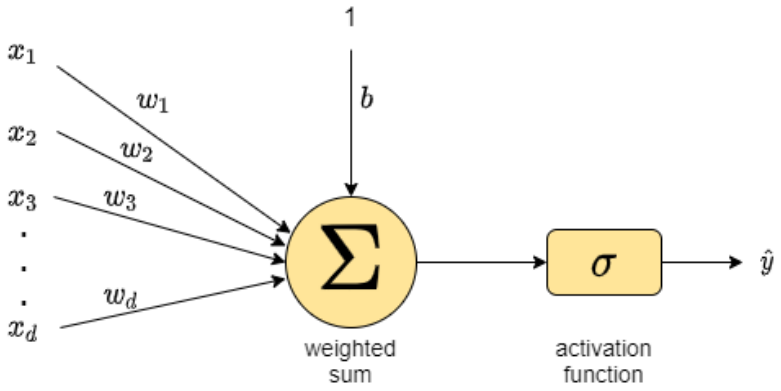


Figure 3.1: One neuron with a non-linearity, the input is **x** of d dimensions, the weights are **w** in the same dimensions as **x** and b is the bias. The weighted sum of the input plus the bias is passed through the activation function σ to get the output.

brain to be processed. Afterwards the brain takes a decision to act, this action could be to drink a glass of water. The decision of the brain to act is the output, this is a highly simplified procedure of a neuron. The same way artificial neuron works, the model with one neuron is called perceptron [66]. The one neuron model is illustrated in Figure 3.1, where the input is denoted by $\mathbf{x}$ of d dimensions, the weights $\mathbf{w}$ and the bias b. The weighted sum of the input plus the bias is passed through the activation function σ to get the output. The activation function is a non-linear transformation and in perceptron is the step function. Generally the activation function is related to the outcome that is need, for example if task is a binary classification, the output must be from zero to one then the sigmoid activation function is suitable. The perceptron equation for one data point looks like $\hat{y} = \sigma(\mathbf{x}^T\mathbf{w} + b)$, where $\mathbf{x}$ and $\mathbf{w}$ are vectors of d dimension, b is a scalar and σ is the step function (which gives 1 if the input is greater than zero, and zero otherwise).

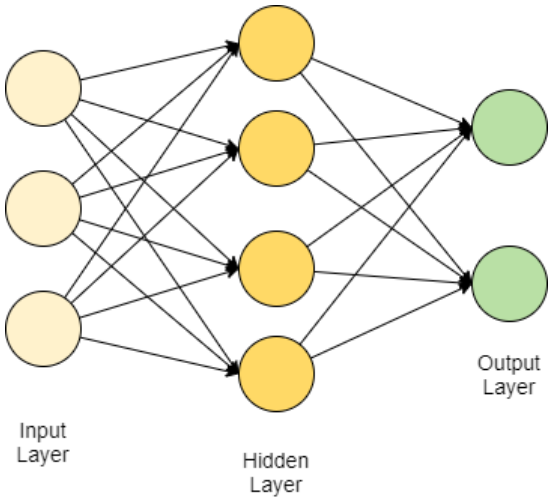


Figure 3.2: A Feedforward Neural Network with one hidden layer. It consists of three input units, four neurons in the hidden layer and two output units.

Combining multiple perceptron in a network resulting the Multi-Layer Perceptron (MLP), also called Feedforward Neural Network (FNN). It is called feedforward because information flows forward to the network and there is no connection that fed back into itself. Usually, the FNN that has more than one hidden layer is called Deep Neural Network (DNN) which are commonly used nowadays. The Figure 3.2 shows a FNN with the input layer of three dimensions, one hidden layer of four dimensions and the output layer of two dimensions. The used mathematical notations is in vectorized form, denoting the input X with $n \times d$ dimensions, where n is the number of observation points and d the dimension of each point (i.e. the number of features). The *forward propagation* is calculated with the following steps:

$$\mathbf{h}(X) = \sigma(XW_1 + \mathbf{b_1}) \tag{3.1}$$

and

$$\mathbf{y}(\mathbf{h}) = \sigma(\mathbf{h}^T W_2 + \mathbf{b_2}) \tag{3.2}$$

where W_1, b_1 and W_2, b_2 are the weights and biases of input to hidden and hidden to output layers respectively. The weights and biases are called parameters of the model and are usually initialized in the beginning by random values close to zero. Regarding the weight initialization, there exists careful considerations about the activations to get a proper flow of the signal and ensure that there is no diminishing or exploding gradients during backpropagation, two common initialization's schemes are Xavier [29] and He [34] initializations. The activation

functions denoted by σ could be any activation function such as Sigmoid, Hyperbolic tangent (tanh), Rectified Linear Unit (ReLU) [53] and others.

$$sigmoid(x) = \frac{1}{1+e^{-x}}, \quad tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}, \quad ReLU(x) = max(0,x) \tag{3.3}$$

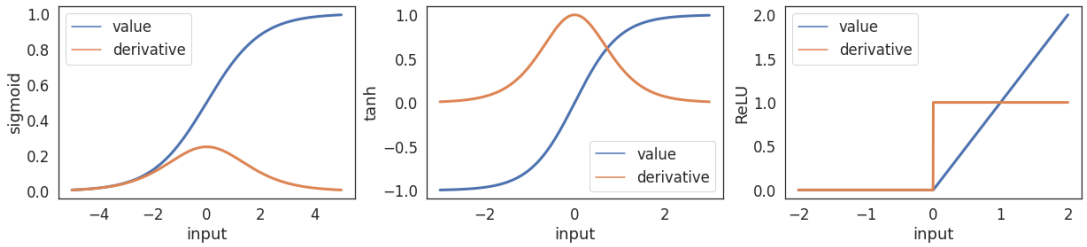


Figure 3.3: The activation functions sigmoid, tanh and ReLU along with their derivatives. Note: all three functions have domain all the real values.

The FNN are powerful models in which usually is needed to introduce some regularization in order to avoid overfitting, the most common and effective way to do that is by using dropout [74]. The dropout layer is usually used after the activation function is applied and it drops each connection of a layer with probability p (which is a hyperparameter), and the output of the neuron is also adjusted by dividing with (1-p) during training to ensure more or less same "signal" after the dropout is turned off, in that way the model does not memorize to activate specific neurons. The network will learn not to rely in particular connections too heavily but to consider more connections.

After the end of the forward pass, the loss is calculated and using *backpropagation* [68] the weights of the neuron are updated.

The loss function depends on the type of problem that is addressed, for example classification, regression and others. Denoting the output of the network $\hat{\mathbf{y}}$ and the actual values $\mathbf{y}$, the loss function is denoted as $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$. Using the *backpropagation* algorithm the gradients of the loss with respect to each layer parameters is computed, propagating through the layers of the network backwards. With a gradient descent algorithm [67] and using a learning rate η the weights of the output layer from the previous example are updated as follows:

$$\begin{aligned} W_2^{new} &= W_2 - \eta \frac{\partial \mathcal{L}}{\partial W_2}, \\ \mathbf{b_2}^{new} &= \mathbf{b_2} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b_2}} \end{aligned} \tag{3.4}$$

where the learning rate η denotes the size of the step that each update will have. Then the other parameters W_1 and $\mathbf{b_1}$ are updated going backwards. There are different variations of the gradient descent algorithms [67], using one observation at a time to update the gradients is called *stochastic gradient descent*, and using all observation is called *batch gradient descent*. Usually a variation in-between is used, the *mini-batch gradient descent*, where the data are split into batches and the parameters are updated after each mini-batch step. One epoch of the training is when all data points (i.e. all mini-batches) are used to update the parameters of the network. For training a neural network a number of epochs is needed (depending on the problem) to update the parameters and find the optimal ones. One of the most widely used gradient descent optimization algorithms is Adaptive Moment Estimation (Adam) [40] which has shown robust performance on large datasets and on non-convex optimization problems. The key elements of Adam is the adaptive learning rates for different parameters in the network, the exponentially decaying average of past gradients $\mathbf{m_t}$ and the exponentially decaying average of past squared gradients $\mathbf{v_t}$, which are estimated of the first ($\mathbf{m_t}$) and second ($\mathbf{v_t}$)

statistical moments of the gradients, i.e. the mean and the variance. The $\beta_1, \beta_2 \in [0, 1)$ are hyperparameters that control the exponential weight decay rates of $\mathbf{m_t}$ and $\mathbf{v_t}$.

$$\begin{aligned} \hat{\mathbf{m}}_\mathbf{t} &= \frac{\mathbf{m_t}}{1 - \beta_1^t} \\ \hat{\mathbf{v}}_\mathbf{t} &= \frac{\mathbf{v_t}}{1 - \beta_2^t} \end{aligned} \tag{3.5}$$

where $\hat{\mathbf{m}}_\mathbf{t}$ and $\hat{\mathbf{v}}_\mathbf{t}$ are the biased-corrected first and second moment estimates.
So the update rule for the Adam optimizer is:

$$\mathbf{w_t} = \mathbf{w_{t-1}} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_\mathbf{t}} + \epsilon} \hat{\mathbf{m}}_\mathbf{t} \tag{3.6}$$

where $\mathbf{w_t}$ is the weight vector at time step t. The suggested values of the hyperparameters from the authors of Adam are: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

3.2 Transformer Neural Network

The Transformer architecture was proposed in the paper "Attention is All You Need" by Vaswani et al. [79] in 2017. The original use case of Transformer was as sequence to sequence model, which was used for the neural machine translation application. Since then it is used for all kind of tasks (having a sequence as input) with different variations of its architecture. For instance, the BERT (Bidirectional Encoder Representations from Transformers) [24] model is used mainly for discriminative tasks, the GPT3 (Generative Pre-trained Transformer 3) [13] is usually used for generative tasks and others.

The big picture of the original Transformer architecture consists of a muber of encoders which receive the input sequence to create meaningful representations of it, and a muber of decoders which generate the output sequence in an autoregressive way (one element of the output sequence at time given the previous sequence positions) as ilustrated in the Figure 3.4.

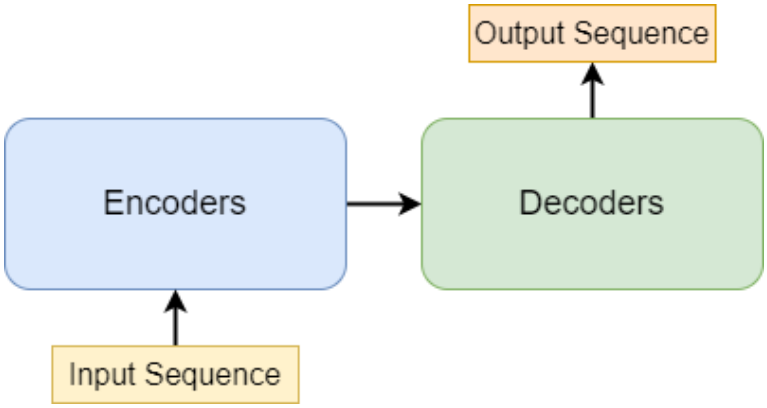


Figure 3.4: The big picture of the original Transformer model [79] which consists of encoder and decoder parts.

3.2.1 Embedding layer and Positional Encoding

The Embedding layer is maping one index to a high dimensional vector. More specifically, all unique tokens of the dataset construct the vocabulary, in natural language context the tokens could be all unique words and symbols of the dataset (see Section 2.4.1). As it is illustrated on Figure 3.5, a numeric index is assigned on each unique token (from blue to green boxes Figure 3.5), so the input text sequence is possible to be represented as a sequence of numeric indixes. The embedding layer maps each index to an d-dimensional vector (from green to yellow boxes

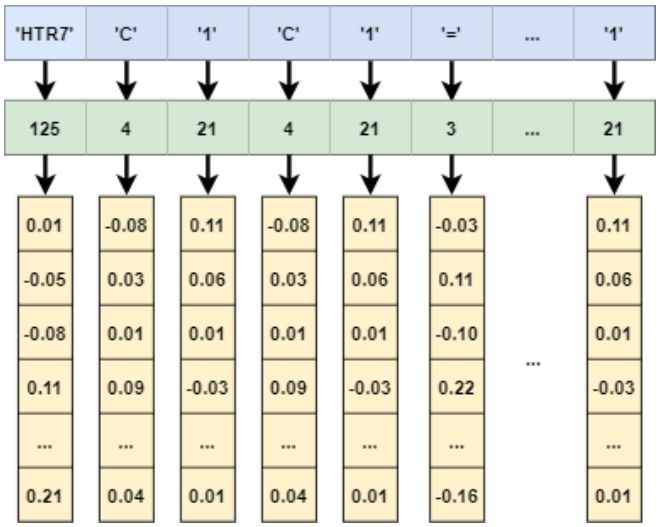


Figure 3.5: The procedure from tokens to embedding vectors. First the input sequence is tokenized (blue boxes), second each unique token is converted into an index of the vocabulary (green boxes), and then each index is mapped into an embedding vector with learnable values (yellow boxes).

Figure 3.5), where usually in literature [24, 60] d=256, 512, 1024. In the beginning of the training phase these d-dimensional vectors are randomly initialized, and they are part of the learnable parameters of the model which are updated in every mini-batch step.

The Recurrent Neural Networks (RNN) [71] receive the input sequence sequentially, one token (input step) at a time, which enables them to capture the order of the input sequence but also makes them relatively slow because of their recurrent nature; they cannot parallelize the computations [79]. The Transformer architecture receives the input sequence all at the same time and perform the computations in parallel, which makes it much faster than RNN but is missing a way to take into account the order in which the input sequence is received. Thus, to address this issue the positional encoding [79] is introduced. It injects in the input sequence the information about relative or absolute position of the tokens. The positional encoding vectors have the same dimension (we denote d_{model} dimensions) as the input embeddings and they are added together to construct the information that corresponds to the input sequence and its order combined. There are different techniques to get the positional embeddings with learnable of fixed values. The original paper [79] introduce it as fixed positional encodings. They use wave frequencies to capture position information, using sine and cosine functions of different frequencies:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{3.7}$$

Where pos is the position of the token in the sequence, i is the specific dimension of the embedding vector and d_{model} is the size of the embedding layer. They take the signals of sine and cosine and interweaves them to construct the positional encoding. Figure 3.6 shows the positional encodings of a sequence with 30 tokens and 256 embedding size, the values of the interweaves are shown as colors. Each row of the figure from top to the bottom, corresponds to the positional encoding that is added to the respective embedding, in order to capture the order of the sequence.

Finally, after the input embeddings are constructed, they pass through a dropout [74] layer to introduce regularization.

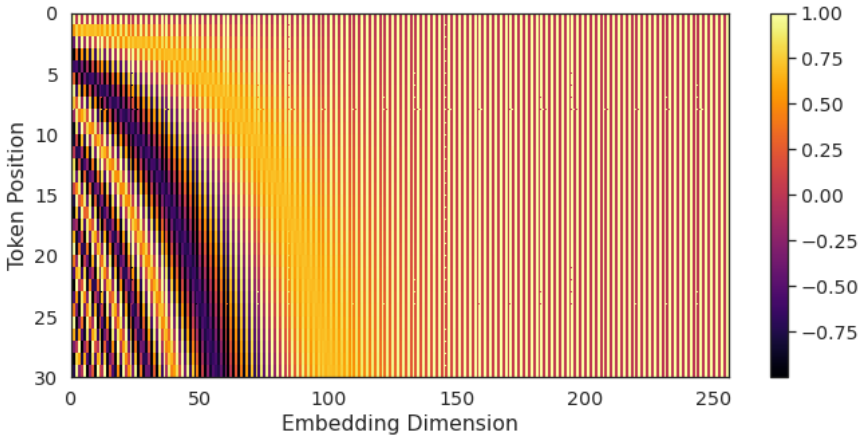


Figure 3.6: The positional encoding as rows, with their values in different color intensity. The x-axis corresponds to the token position and the y-axis the embedding dimension. The color represent the interweaves of sin and cosine.

3.2.2 Attention Mechanism

The Attention Mechanism helps the model to focus on important tokens of the sequence and to relate each individual token to each of the other tokens of the sequence. First step, the matrices of queries Q, keys K and values V need to be calculated, where each matrix is composed of query, key and value vectors, respectively. The names query, key and value are just to distinguish between these three matrices, the names are inspired from the field of databases where for example when someone searches in a database for a file giving a word (query), this is assosiated with the file names of the database (keys) and the best matched files (values) are provided. In the Transformer network content queries, keys and values are representations of each token of the sequence. Notice that for the computations, matrix notation will be used. The input X (sum of embedding vectors and positional encodings) is replicated three times and each gets through a linear layer without activation function (each linear layer has its own weights) and produces the three matrices Q, K and V as follows:

$$Q = W_Q X, \quad K = W_K X, \quad V = W_V X \tag{3.8}$$

where X is the input embeddings, and W_Q, W_K and W_V are the weight matrices.

For now, these three matrices can be thought as abstraction that will be useful for calculating and thinking about attention. Second step is to calculate the *Scaled Dot-Product Attention* which is the dot product between each query and key to determine how related they are. For all queries and keys the dot product is simplified as the matrix multiplication QK^T, then it is scaled by the square root of the d_k-dimensions of the keys. The previous outcome goes through the softmax function (which scales the values to be between zero and one, and to sum up to one), afterwards the softmax output is getting through a dropout layer and then multiplied with the matrix V. The equetion of Scaled Dot-Product Attention is:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.9}$$

All the combinations of how relevant is the token represented by query vector to all other key vectors construct the scaled weighted matrix: $softmax(\frac{QK^T}{\sqrt{d_k}})$ and the value vector is the input information from which specific parts will be highlighted when it is multiplied by the scaled weighted matrix. This Scaled Dot-Product Attention is performed multiple times which gives attention to different aspects of the input sequence by having different weight matrices

W_Q, W_K and W_V. Each Scaled Dot-Product Attention is called attention head, and its output is denoted as $Z_h = Attention(Q_h, K_h, V_h)$ of the attention head h. In the original paper [79], they used $h = 8$ heads. After computing all attention head in parallel, they are concatenated horizontal into one long matrix Z which captures information from all heads. This matrix is fed to a linear layer performing the matrix multiplication $W_Z Z$ and gets the output of the Multi-Head Attention, as illustrated in Figure 3.7.

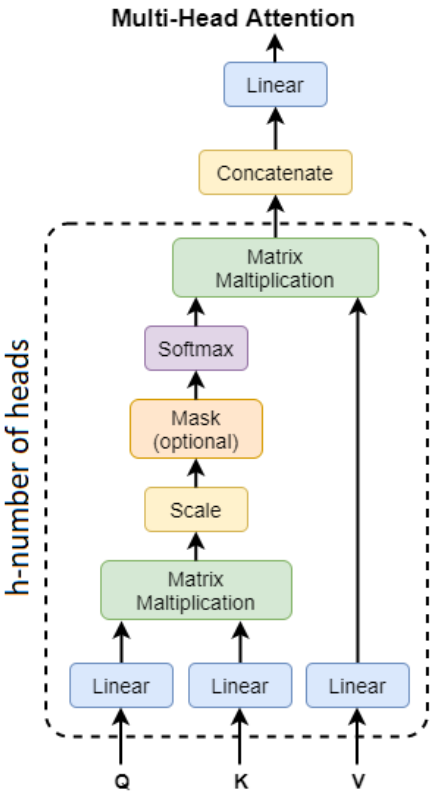


Figure 3.7: The operations of Multi-Head Attention (following Vaswani et al. 2017 [79]).

An additional feature of Scaled Dot-Product is Masking, which is used in the decoder part of the Transformer. Usually the model should not know the context of the next steps of the sequence, thus before the softmax function masking is applied. Masking sets all tokens of the matrix that correspond to next step to minus infinity so after the softmax function, they turn into zero, meaning that the model will give zero attention to the next steps of the position that it currently is.

3.2.3 Transformer model architecture

First, the inputs, outputs and output probabilities that are illustrated in the Figure 3.8 will be explained with an example. Let us assume an example from machine translation of NLP, and assuming that each sequence is split into words. We want to translate from Swedish to English the sentence ”Jag dricker vatten” to ”I drink water”. In the beginning of the translation the input is the sequence ”Jag dricker vatten” while the output (that gets as input on the decoder) is just the start token and the model should predict the word ’I’ which is the chosen from the max of the output probabilities. Then the output would be the start token and the word ’I’, so the network tries to predict the word ’drink’. Finally, the output would be the start token and the words ’I drink’, thus the models should predict ’water’. The prosedure of hidding the future positions of the output sequence is called masking.

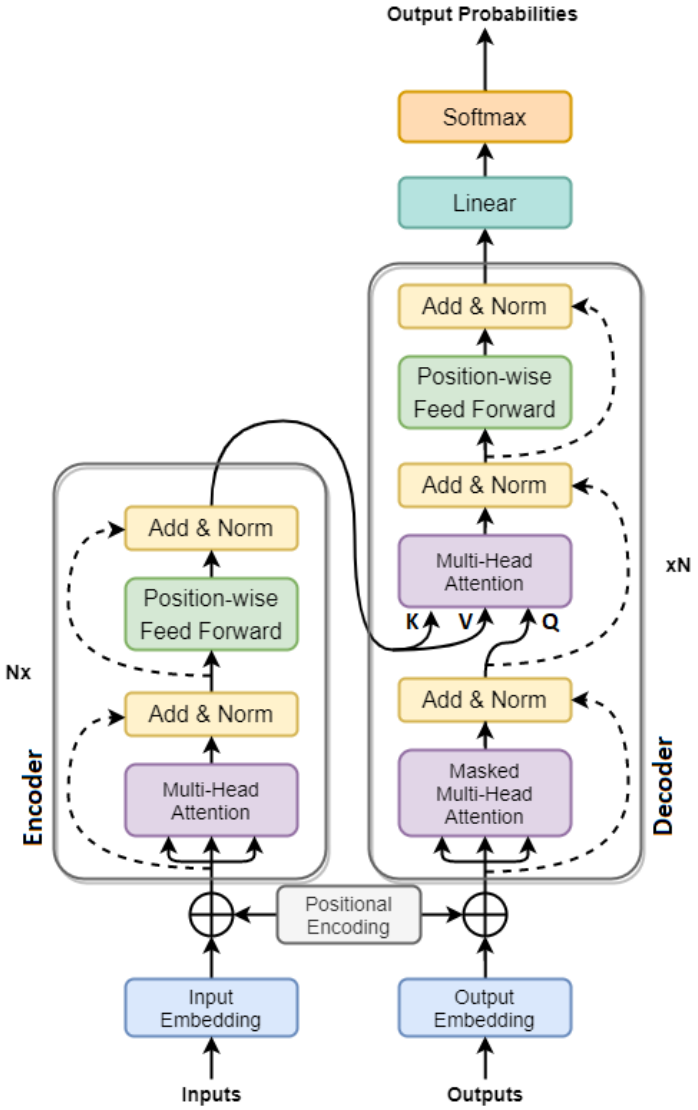


Figure 3.8: The original encoder-decoder Transformer model architecture (following Vaswani et al. 2017 [79]).

An encoder block of the Transformer Neural Network consists of a Multi-Head Attention with h number of attention heads (which is expained in the previous subsection), afterwards a residual connection is used, followed by layer normalization. Residual connection [33] simply means that the input of the Multi-Head Attention is added to its output (i.e. in the first residual connection we have $X + Z$). The residual connection serves two main purposes, knowledge preservation meaning that early information can be preserved through the network no matter how big it is, and vanishing gradient problems in which during backpropagation if the network is large the gradients may get really small and could vanish. Layer normalization [6] normalizes across the features dimension the output of the activation function of a layer to have zero mean and one variance which potentially could reduce the training time.

As mentioned in the paper [79], after the layer normalization each position of the sequence is fed separately into a Feedforward Neural Network by using the same weight matrices (W_1, W_2) and biases (b_1, b_2) for all elements of the sequence, which is called Position-wise Feed-Forward.

Each of the FNN consists of two linear transformations with a ReLU [53] activation in between, passing through a dropout layer in the end (the light green box in Figure 3.8):

$$FNN(\mathbf{x}) = max(0, \mathbf{x}W_1 + \mathbf{b_1})W_2 + \mathbf{b_2} \tag{3.10}$$

After the feedforward network again a residual connection and layer normalization are applied. The above layers that are mentioned construct the encoder block as it is illustrated in a rounded box on the left part of the Figure 3.8. The output and input of an encoder block have the same size. The encoder is composed of N stack identical blocks, the original paper [79] uses N=6, where stack the encoder (or decoder) blocks means that the output of the first encoder (or decoder) block becomes the inputs of the second block and so on.

Masked Multi-Head Attention layer means that the input of this layer is masked until the position of the sequence that the decoder is currently processing. The input to this layer follows the same operations as in Multi-Head Attention but with the future positions of the sequence masked.

A decoder block consists of a Masked Multi-Head Attention layer, then a residual connection is added and then passed through a layer normalization. After that layer a cross Multi-Head Attention layer is added, which has as inputs the output key and value vectors of last encoder block and the query vectors of the decoder's Masked Multi-Head Attention. It is followed by residual connection and layer normalization. Last piece of the decoder block is the feedforward network (equetion 3.10) with a residual connection and a layer normalization. The decoder consists of N identical blocks stacked as explained in the encoder.

Finally, to make predictions the output of the decoder at each step of the sequence has been flatten into a vector and being fed into a linear layer of output size equal to the vocabulary size, afterwards a softmax function is applied to transform the logits into probabilities. The loss function to be minimized, that calculates the error between the target and the predicted output is the cross entropy loss.

3.3 Transfer Learning

Deep learning models require large datasets to achieve high performance which is not always affordable. This motivates the need to explore other directions, one of them is to transfer knowledge from one model to another, which is called transfer learning [89]. In literature exists numerous transfer learning categories, and in this work, we will focus on the Sequential Transfer Learning [4]. Sequential Transfer Learning refers to the process of learning different tasks sequentially. For instance, a model M is trained (we call this pre-trained model) on a task T_0 and we want to transfer the learning to another task T_1, where usually T_0 is a more general task, and T_1 a more specific task which is called downstream task. Additionally, the model M could be modified to adapt to the downstream task T_1.

Following the paper [4], Sequential Transfer Learning can be further divided into subcategories, in this thesis we will use the fine-tuning approach. In fine-tuning, given a pre-trained model M on task T_0 with parameters W, the model M is used on a new task T_1 to learn a new function f that maps the parameters $f(W) = W'$. This fine-tuning approach of transfer learning is chosen for this thesis and it is the most common approach in recent years with Transformer models [24, 44, 60, 13].

3.4 Feature-based model

In this section the Support Vector Regression model and the 'kernel trick' will be explained. Moreover, an efficient method to find the optimal hyperparameters of a machine learning algorithm called Bayesian Optimization is explained.

3.4.1 Support Vector Regression

Support Vector Machines (SVM) were introduced by Vladimir Vapnik [21] and are characterized by the property of sparseness. SVM is called sparse model because it emphasize the predictions only on some specific data points, called support vectors which are crucial for the prediction. SVM model is extended to Support Vector Regression (SVR) [8] for regression problem using the same idea of support vectors.

Let the random variables $\mathbf{x}_n \in IR^d$ and $y_n \in IR$, and the training set $\{(\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_n$ is the input and y_n is the target. In Ridge Regression (linear regression with L2 regularization), the error function to be minimized is given by

$$\frac{1}{2}\sum_{n=1}^{N}(\hat{y}(\mathbf{x}_n) - y_n)^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{3.11}$$

where $\hat{y}_n = \hat{y}(\mathbf{x}_n) = \mathbf{w}^T\mathbf{x}_n + b$, $\mathbf{w}$ is the coefficients (weights) and b is the intercept (bias). In order to preserve the sparsity property of SVM the quadratic error function is replaced by an ϵ – insensitive error function [21]:

$$E_\epsilon(\hat{y}(\mathbf{x}) - y) = \begin{cases} 0, & \text{if } |\hat{y}(\mathbf{x}) - y| < \epsilon; \\ |\hat{y}(\mathbf{x}) - y| - \epsilon, & \text{otherwise} \end{cases} \tag{3.12}$$

The $\epsilon - insensitive$ error function gives zero error if the absolute value of the error is less than ϵ where $\epsilon > 0$. It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.

Therefore to get sparse solution the $\epsilon - insensitive$ regularized error function is minimized:

$$C\sum_{n=1}^{N}E_\epsilon(\hat{y}(\mathbf{x}_n) - y_n) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.13}$$

where C>0 controls the regularization, specifically C is the inverse regularization parameter.

To have a more flexible model and tolerance in the error two slack variables $\xi_n, \hat{\xi}_n \geqslant 0$ for each data point are introduced.

$$\xi_n = \begin{cases} y_n - \hat{y}(\mathbf{x}_n) - \epsilon, & \text{if } y_n > \hat{y}(\mathbf{x}_n) + \epsilon; \\ 0, & \text{otherwise} \end{cases} \tag{3.14}$$

and

$$\hat{\xi}_n = \begin{cases} \hat{y}(\mathbf{x}_n) + \epsilon - y_n, & \text{if } y_n < \hat{y}(\mathbf{x}_n) + \epsilon; \\ 0, & \text{otherwise} \end{cases} \tag{3.15}$$

As illustrated in Figure 3.9, the regression curve is the red solid curve and the ϵ-insensitive "tube" is the yellow shaded area. The data points which are inside the ϵ-tube have $\xi = \hat{\xi} = 0$, the points above the ϵ-tube have $\xi > 0$ and $\hat{\xi} = 0$, and the points below the ϵ-tube have $\hat{\xi} > 0$ and $\xi = 0$. Only the point outside the ϵ-tube contribute to the final cost.

Taking into account the additional constrains the error function can be reformulated as the following optimization problem that can be solved using Lagrange multipliers.

$$\min_{w,b,\xi,\hat{\xi}} C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.16}$$

$$\begin{aligned} \text{subject to } & y_i - \hat{y}(\mathbf{x}_i) \leqslant \epsilon + \xi_i, \\ & \hat{y}(\mathbf{x}_i) - y_i \leqslant \epsilon + \hat{\xi}_i, \\ & \xi_i, \hat{\xi}_i \geqslant 0, i = 1, ..., n \end{aligned} \tag{3.17}$$

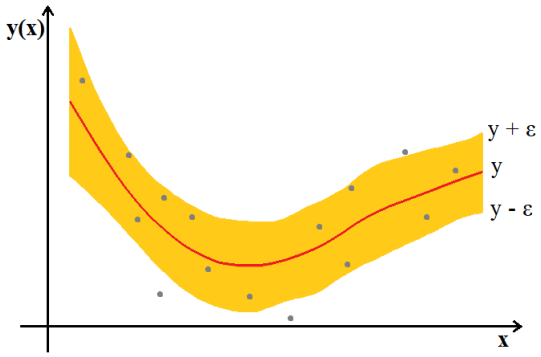


Figure 3.9: The Support Vector Rregression, showing the regression curve in red, the ϵ-tube in yellow and the datapoints as grey dots.

When accurate prediction is not feasible in the input space, the standard scalar product $\langle \cdot, \cdot \rangle$ is replaced by a kernel function $k(\cdot, \cdot)$. A kernel can be though as a similarity function and is defined by $k(x, x') = \phi(x)^T \phi(x')$ where $\phi(\cdot)$ is mapping the input into a higher dimensional feature space, without the need to compute the mapping function $\phi(\cdot)$. This is called the kernel 'trick', by using it the SVR can be adapted to a more powerful non-linear regression model.

The optimization problem 3.16 can be solved by introducing the Lagrange multipliers $\alpha_n \geqslant 0$, $\hat{\alpha}_n \geqslant 0$, $\mu_n \geqslant 0$, $\hat{\mu}_n \geqslant 0$ and optimizing the Lagrangian:

$$\begin{aligned} L =& C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} (\mu_n \xi_n + \hat{\mu}_n \hat{\xi}_n) \\ & - \sum_{n=1}^{N} \alpha_n (\epsilon + \xi_n + \hat{y}_n - y_n) - \sum_{n=1}^{N} \hat{\alpha}_n (\epsilon + \hat{\xi}_n - \hat{y}_n + y_n) \end{aligned} \tag{3.18}$$

Then by substituting $\hat{y}_n = \mathbf{w}^T \mathbf{x}_n + b$ and setting the derivatives of the Lagrangian to zero with respect to $\mathbf{w}$, b, ξ_n, and $\hat{\xi}_n$, we get:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}} &= 0 \Rightarrow \mathbf{w} = \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) \phi(x_n) \\ \frac{\partial L}{\partial b} &= 0 \Rightarrow \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) = 0 \\ \frac{\partial L}{\partial \xi_n} &= 0 \Rightarrow \alpha_n + \mu_n = C \\ \frac{\partial L}{\partial \hat{\xi}_n} &= 0 \Rightarrow \hat{\alpha}_n + \hat{\mu}_n = C \end{aligned} \tag{3.19}$$

where $\phi(\cdot)$ is a function which we do not need to know that maps the input into a higher dimensional feature space, as indroduced above by the kernel 'trick'.

Now by eliminating the corresponding variables from the Lagrangian, the problem can be seen as maximizing the following Lagrangian with respect to $\{\alpha_n\}$ and $\{\hat{\alpha}_n\}$ and by introducing the kernel $k(x, x') = \phi(x)^T \phi(x')$:

$$\begin{aligned} \widetilde{L}(\alpha_n, \hat{\alpha}_n) =& -\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} (\alpha_n - \hat{\alpha}_n)(\alpha_m - \hat{\alpha}_m) k(x_n, x_m) \\ & - \epsilon \sum_{n=1}^{N} (\alpha_n + \hat{\alpha}_n) + \sum_{n=1}^{N} (\alpha_n + \hat{\alpha}_n) y_n \end{aligned} \tag{3.20}$$

where it is again a constrained maximization which require $0 \leqslant \alpha_n \leqslant C$ and $0 \leqslant \hat{\alpha}_n \leqslant C$.

By substituting the first equetion of 3.19 into $\hat{y}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$, the predictions for a new input (x) can be expressed in terms of the kernel function:

$$\hat{y}(x) = \sum_{n=1}^{N} (\alpha_n - \hat{\alpha}_n) k(x, x_n) + b \tag{3.21}$$

There exist many kernels that are used with SVR, for example the polynomial of d-degree $k(x, x') = (x^T x')^d$ and the gaussian radial basis function $k(x, x') = exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$.

3.4.2 Bayesian Hyperparameter Optimization

There exist different ways to find the hyperparameters of a machine learning algorithm. One way is to search manually some specific values, another is to make a grid search on an interval or search randomly values from an interval. These approaches require expert experience, brute-force search, or sometimes luck.

Bayesian optimization [73, 11] uses a powerful strategic search on finding the extrema of an objective function which could be the validation error of a machine learning algorithm. It is called Bayesian because it uses the "Bayes' theorem", simplifying it says that the posterior probability of the model M given the data D is proportional to the likelihood of D given M multiplied by the prior probability of M:

$$P(M|D) \propto P(D|M)P(M) \tag{3.22}$$

The prior represents the belief about the search space for the optimal hyperparameters. Since the objective function (e.g. validation error) of a machine learning algorithm is unknown given different hyperparameter values, a *surrogate model* is used to estimate it under some uncertainty. A common choice of surrogate model is a Gaussian Process(GP) [61], which is a flexible and has the ability to give uncertainty estimates. It is not possible to evaluate the model infinite times randomly because it is expensive, thus a function to choose strategically where to evaluate next will be used. It needs to balance exploring the objective function for the optimal value but also exploit it, focusing on regions of the potential optimum. This decisions are made using an *acquisition function*. Acquisition function is a heuristic to show how desirable is to evaluate a point, based on the previous evaluations and the surrogate model.

The goal of Bayesian optimization is to find the location where the optimum (global minimum or maximum) of a function is. The algorithm is described as follows:

Algorithm 1: Bayesian Optimization

Choose a surrogate model for modeling the objective function f and define its prior.
for $t = 1, 2, ...$ **do**
  Given the set of function evaluations, use Bayes rule to obtain the posterior;
  Use an acquisition function $\alpha(x)$, which is a function of the posterior, to decide the next sample point $x_t = argmax_x \alpha(x)$;
  Add new sampled evaluation to the set of function evaluations;
end

In this thesis for the SVR hyperparameters, as surrogate function the Gaussian process (GP). A GP is specified by its mean function $m(x)$, and covariance function $k(x, x')$. We will use Matern kernel while the noise is assumed to be iid gaussian. Let $r = \|\mathbf{x} - \mathbf{x'}\|$, the Matern kernel is defined as:

$$k_{matern}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} r\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}}{\ell} r\right) \tag{3.23}$$

where the parameter $\nu > 0$ controls the smoothnes, $\ell > 0$ is the length-scale (the distance we have to move in the input space for the function to vary signicantly), $\Gamma(\cdot)$ is the gamma function, and $K_\nu(\cdot)$ is a modified Bessel function [61].

Let $x_{1:n} = \{x_1, ..., x_n\}$ be a set of data points (i.e. for this thesis the hyperparameters of SVR), f be the objective function, and the values of the objective function $f_{1:n} = \{f(x_1), ..., f(x_n)\}$ can be approximated by a multivariate gaussian distribution

$$f_{1:n} \sim N(m(\mathbf{x}_{1:n}), \mathbf{K}) \tag{3.24}$$

where $m(\mathbf{x}_{1:n}) = [m(\mathbf{x}_1), ..., m(\mathbf{x}_n)]^T$, and $\mathbf{K}$ is the $n \times n$ kernel matrix i.e. $[K]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

While the chosen acquisition function is the *expected improvement (EI)*. It is defined as follows:

$$EI(x) = \mathbb{E}[max\{0, f(x) - f(x^+)\}] \tag{3.25}$$

where x^+ is the current optimal set of hyperparameters. The expected improvement for a GP model using integration by parts can be written in closed form solution as:

$$EI(x) = \mathbb{E}[max\{0, f(x) - f(x^+)\}] = \begin{cases} (m(\mathbf{x}) - f(\mathbf{x}^+))\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) = 0 \end{cases} \tag{3.26}$$

where $\sigma(\mathbf{x})$ is the standard deviation, $Z = \frac{m(\mathbf{x}) - f(\mathbf{x}^+)}{\sigma(\mathbf{x})}$, Φ is the cumulative distribution of the standard normal distribution and $\phi(z)$ is its the probability density.

Deriving that EI is high when the uncertainty $\sigma(\mathbf{x})$ around $\mathbf{x}$ is high, or when the posterior of the objective function $m(\mathbf{x})$ is higher that the current optimum value $f(\mathbf{x}^+)$.

In the Figure3.10 are illustrated four steps of the Bayesian optimization procedure from a toy example. In iteration zero (i.e. first plot on top), there is one observation evaluated denoted with black dot, where close to this point the uncertainty is low. The acquisition function with green curve at the right indicate its maximum with blue cross, and the next suggested point to evaluate the objective function is shown with red dot. In iteration one, the point that was indicated by the acquisition function in the previous step is evaluated. The uncertainty of GP is updated, and the acquisition function suggest another point to evaluate and so on. It can be seen from iteration three that the max value of the acquisition function is at the neighbor area of the previous step where the algorithm exploits that area. The acquisition function chose to explore first and then to exploit in a specific space on iteration three, which actually was the optimal choice as it is shown by the true objective function.

3.5 Statistical comparison

This section will introduce the methods that are used to evaluate and compare the performance of the implemented models. All approaches are evaluated on the same unseen test sets to measure their generalization performance.

3.5.1 Root Mean Square Error

The Root Mean Square Error (RMSE) shows standard deviation of the error. The error measures how far the predicted values $\hat{y}_i$ from the true values y_i are ($i = 1, 2, ..., n$), and RMSE measures how spread out these errors are, i.e. measures the expected variation of the errors.

$$RMSE(\hat{\mathbf{y}}, \mathbf{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.27}$$

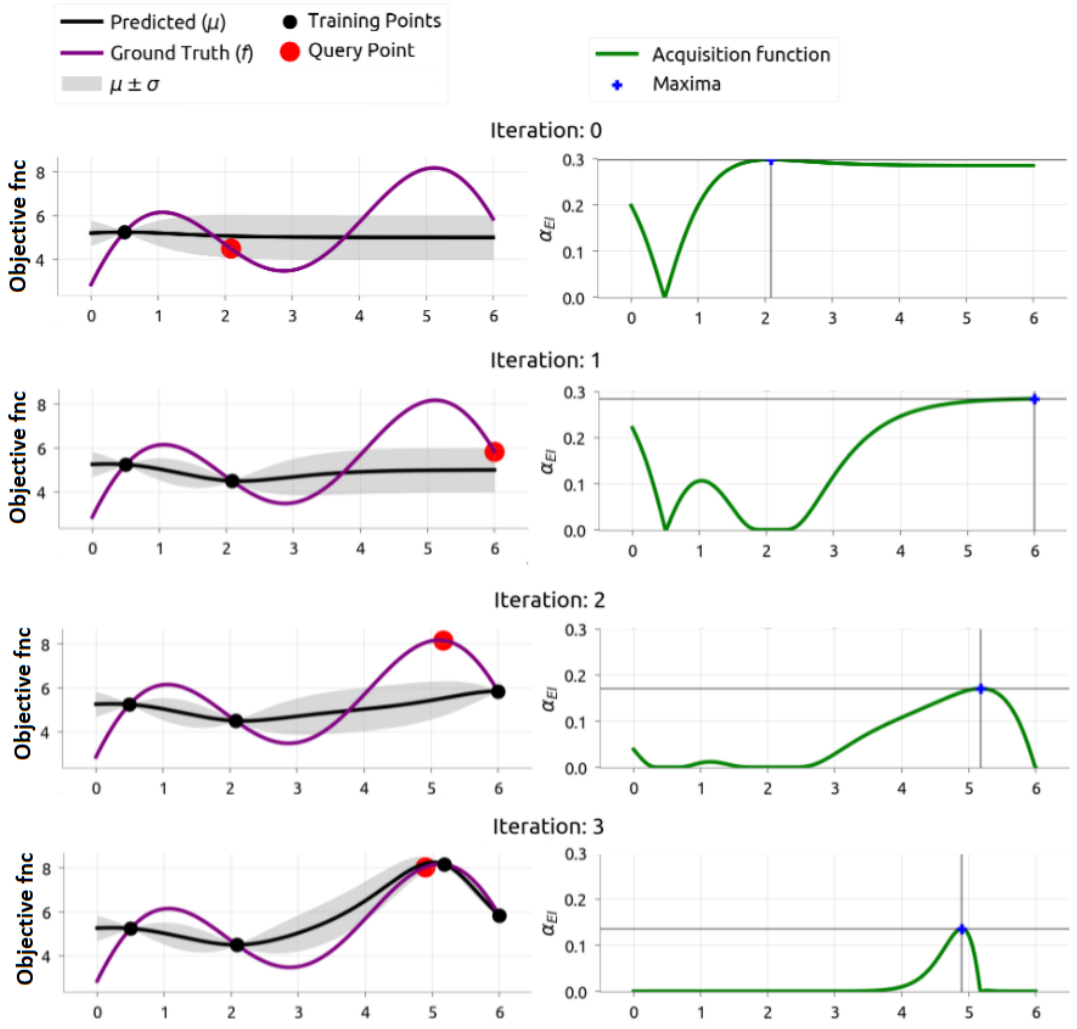


Figure 3.10: The Bayesian Optimization procedure. The unkown objective function is illustrated as purple curve, the mean of GP is the black curve, and its posterior uncertainty is the grey shaded area. The evaluated observations are the black dots. In the plots on the right side, the acquisition function is the green curve and its maximum is denoted with a blue cross. Figure adapted from: Agnihotri A. and Batra N., "Exploring Bayesian Optimization", Distill, 2020, CC-BY 4.0. [2]

3.5.2 Coefficient of Determination

Pearson's [58] correlation coefficient measures how linearly related are two variables x and y. The formula for correlation coefficient is $r_{xy} = \frac{s_{xy}}{s_x s_y}$, where s_{xy} is the covariance. The values that r takes are from -1 to 1, where -1 denotes high negative linear correlation, +1 high positive linear correlation and 0 denotes no correlation.

The Coefficient of Determination (R^2) measures how closely two variables follow a linear relationship, in this case the two variables are the estimated value $\hat{y}$ and the true value y. The R^2 take values from 0 to 1, where higher value indicates that the relationship is more linear.

$$R^2(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.28}$$

where $\bar{y}$ is the mean value of $\mathbf{y}$. On the above equation3.28 the nominator is the variance of the linear fit and the denominator is the variance of y.

3.5.3 Error bars for R^2

To calculate the error bars i.e. confidence intervals of R^2 using the analytic form as described in the paper by Nickolls [54]. First, we need calculate the Pearson correlation coefficient (r). The values of r have range $[-1, 1]$, r cannot assumed to be Gaussian distributed, so they should be transformed somehow to $(-\infty, +\infty)$. The Fisher transform is used to transform the correlation coefficient r approximately to a Gaussian distribution with a standard deviation close to one, making its confidence intervals easy to compute. The Fisher transform is calculated as follows:

$$F(r) = \frac{1}{2} ln \frac{1+r}{1-r} \tag{3.29}$$

The confidence intervals for r are calculated by assuming that F is normaly distributed and by taking the $t-statistic$ of 95% confidence intervals from F values. Then these F values are back-transformed into r values. It is shown by Fisher that the standard error decreases with respect to $\sqrt{N-3}$ rather than $\sqrt{N}$. The back-transform function from F to r-values is:

$$r(F) = \frac{e^{2F}-1}{e^{2F}+1} \tag{3.30}$$

So the procedure is first calculate r, then transform to $F(r)$, add and subtract $\{t - statistic/\sqrt{N-3}\}$ for the confidence intervals of F, afterwards back-transform them to r-values.

3.5.4 Nonparametric hypothesis test

The Wilcoxon Signed-Rank Test is a nonparametric test that tests if two paired samples come from the same distribution [81, 20]. The assumptions of Wilcoxon test are: the observations in each sample can be ranked and are independent and identical distributed, and the observations are paired across each sample.

Let's assume n paired samples of the form (X_i, Y_i), $i = 1, 2, ..., n$. Step 1: the differences $d_i = X_i - Y_i$ are calculated, and the ones that are equal to zero are eliminated and the number of pairs (n) is reduced accordingly. Step 2: the signs of each difference is noted, and the absolute value differences are ranked (writing down the rank number). If two or more absolute differences are tied for the same rank then as rank for the tied differences the average of those ranks is used. Step 3: the sum of the ranks from the negative differences is calculated (denoted as W^-), and the sum of the ranks from the positive differences is calculated (denoted as W^+).

The null hypothesis states that the distribution of of X's and Y's are identical (or it can be rephrased as the medians of X and Y are equal). For a two-tailed test the $W = min(W^+, W^-)$ is used as test statistict to test the null hypothesis. Thus we reject the null hypothesis in favor of the alternative, if W is less than or equal to some value W_0, where W_0 is the critical value. For a one-tailed test if we want to detect that the distribution of X's is shifted to the right of Y's then the W^- is used, and we reject the null hypothesis if $W^- \leqslant W_0$. Similarly if we want to detect that the distribution of Y's is shifted to the right of X's the W^+ is used and we reject the null hypothesis if $W^+ \leqslant W_0$.

H_0: The distributions of X's and Y's are identical.

H_1: (a) The distributions differ in location (two-tailed). (b) The distribution of X's is shifted to the right of Y's (one-tailed).

In this thesis the Wilcoxon Signed-Rank Paired Test is used to examine if the differences in R^2 between two models are statistically significant. One-tailed hypothesis test is performed

to determine whether a model has higher median coefficient of determination from another model with significance level of 0.05. The Wilcoxon signed-rank test is run using the python package SciPy [80] version 1.4.1[1].

[1] https://docs.scipy.org/

4 Method

The main aim of this thesis is to introduce and experiment with a novel architecture to approach multi-task regression prediction with a text-base neural network, inspired by the field of natural language processing, and the paper "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer" [60]. As the recent literature indicates [26, 18, 57, 82], the Transformer Neural Networks are showing competitive results on QSAR/QSPR tasks. Additionally, in the work of Colin Raffel et al. [60], they introduced a Transformer architecture that can handle multiple natural language tasks with the same model. Inspired from the above works on cheminformatics and NLP, this thesis will experiment on predicting multi-task activities and properties with the same Transformer model. The idea is illustrated in Figure 4.1 for the bioactivities tasks, where the input consists of the gene symbol (the name of the task) and the SMILES string (the molecule), while the output is the activity value of the molecule on that specific gene symbol. This approach is different from the existing multitask QSAR prediction of published papers [50, 76] where they used a neural network with multiple output neurons, predicting if a molecule is active or not, on different tasks. They need the targets of all tasks for each molecule to train the network, in contrast this thesis' approach does not require that, by specifying the task in the input sequence. As output of the model that is used in this thesis, is the activity value itself not just if the molecule is active or not, which defines the problem as regression. Moreover, we have the hypothesis that by training simultaneously on all gene symbols or properties could share information between them and improve performance.

4.1 Workflow

For the text-base approach, the molecular structure of each observation from the datasets is represented as SMILES strings (see Section 2.1) which are preprocessed to be fed into the Transformer Neural Network model using tokenization (see Section 2.4.1). Afterwards, the Transformer network is build to perform multi-task QSAR prediction with two approaches, without or with using Transfer Learning. If Transfer Learning is not used, the model is trained with randomly initialed weights of the Embeddings, the Encoder blocks and the Feedforward Neural Network. Otherwise, the weights from the Embeddings and the Encoder are transferred from the pretrained model, and the FNN is randomly initialized, so the Transformer network

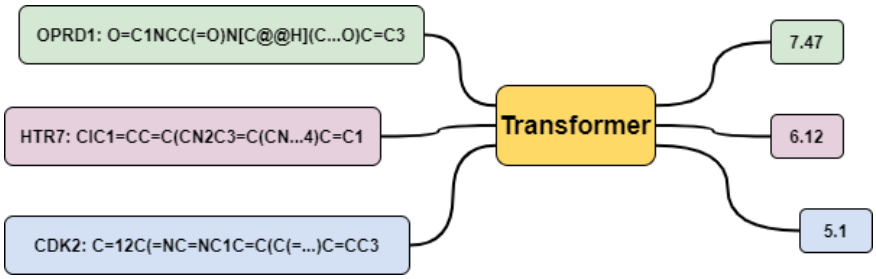


Figure 4.1: The input and output of Multi-task QSAR regression using the Transformer Neural Network. The input is the gene symbol and the chemical compound as one string and the output is the activity value (a real number).

fine-tunes all parameters. Each Transformer model that is used is trained simultaneously on all biological activities or on all physical chemistry properties.

For comparison of the text-based models, the feature-based model SVR is used which is extensively explored in the literature [64, 47]. Now, the molecular structure of each observation from the datasets is represented as extended-connectivity fingerprints (see Section 2.5) which are calculated features. These models are trained on each task separately. The hyperparameters of all models are searched using the Bayesian Optimization technique, providing a strategical way of searching the optimal ones.

The training workflow diagrams for both text-base and feature-based approaches are illustrated in Figure 4.2. Finally, the results are compared using the root mean squared error (RMSE) and the coefficient of determination (R^2) on each task separately for all model approaches.

All experiments are coded in Python[1] version 3.7 computer language utilizing a number of open-source packages. For the visualizations the packages matplotlib [38] (version 3.3.4[2]) seaborn [83] (version 0.11.1[3]) is used, and for the diagrams the site draiw.io[4]. An extensive list with all packages that are used in this thesis can be found in the Appendix.

4.2 Transformer Encoder-Regression

For implementing the Transformer networks, the packages PyTorch [56] (version 1.8.0[5]) and PyTorch Lightning [28] (version 1.2.3[6]) are used.

The Transformer network that is used has as input the sum of the input sequence embeddings and the positional encoding (see Section 3.2.1) for each molecule input. Then a number of encoder blocks is used, as described in Section 3.2.3, to encode the information and create meaningful representation of the input sequence. The encoder block has the following hyperparameters: embeddings dimenstions, number of attention heads, dropout probability and neurons of the FNN which belongs to the encoder block. As mentioned in Section 3.2, the input and output of an encoder block has the same size, so the input flows from the first encoder block and then to the second encoder block. As it is illustrated in Figure 4.3, from the output of the second encoder block, only the array that is align with the name of the task is used. This approach of using only the first output of the last encoder block for discriminative tasks has first seen in the field of NLP, particularly in BERT [24] model.

[1] http://www.python.org
[2] https://matplotlib.org/
[3] https://seaborn.pydata.org/
[4] https://draw.io/
[5] https://pytorch.org/
[6] https://www.pytorchlightning.ai/

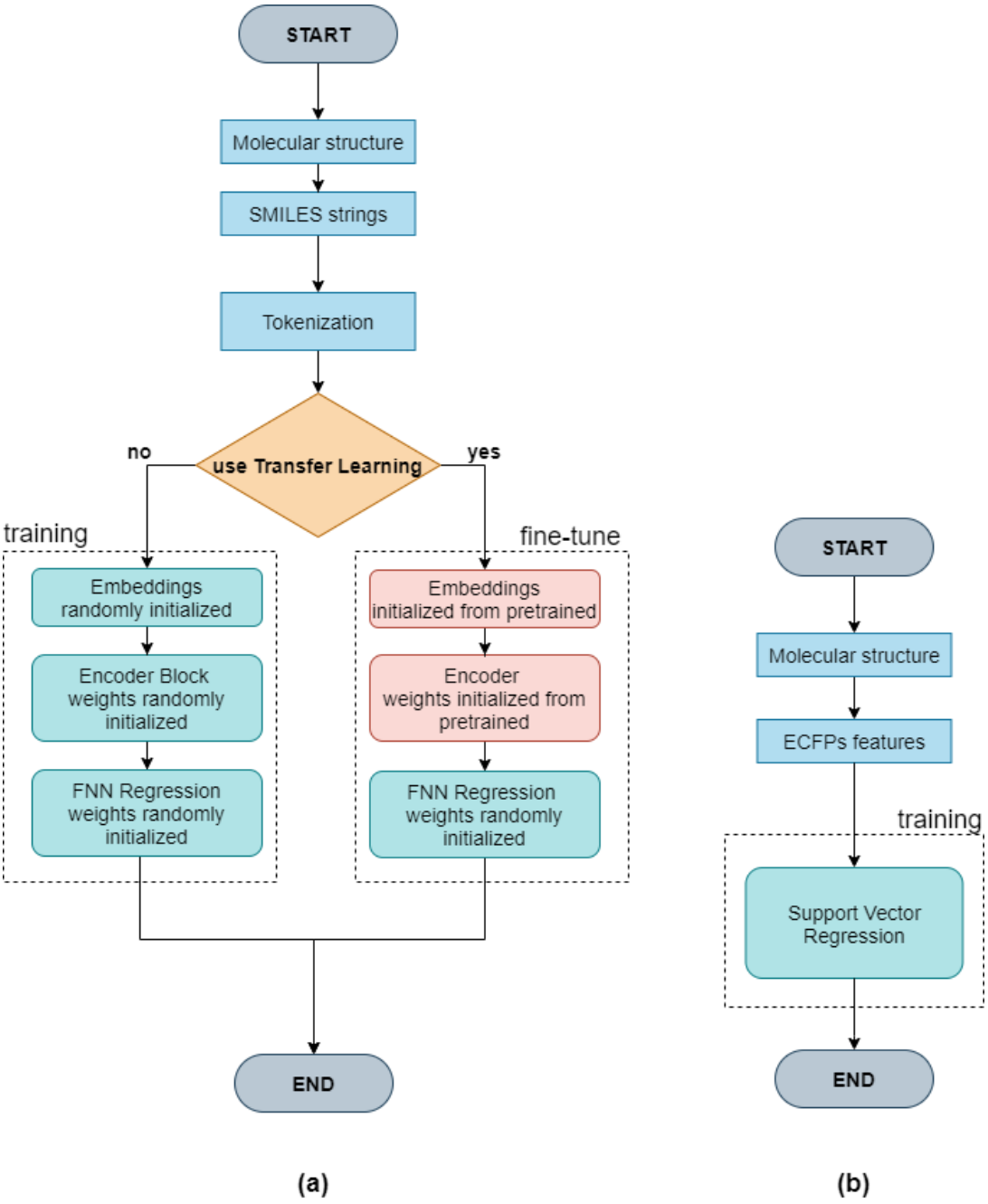


Figure 4.2: On the left the workflow diagram of the text-based Transformer model without and with Transfer Learning (a). On the right the workflow diagram of the feature-based Suport Vector Regression model. ECFPs stands for Extended-connectivity fingerprints which are the features that are created from the molecular structure (b).

Afterwards, the chosen encoder output goes through a FNN to get the output value. The FNN has the ReLU activation function between its layers except the last layer which has no activation for predicting the molecular activities/properties. The hyperparameters of this FNN are the number of layers and neurons and the dropout probability. The chosen hyperparameters for all Transformer models can be found in the Appendix's Section A.3.

The Transformer encoder-regression architecture is illustrated on Figure4.4.

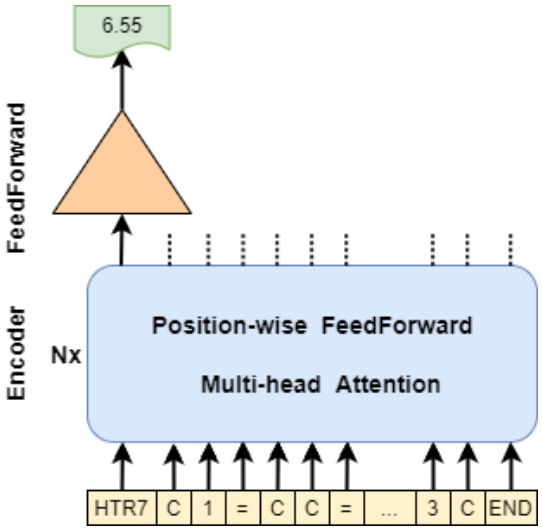


Figure 4.3: The representation of the Encoder Regression model. The input sequence is fed into the encoder blocks and then from the output of the last block only the vector that is aligned with the task token goes to the Feedforward neural network for Regression.

The error of the Transformer network is assumed to be normally distributed, thus the maximum likelihood estimate of the normal distribution will be optimized. By doing the derivations this can be simplified as minimizing the mean squared error (MSE).

$$MSE(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n}(\hat{y_i} - y_i)^2 \tag{4.1}$$

where y denotes the true value and $\hat{y}$ the estimated value of the network.

Thus, as loss function to be minimized the MSE has been chosen. The optimizer is Adam [40]. The learning rate scheduler One Cycle is utilized where the learning rate is increasing for a number of epochs (which is called warm-up phase) until it reaches its maximum value and then it is gradually decreasing, helping on quicker convergence [72].

In order to find the suitable hyperparameters for the Transformer model two-steps were used. First, some initial experiments were conducted, with manual choices and sort running times, to determine the suitable intervals for each hyperparameter. Second, the search for the optimal hyperparameters was done by using Bayesian Optimization with the Optuna [3] python package(version 2.3.0[7]). The search space for each hyperparameter was chosen based on the intervals from the first step and using 100 trials per model. The hyperparameters have the following search spaces: the learning rate has the loguniform [0.000001, 0.01], the weight decay has the loguniform [0.000000001, 0.01], the embedding size has the set 256, 512, the number of heads have the set {4, 8, 16}, the number of encoder blocks take uniform integers in [1, 4], the dimension of the FNN in the encoder block has the set {512, 1024, 2048}, the dropout of the encoder blocks takes values from the uniform [0.1, 0.5], the FNN of the regression part has the set {512, 1024, 2048}, and the dropout of the FNN takes values from the uniform [0.1, 0.8]. The chosen hyperparameters that achieved the lowest validation loss for each model can be found in the Appendix.

4.3 Transfer Learning in Cheminformatics

Transfer Learning [89] is a widely used technique in variety of deep learning problems, also in cheminformatics and drug discovery [14]. The idea of it is to transfer the gained knowledge

[7]https://optuna.org/

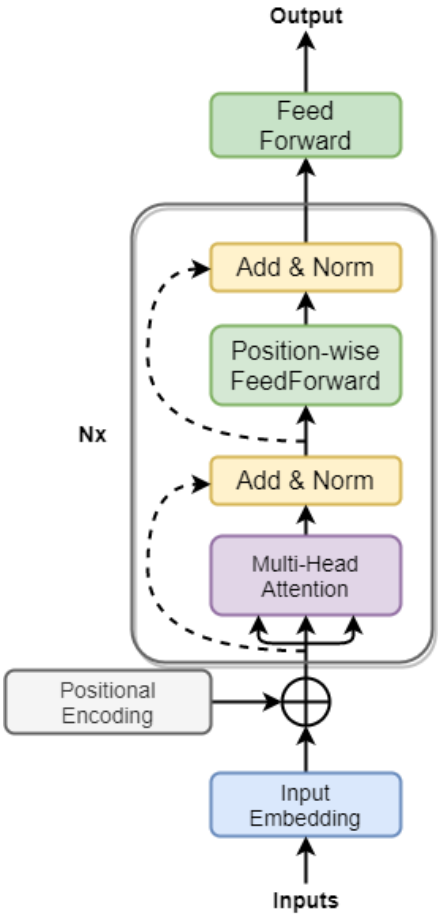


Figure 4.4: The Encoder-Regression Transformer architecture (following Vaswani et al. 2017 [79]), with N encoder blocks and a FeedForward Neural Network with three layers, where the last layer has one output neuron.

from solving one problem to another (see Section 3.3). In this thesis, transfer learning is performed by experimenting with two pretrained Transformer neural networks trained on unlabeled data. These two Transformer are pretrained by a scientist from AstraZeneca, and their weights are provided for this work. The first model is trained on around 1.5 million molecules from ChEMBL [51] database and the second is a larger Transformer trained on around 100 million molecules from ZINC [75] database. The goal of the pretrained model is to learn the representation and the syntax of SMILES strings. The pretrained model is the Bidirectional and Auto-Regressive Transformers (BART) [44] model which consists of encoder decoder Transformer architecture.

As it is illustrated in Figure 4.5, the training is performed by masking some parts of the input sequence (molecule), as it is displayed one token can be masked (here the last token '1') or a span of more than one tokens can be masked (here 'C' and '1' are masked together). These masking techniques erase some elements of the input sequence and introduces noise to the input. Then the corrupted input is fed into the encoder. The decoder is trying to construct the whole original sequence in an autoregressive way which means that the decoder is generating one by one the tokens from left to right given the previous tokens of the sequence. For example, to predict the third token 'C' in Figure 4.5, the decoder receives as input the encoder's output and the previous tokens of the sequence ('<s>' (starting token) and 'Cl').

As mentioned above the pre-trained weights of these two models are provided by the company. For this thesis only the parameters from the embedding layer and the encoder are

used and after the extracted encoder block, m number of fully connected layers are added, with ReLU activation and dropout between the fully connected layers, converting it into a regression model. Then, the models are fine-tuned on the multi-task QSAR/QSPR problems using the same optimizer. The hyperparameters for all Transformer models can be found in the Appendix.

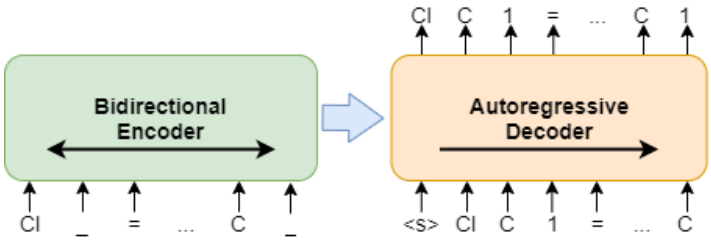


Figure 4.5: The encoder decoder BART Transformer model with the masked input and the autoregressive generation of the input sequence (following Lewis M., Liu Y. et al., 2019 [44]).

The first pre-trained model on ChEMBL has 4 encoder layers with embedding size of 256 dimensions, 8 attention heads, the FNN of the encoder block has 2048 neurons and afterwards a FNN with m_S layers for regression is added, which has in total around 5 million parameters. While the larger pre-trained model on ZINC has 6 encoder layers with embedding size of 512 dimensions, 8 attention heads, the FNN of the encoder block has 2048 neurons and then a FNN for regression is added m_L layers, which has in total around 20 million parameters. The full list of hyperparameters from the additional FNN blocks for all models can be found in the Appendix. The hyperparameters of the pretrained part are fixed and the rest are chosen using the Optuna framework as described on the Subsection 4.2.

4.4 Support Vector Rregression using ECFPs

In this thesis the Tanimoto kernel (also called Tanimoto index) is used for the Support Vector Rregression (SVR) models, which is a well known tool to capture the similarity between sets of binary vectors (in our case the ECFPs). In the paper "Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?", the authors compare eight similarity metrics on a large dataset of molecular fingerprints, and they concluded that Tanimoto index is one of the best metrics on molecular similarity. In cheminformatics, it is one of the most popular kernels in literature [48, 7] for molecular fingerprint representations. Additionally, it is quick to be calculated as it consists of four dot products between binary vectors as shown below:

$$k(x, x') = \frac{\langle x, x' \rangle}{\langle x, x \rangle + \langle x', x' \rangle - \langle x, x' \rangle} \tag{4.2}$$

The SVR models are trained in each task with ECFPs as features using the scikit-learn [59] package (version 0.22.2[8]) and the custom Tanimoto kernel implemented in numpy [30] package (version 1.19.2[9]). The hyperparameters are chosen using Bayesian Optimization, utilizing the scikit-optimize [35] (version 0.8.1[10]) package. As described in section3.4.2, as surrogate model the gaussian process is used, which can capture the uncertainty around points, to model the validation loss, and the expected improvement as acquisition function to choose the next point to evaluate. The prior distributions of the hyperparameters are: uniform distribution on [0.1-50.0] for C, uniform distribution on [0.00001-1.0] for γ and a uniform distribution on

[8] https://scikit-learn.org/
[9] https://numpy.org/
[10] https://scikit-optimize.github.io/

[0.001-1.0] for ϵ. The optimal hyperparameters for all SVR models on all datasets are shown on the appendix section.

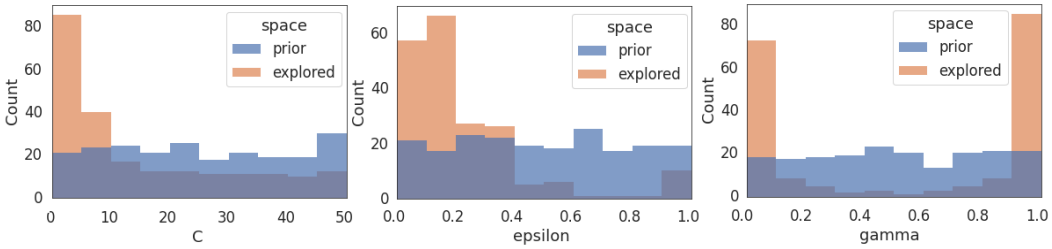


Figure 4.6: The histograms of the configuration space (uniform prior on specified interval) and the explored space of the bayesian optimization. The three hyperparameters are C, ϵ, γ of SVR, where 200 trials are conducted on the dataset of OPRD1 gene symbol.

An example of Bayesian Optimization search space is illustrated on Figure 4.6, to find the SVR's hyperparameters on the QSAR task of the gene symbol OPRD1. The histograms of the hyperparameters configuration (which we chose as prior knowledge) and explore spaces are shown, where 200 trials were performed to illustrate the behavior of Bayesian Optimization, but as observed around 30 trials were needed to find optimal hyperparameters. For the inverse regularization hyperparameter C, with uniform prior the algorithm explored a bit almost all the given space and focused on the values around 5, where the lowest validation error found for C=4.11. The same behavior is seen from the hyperparameter epsilon which given a uniform prior, again the algorithm explored a bit the whole space and focused the search exploiting around 0.1 where the optimal epsilon was on 0.11. To find γ the algorithm exploits highly on the upper and lower boundary of the configuration space, around 0.1 and 0.9, probably the validation loss had not big difference choosing high or low value for gamma, where the chosen value was 0.92

4.5 Oversampling

In literature exists different techniques to oversample a dataset [32]. Most of these techniques only suitable for tabular data, where we want to oversample text-based data (SMILES strings), thus these oversampling techniques are not considered. Usually, oversampling in machine learning is referred to classification problems where the classes are imbalanced. In this thesis, multiple regression tasks are addressed and the main approach is to train simultaneously on a number of regression tasks (molecular activity or property prediction) using Transformer models. Inspired from the oversampling approaches on imbalanced classes, we apply oversampling on regression tasks. Some of the regression tasks have less observations (molecules) than others, thus as oversampling in this work we imply that the tasks with the less observations will be duplicated m times in order to have approximately the same size as the task with the most observations. For example, the training sets of the three physical chemistry properties Lipophilicity, ESOL and FreeSolvation have 3150, 846 and 482 molecules, respectively. The tasks that will be oversampled are ESOL and FreeSolvation where the first will be duplicated 3.5 times and the latter 6 times, resulting to have 2961 and 2892, respectively.

This oversampling approach is promising on SMILES strings when data augmentation (see Section 2.4.2) is used. On each mini-batch every SMILES string is augmented on-the-fly (i.e. while the model is training), thus the Transformer models receive different SMILES representations of the same molecule. SMILES augmentation helps the model not to see the same molecules and avoids memorizing them. Finally, undersampling is not a valid option on

Transformer models while having limited amount of training data, because these models are data hungry [1].

5 Results

This chapter is divided into two sections. First the results of the 133 molecular biological activity (QSAR) tasks are presented, and second the results of three molecular physical chemistry property (QSPR) tasks.

A number of experiments were conducted to answer the research question 1: Can a text-based model, with the same parameters and architecture, predict multiple molecular activities/properties and if yes, to which extend? As text-based model the Transformer is used because we want to test the novel approach of multi-task regression with Transformers (see Chapter 4) which was inspired from the paper [60]. It is trained simultaneously on QSAR or QSPR tasks, and its architecture consists of a number of Encoder blocks and Feedforward Neural Network for regression (see Section 4.2). Moreover, two pretrained Transformer models are used to answer the research question 2: Can transfer learning improve the performance of the text-based model, by giving prior knowledge on the model about the syntax of chemical compounds? Finally, the Transformer model with higher generalization performance was chosen, in aftermath of the previous questions, in order to be compared with a feature-based to answer the research question 3: How does it perform compare to a feature-based traditional machine learning algorithm? As features, to represent the molecular structure, the ECFP fingerprints are used, they are widely used in QSAR and QSPR tasks with high performance. While as feature-based model the SVR with Tanimoto kernel is used because it performs well as shown in literature and is relatively quick to train.

Through this chapter, the Transformer model is denoted as EncRegr meaning that it consists of Encoder and Regression parts. The pretrained models which are fine-tuned are denoted as EncRegrTL_ChEMBL and EncRegrTL_ZINC, meaning Encoder Regression with Transfer Learning on ChEMBL and ZINC database respectively. As far as the size of these two pretrained models, the EncRegrTL_ZINC has four times more parameters than EncRegrTL_ChEMBL which has around 4 million parameters, more details can be found on Section 4.3. While if oversampling is used the model is denoted as oversEnvRegrTL_ChEMBL which means oversampled data using the Encoder Regression with Transfer Learning on ChEMBL database.

In result tables the best score is marked **bold**. To address any ambiguity, next to each evaluation metric, arrow up (↑) means that higher values are better and arrow down (↓) means that lower values are better. Specifically, for R^2 higher values are better and for RMSE lower

values. The numerical results are rounded on three decimal places for the first section and on four decimal places for the second.

5.1 Biological Activities

The biological activity datasets consists of 133 gene symbols (i.e. regression tasks). The Transformer models are trained simultaneously on all tasks and evaluated afterwards separately on each task, in order to be easily comparable. SVR models are trained and evaluated on each task separately. The same train/test splits are used for all model approaches, having fixed information as input and output between different model evaluations and being consistent on their comparison.

The bar chart Figure 5.1 shows the coefficient of determination R^2 of eight gene symbols on the test set, and their error bars which are calculated using the Fisher transformation on Pearson correlation (see Subsection 3.5.3). The gene symbols that are displayed are chosen to have different number of molecules, in order to observe the behavior on these tasks with different sizes, the rest bioactivity tasks can be found in the Appendix A.4. It is noticeable that ADAM17, CHRNA7, GSK3A and MAOB have wide error bars because these four gene symbols have small number of molecules compared to the other four tasks of the figure. Observing the Transformer models, the pretrained on ZINC EncRegr outperforms significantly the pretrained on ChEMBL and the one that does not use transfer learning; on all eight displayed tasks. In five out of the eight tasks that are illustrated, the feature-based model (SVR) has higher R^2 that the best Transformer model (EncRegrTL_ZINC). Overall, SVR outperforms EncRegrTL_ZINC on 99 of the 133 gene symbols, comparing their test performance (see Appendix).

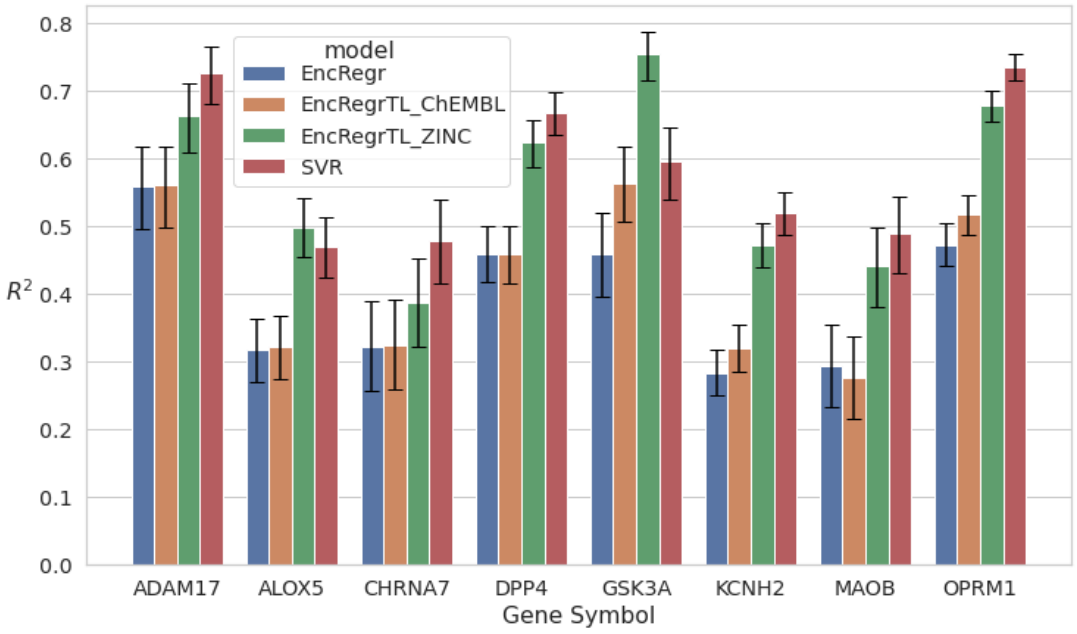


Figure 5.1: The R^2 on test sets of 8 molecular bioactivities, where their error bars are calculated using the Fisher transformation on Pearson correlation. The three models are the Transformer without Transfer Learning (EncRegr), with Transfer Learning on ChEMBL and on ZINC (EncRegrTL_ChEMBL and EncRegrTL_ZINC respectively), and the Support Vector Regression (SVR).

Table 5.1 shows the RMSE and R^2 on train and test set of the same eight gene symbols as the bar chart above; next to each gene symbol the number of molecules that it has is displayed.

Gene Symbol	EncRegr	EncRegrTL C	EncRegrTL Z	SVR
ADAM17 (1371 mols)				
Train RMSE	0.5681	0.6135	0.2488	0.1695
Test RMSE (↓)	0.7549	0.7437	0.6635	**0.5942**
Train R^2	0.7238	0.6768	0.9469	0.9764
Test R^2 (↑)	0.5599	0.5613	0.6633	**0.7259**
ALOX5 (2891 mols)				
Train RMSE	0.5262	0.5335	0.2727	0.2667
Test RMSE (↓)	0.6730	0.6609	**0.5675**	0.5762
Train R^2	0.5568	0.5439	0.8809	0.8914
Test R^2 (↑)	0.3171	0.3218	**0.4987**	0.4700
CHRNA7 (1443 mols)				
Train RMSE	0.6825	0.7117	0.3923	0.4417
Test RMSE (↓)	0.9392	0.9304	0.9307	**0.8155**
Train R^2	0.6593	0.6295	0.8874	0.8607
Test R^2 (↑)	0.3235	0.3253	0.3875	**0.479**
DPP4 (3293 mols)				
Train RMSE	0.6100	0.6247	0.2853	0.2283
Test RMSE (↓)	0.7886	0.7827	0.6539	**0.6093**
Train R^2	0.6364	0.6172	0.9201	0.9486
Test R^2 (↑)	0.4602	0.4593	0.6239	**0.6681**
GSK3A (1606 mols)				
Train RMSE	0.4354	0.4323	0.1825	0.1169
Test RMSE (↓)	0.6779	0.5828	**0.4546**	0.5604
Train R^2	0.7704	0.7730	0.9596	0.9866
Test R^2 (↑)	0.4595	0.5643	**0.7541**	0.5957
KCNH2 (5275 mols)				
Train RMSE	0.5002	0.5338	0.2212	0.1389
Test RMSE (↓)	0.6941	0.6706	0.5966	**0.5554**
Train R^2	0.6052	0.5502	0.9227	0.9726
Test R^2 (↑)	0.2842	0.3194	0.4733	**0.5202**
MAOB (1688 mols)				
Train RMSE	0.7038	0.7256	0.2860	0.3200
Test RMSE (↓)	0.9372	0.9404	0.8341	**0.7684**
Train R^2	0.5939	0.5679	0.9329	0.9203
Test R^2 (↑)	0.2934	0.2763	0.4411	**0.4893**
OPRM1 (5830 mols)				
Train RMSE	0.7585	0.7676	0.4355	0.4531
Test RMSE (↓)	1.0570	1.0020	0.8162	**0.7321**
Train R^2	0.7145	0.7074	0.9058	0.8995
Test R^2 (↑)	0.4734	0.5173	0.6796	**0.7366**

Table 5.1: The RMSE and R^2 on train and test set of the models Transformer without Transfer Learning (EncRegr), with Transfer Learning on ChEMBL (EncRegrTL_ChEMBL), on ZINC (EncRegrTL_ZINC), and the Support Vector Regression (SVR), on 8 gene symbols. Next to these gene symbols the are shown the number of molecules that they have.

It seems that the two best models, the EncRegrTL_ZINC and the SVR, have the tendency to overfit in the training set, but the selected hyperparameters shown that both models generalize better on the validation set by having this behavior on train set.

Figure 5.2 shows the R^2 comparison on test set between text-based Transformer models and feature based SVR on all 133 gene symbols. The x-axis is the R^2 of SVR and the y-axis is the R^2 of Transformer model. There are 133 blue dots that represent each gene symbol comparing the Transformer without Transfer Learning and the SVR, showing that SVR performs better on all gene symbols, all blue dots are below the red diagonal line where the red diagonal line denotes that Transformer and SVR performs equally good. While the 133 orange dots compare the Transformer with Transfer Learning (pretrained on 100 million molecules from ZINC) and the SVR, where in the majority of the tasks SVR outperforms the Transformer with Transfer Learning. More specifically, SVR performs better on 99 of the 133 QSAR tasks. It is noticeable the big improvement of the Transformer model by using Transfer Learning, where the line that captures the overall trend is moved closer to the red line, i.e. the blue line is shifted to the orange which is close to the red diagonal line.

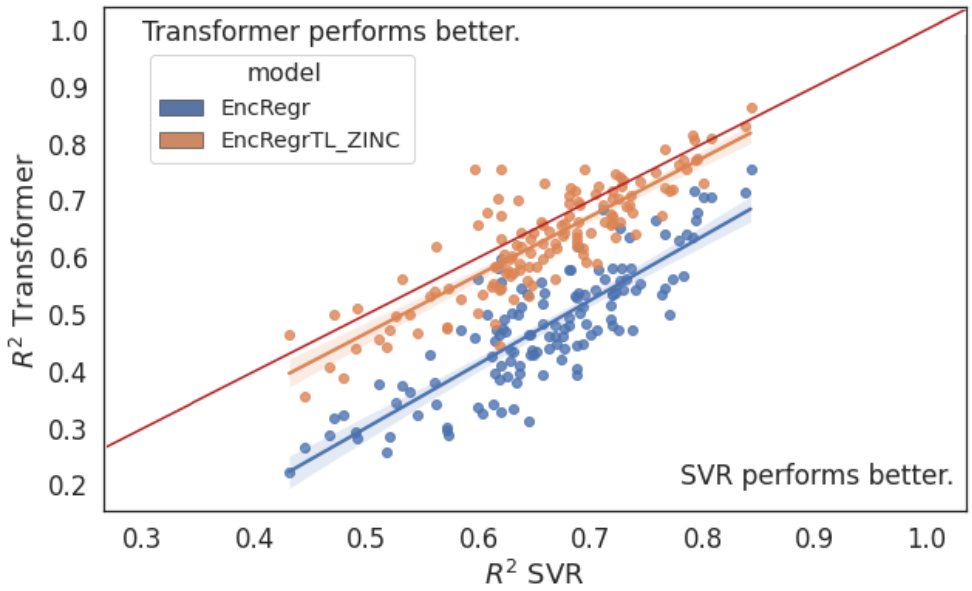


Figure 5.2: Comparison of the R^2 on test sets of SVR against the Transformer model. The blue dots denote the 133 regression tasks (bioactivities) where the x-axis is the R^2 of SVR and the y-axis the R^2 of the Transformer without Transfer Learning (EncRegr). While the orange dots denote again the 133 QSAR tasks but now the y-axis is the R^2 of the Transformer using Transfer Learning on ZINC (EncRegrTL_ZINC).

On Figure 5.3 the histogram of the R^2 (on the test sets) differences between the SVR and the pretrained on ZINC Transformer model is displayed, for all 133 biological activities. The positive differences indicate that SVR model performs better. Most of differences are between -0.1 and 0.1 showing that most of the performances are close. Moreover, the majority are positive differences indicating that SVR outperforms the pretrained on ZINC Transformer model in most of the tasks, specifically on 99 out of the 133 QSAR tasks.

In order to compare if the SVR performs statistically significantly better than the Transformer with Transfer Learning on all the 133 bioactivities, the nonparametric paired test Wilcoxon signed-rank test (see Section 3.5.4) is used. The paired samples are the test R^2 of the SVR and the EncRegrTL_ZINC evaluated on the 133 biological activities, which gives 133 paired values. The null hypothesis states that the median (m_A) of the test R^2 using the SVR model is less or equal than the median (m_B) of test R^2 using EncRegrTL_ZINC, while the

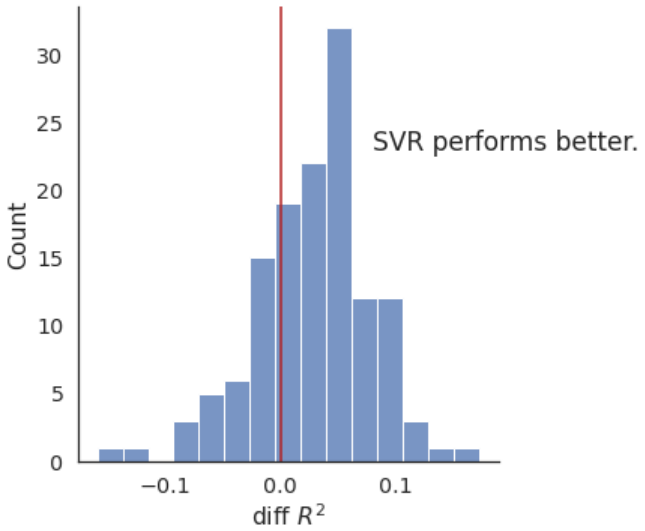


Figure 5.3: The histogram of the differences from all 133 biological activities in test R^2 of the SVR minus the pretrained on ZINC Transformer model. The differences that have positive value (i.e. belong on the right side of the vertical red line) denote that SVR performs better.

alternative hypothesis states that the median of the test R^2 using the SVR model is greater than the median of test R^2 using EncRegrTL_ZINC.

- H_0: $m_A \leq m_B$
- H_1: $m_A > m_B$

We reject the null hypothesis with p-value $= 5 \cdot 10^{-9}$ at a 5% significance level. Overall the SVR performs statistically significantly better on the bioactivities than the Transformer model that was pre-trained on ZINC database.

5.2 Physical Chemistry Properties

The physicochemical property datasets are split 20 times on random train test splits and the same splits are used to train each model (SVR and Transformers). Note that some molecules overlap between different property datasets (eg. some molecules are both in ESOL and FreeSolv datasets), thus these molecules have been chosen to be on the train set in all random splits. This prevents from overestimated results, for example a molecule could be in the train set with the ESOL value and in the test set with the FreeSolv value, so it will be easy for the model which is trained simultaneously on all three properties to predict the FreeSolv on the test set because ESOL and FreeSolv are associated chemically. First the Transformer models will be compared to find the higest perfomed on and then the one with the best performance will be compared with the baseline SVR.

Table 5.2 displays the mean plus/minus the standard error (standard deviation over the square root of the number of splits) of RMSE and R^2 from the 20 random splits on train and test sets for all three physicochemical properties. The four models that it shows are the Transformer without Transfer Learning (EncRegr), the Transformer that was pretrained on around 1.5 million molecules from ChEMBL database (EncRegrTL C), the same pretrained model on ChEMBL but using oversampling on the ESOL and FreeSolvation properties in order to have approximately the same size as Lipophilicity is denoted as ovrEncRegrTL C, and the pretrained Transformer on around 100 million molecules from ZINC database with oversampling is denoted as ovrEncRegrTL Z.

Property	EncRegr	EncRegrTL C	ovrEncRegrTL C	ovrEncRegrTL Z
Lipophilicity				
Train RMSE	0.129 ± 0.001	0.094 ± 0.000	0.090 ± 0.000	0.047 ± 0.001
Test RMSE (↓)	0.161 ± 0.001	0.117 ± 0.000	0.115 ± 0.001	**0.111** ± 0.001
Train R^2	0.590 ± 0.003	0.780 ± 0.002	0.799 ± 0.002	0.945 ± 0.003
Test R^2 (↑)	0.382 ± 0.005	0.667 ± 0.003	0.676 ± 0.004	**0.698** ± 0.005
ESOL				
Train RMSE	0.058 ± 0.000	0.052 ± 0.000	0.042 ± 0.000	0.026 ± 0.001
Test RMSE (↓)	0.070 ± 0.001	0.061 ± 0.001	0.056 ± 0.001	**0.052** ± 0.001
Train R^2	0.867 ± 0.002	0.890 ± 0.002	0.927 ± 0.001	0.972 ± 0.001
Test R^2 (↑)	0.811 ± 0.005	0.853 ± 0.005	0.876 ± 0.005	**0.895** ± 0.004
FreeSolvation				
Train RMSE	0.059 ± 0.000	0.052 ± 0.000	0.037 ± 0.000	0.021 ± 0.000
Test RMSE (↓)	0.077 ± 0.002	0.070 ± 0.003	0.059 ± 0.002	**0.053** ± 0.002
Train R^2	0.819 ± 0.004	0.859 ± 0.003	0.927 ± 0.002	0.976 ± 0.001
Test R^2 (↑)	0.736 ± 0.013	0.786 ± 0.014	0.848 ± 0.010	**0.874** ± 0.010

Table 5.2: RMSE and R^2 of the models trained on the same 20 random train/test splits, the mean plus/minus the standard error (standard deviation over the square root of the number of splits) are shown for the three physical chemistry properties. The models are the Encoder Regression Transformer (EncRegr), the EncRegr that was pretrained on ChEMBL (EncRegrTL C), the same model pretrained on ChEMBL but with oversampling of the minority tasks ESOL and FreeSolvation denoted as ovrEncRegrTL C, and the pretrained EncRegr on ZINC with oversampled properties is denoted as ovrEncRegrTL Z.

The ovrEncRegrTL_Z model outperforms the other Transformer model approaches looking at the test RMSE or R^2 on all three physicochemical properties from the Table 5.2. Moreover, using Transfer Learning it seems that improves performance by looking how EncRegr increases the test R^2 using the pretrained on ChEMBL weights (EncRegrTL C model). When oversampling the minority tasks ESOL and FreeSolvation, it seems that we get better performance as seen from the model EncRegrTL C, where test R^2 increases for Lipophilicity from 0.66 to 0.676, for ESOL from 0.853 to 0.876 and for FreeSolvation from 0.786 to 0.848.

Figure 5.4 shows the improvements of the Transformer model performance by using Transfer Learning and by oversampling the minority properties. The x-axis shows the three physical chemistry properties while the y-axis shows the R^2 on their test sets. We can notice the behavior of EncRegr without and with the use of Transfer Learning, where using Transfer Learning increases the performance on all properties. For Lipophilicity EncRegr has test R^2 0.382 that increased to 0.667 with EncRegrTL_ChEMBL, similarlly for ESOL test R^2 increased from 0.811 to 0.853 and FreeSolvation test R^2 increased from 0.736 to 0.786. Moreover, we observe increase on generalization performance by using oversampling on the pretrained on ChEMBL Transformer (EncRegrTL_ChEMBL), the coefficient of determination increases on all properties. In addition, the variance of the R^2 evaluations on the 20 random train/test splits is decreased in ESOL and FreeSolvation by using oversampling. It can also be seen by using oversampling that a bigger Transformer architecture, pretrained on 100 million molecules from ZINC, outperforms all Transformer model approaches that are used on this thesis. Thus, this model will be compared with the feature-based SVR.

Figure 5.5 illustrates the impact of using Transfer Learning on the validation MSE. The x-axis shows the steps of training where each step is one mini-batch update, and the y-axis shows the validation MSE. The Transformer model is the orange curve while the Transformer

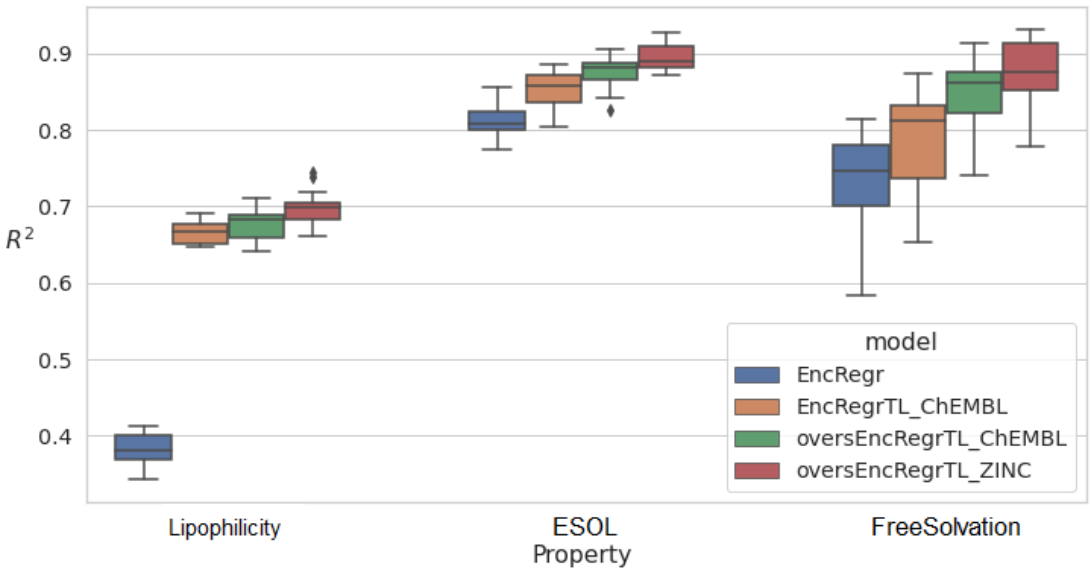


Figure 5.4: The R^2 on test sets of the three physicochemical properties from four approaches of the Transformer model, on 20 random train/test sets. The models are the Encoder Regression Transformer (EncRegr), the EncRegr that was pretrained on ChEMBL (EncRegrTL_ChEMBL), the same model pretrained on ChEMBL but with oversampling of the minority tasks ESOL and FreeSolvation denoted as oversEncRegrTL_ChEMBL, and the pretrained EncRegr on ZINC with oversampled properties is denoted as oversEncRegrTL_ZINC.

with Transfer Learning on ChEMBL is the blue curve. It can be observable that the blue curve (using Transfer Learning) on zero step its loss has lower value (around 0.04) than the orange curve which has value higher than 0.18. Additionally, the blue curve ends up having lower validation MSE than the orange curve.

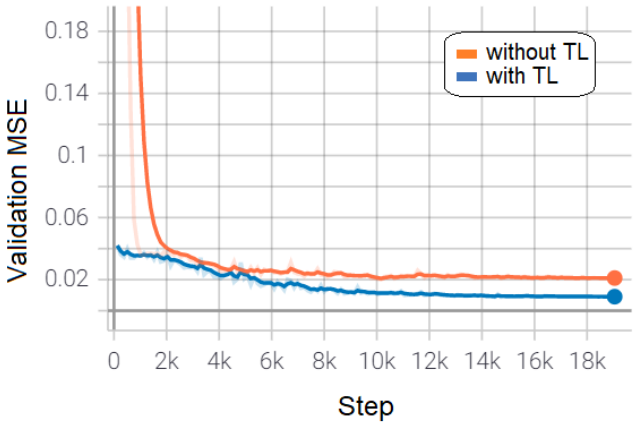


Figure 5.5: Learning curves on validation set of the EncRegr Transformer without Transfer Learning (orange curve) and with Transfer Learning using the pretrained model on ChEMBL (blue curve). The x-axis is the steps (i.e. the mini-batch update), and the y-axis is the validation MSE.

Train and test evaluations (RMSE, R^2) of the best Transformer model oversEncRegrTL_ZINC that is shown in Figure 5.4 and the feature-based model SVR on the three properties are shown on Table 5.3. Observing the train and test R^2, especially on Lipophilicity property, it seems that again oversEncRegrTL_ZINC and SVR are overfitting but this is not the case here because the hyperparameters are chosen reagarding the generalization error, i.e. the lowest validation loss.

Property	**oversEncRegrTL ZINC**	**SVR**
Lipophilicity		
Train RMSE	0.047 ± 0.001	0.006 ± 0.000
Test RMSE (↓)	**0.111** ± 0.001	0.125 ± 0.001
Train R^2	0.945 ± 0.003	0.999 ± 0.000
Test R^2 (↑)	**0.698** ± 0.005	0.617 ± 0.004
ESOL		
Train RMSE	0.026 ± 0.001	0.023 ± 0.001
Test RMSE (↓)	**0.052** ± 0.001	0.078 ± 0.001
Train R^2	0.972 ± 0.001	0.978 ± 0.001
Test R^2 (↑)	**0.895** ± 0.004	0.766 ± 0.010
FreeSolvation		
Train RMSE	0.021 ± 0.000	0.002 ± 0.000
Test RMSE (↓)	**0.053** ± 0.002	0.078 ± 0.005
Train R^2	0.976 ± 0.001	0.980 ± 0.000
Test R^2 (↑)	**0.874** ± 0.010	0.754 ± 0.021

Table 5.3: RMSE and R^2 of the models trained on the same 20 random train/test splits, the mean plus/minus the standard error (standard deviation over the square root of the number of splits) are shown for all physical chemistry properties.

The results for the comparison on physical chemistry properties between the text-based Transformer model oversEncRegrTL_ZINC and the feature-based SVR model are visualized on Figure 5.6. Again the models are evaluated in 20 random train/test splits, the boxplots with blue color are the oversEncRegrTL_ZINC and with orange the SVR. It is clear that oversEncRegrTL_ZINC outperforms SVR on all 20 splits of Lipophilicity and ESOL.

We can observe that the oversEncRegrTL_ZINC model has higher R^2 than SVR on the test sets of all three physical chemistry properties.

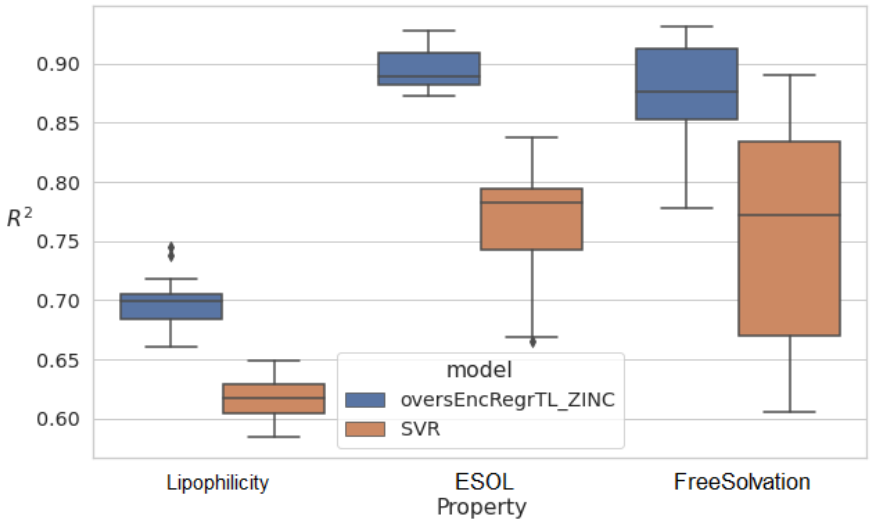


Figure 5.6: The R^2 of test sets on the same 20 random train/test splits. The two models are the Transformer with Transfer Learning and oversampling of the manority tasks, and the SVR trained on single task at a time.

6 Discussion

In this chapter the results and methods that are used will be analyzed and discussed, having as main aim to answer and comment on the research questions that are stated on Section 1.3. Additionally, the societal aspects that are related with this thesis have been discussed on a subsection in the end of this chapter.

6.1 Results

Biological Activities

The impact of Transfer Learning on Transformer models for predicting molecular biological activity values is crucial as seen from Section 5.1. More specifically, the model without Transfer Learning (EncRegr) and the pretrained model on 1.5 million molecules from ChEMBL (EncRegrTL_ChEMBL) perform similarly. The improvement that we observe using EncRegrTL_ChEMBL is not significant, which is visualized on Figure 5.1. In contrast, when we fine-tuned a larger model which was pretrained on more observations, i.e. 100 million molecules from ZINC database (EncRegrTL_ZINC), a huge improvement is observed. The EncRegrTL_ChEMBL model has around 6 million parameters while the EncRegrTL_ZINC model has 20 million parameters. To investigate if using just a bigger model will give better performance, we trained a Transformer with the same number of parameters as EncRegrTL_ZINC but without Transfer Learning, initializing the weights randomly. The results were worse than using a smaller EncRegr model. These experiments indicate that by using a larger model the performance is not improved; however, the combination of a larger model that was pretrained on a big chemical space (here 100 million molecules) has huge improvement on predicting the bioactivity values. Thus, a large Transformer model pretrained on a wide variety of chemical compounds can play a key role in improving the performance on downstream tasks.

Comparing the text-based Transformer models with the feature-based SVR on the 133 bioactivities (see Figure 5.2), we observed that the EncRegr model performs poorly as opposed to SVR, while EncRegrTL_ZINC is comparable with SVR. The SVR model outperforms EncRegrTL_ZINC on the majority of the QSAR tasks and it is overall significantly statistically better as shown in Section 5.1. The behavior that text-based models cannot achieve better results than feature-based models using ECFP features is shown on another recent study "Using Molecular Embeddings in QSAR Modeling: Does it Make a Difference?" [69] (here embed-

dings mean the latent representation of molecules, not specifically the embedding layer that is explained in Section 3.2.1). They used different ways to represent molecules with learnable embeddings, and they compared these models to other models using traditional molecular representations (deterministic features), computed with feature engineering. The authors concluded that there is no evidence that learnable embeddings perform better. Futhermore, in the work of Rodríguez-Pérez R. and Bajorath J. [62], the authors trained SVR models with Tanimoto kernel and DNN models on biological activity tasks, by using ECFPs representation of the molecules. Their results showed that the mean (± standard deviation) of MSE was 0.586 ± 0.198 for DNN and 0.519 ± 0.291 for SVR. Additionally, in another study by Yadi Zhou et al. [88] the authors trained SVM and DNN models using ECFP to represent the molecules on classification and regression molecular activities tasks. They state that "DNN and SVM models showed no significant difference". Taking into account the findings from these studies, we conclude that using a DNN with ECFPs would have a similar performance compared to the SVR. Additionally, SVR models are used as a strong baseline to compare our novel approach of multi-task regression.

Looking at the literature our intuition on the good performance of SVR using the ECFP features is connected with the way the ECFPs features are created. The molecular bioactivity values are highly related to the topological structure of the molecule and the ECFPs descriptors are created taking into account the neighbor structure of each atom of the molecule, where the neighbor area is indicated by a radius [65]. As mentioned in the "Extended-Connectivity Fingerprints" paper [65] "ECFPs were developed specifically for structure−activity modeling", meaning that the ECFPs descriptors are designed to capture molecular features related to molecular activity. This might be an indication why SVM performs better than the Transformer models on bioactivity data.

Each biological activity task has different number of molecules, which varies from 5830 to 1241. As it has stated in this work, the Transformer models are trained simultaneously on all 133 bioactivity datasets. Thus, it is tried to use oversampling on the tasks that have lesser molecules. As explained in Section 4.5, oversampling SMILES might improve performance due to the augmention on SMILES strings in each mini-batch so the model does not see the same molecular representations repeatedly in an epoch. The experiments shown that it is not improving the performance on biological activity tasks, so these results have not been included.

Another useful outcome of Figure 5.2 is that the orange and blue lines, which describe the trend of the performance comparison between the Transformer and the SVR, are parallel to the red diagonal. The red diagonal indicates that the two models perform equally, thus being parallel with the diagonal means that the difficult to predict regression tasks for the Transformer model; are also difficult for the SVR. It implies that the biological activity datasets are not easy to predict. In QSAR tasks the goal is to prioritize large number of molecules, thus to be less accurate in individual predictions is not of high importance, yet of course higher prediction accuracy is always desirable [17].

Physical Chemistry Properties

Comparing the results from physicochemical properties, it can be seen that Transfer Learning again improves the Transformer model's performance on all three properties. A question might rise looking the Figure 5.4, why is that huge the improvement between EncRegr and EncRegrTL_ChEMBL (i.e. by using Transfer Learning) only on Lipophilicity property? To answer this question one might consider that as seen on Figure 2.4, Lipophilicity has longer SMILES string lengths (mean around 50 characters) than the other two properties ESOL and FreeSolvation (mean around 20 characters). These longer sequences make the prediction of Lipophilicity more difficult for the Transformer model that is not using transfer learning. By transferring the knowledge from the pretrained on ChEMBL model, the Transformer has seen a wide variety of molecules and has learned a lot about the syntax of SMILES. In addition, the average length of SMILES string in ChEMBL is around 50 characters, similar to the SMILES

in Lipophilicity, thus the molecules that the model has seen in the pretraining face helped a lot on the performance improvement. Furthermore, Transfer Learning not only improved performance by having lower loss, but converged quicker (starting from a lower loss), as seen by the learning curves of Figure 5.5.

On physical chemistry properties is observed the same behavior as in the biological activities, where by using a larger Transformer model that was pretrained on a huge and wide chemical space (100 million molecules from ZINC) improves the performance. Combining the pretrained model on ZINC and oversampling of the minority properties provides the Transformer model with the highest generalization performance that we got from this thesis experiments.

As seen on Figure 5.6 comparing the oversEncRegrTL_ZINC model and the SVR with ECFPs features shows that oversEncRegrTL_ZINC outperforms significantly on all three properties. SVR using ECFPs features is a strong baseline and quick to train and tune for physical chemistry properties but not the best that exists in the literature. In the paper "MoleculeNet: A Benchmark for Molecular Machine Learning" [87], provides the state of the art model of these datasets on the time that this paper was published. We did not include these results because they come from different train/validation/test splits which were not provided with the datasets, the split is crucial to determine which models performs better because the performances were really close.

6.2 Methods

In literature, all approaches on multi-task property prediction with DNNs are described as having in the output layer of the network multiple neurons where each output corresponds to a different molecular property [47]. This approach requires all different property values for each molecule of the dataset which is not usually possible; the molecule-property labeled datasets are limited and expensive to compute experimentally. Thus, our novel multi-task method utilizing Transformer models overcomes the obstacle of requiring all properties for each molecule. In our aproach as illustrated in Figure 4.1, the predicted property is specified in the input sequence hence there could be molecules in the dataset that only one property value is known and be included in the simultaneously training.

In the experiments of this thesis, the hyperparameter selection of Transformer models was conducted with manual search on short training times and then using extensive search with the Optuna framework which does not give significant improvement on performance than choosing the hyperparameters carefully with some manual trials. This shows that Transformer Neural Network is a robust architecture and can be used without a big effort on tuning its hyperparameters. On the other hand, an extensive search of the hyperparameter space is crucial for the SVR, where we used Bayesian Optimization with the framework scikit-optimize which improved the performance significantly.

SVR models seems to overfit on biological activity datasets (see Table 5.1 and on Appendix), meaning that the training error is notably lower than validation and test error, having a big gap between them. Even by decreasing C the inverse regularization hyperparameter; the validation loss increases. Additionally, the same behavior is observe by the large Transformer model EncRegrTL_ZINC which seems to slightly overfit, and if we increase the dropout probability or the weight decay, then the validation error increases. Thus a question rise, does this mean that these models are overfitting? In this thesis we did not consider this as a training problem. The hyperparmeters on all models are chosen by having the lowest possible validation loss, even if the models look like they overfit the train data.

In this work the experimental time was limited to more or less three months, thus some aspects could have conducted differently. On biological activity tasks each protein array is represented by a name called gene symbol which is one token of the input sequence. On the contrary, the protein array could be a sequence itself and be concatenated with the input

sequence of the molecule in the embedding layer. This approach of using the whole gene expression of the protein array as sequence is used by Oskooei A., Born J., Manica M. et al. [55]; however, on other datasets with classification problems and utilizing Recurrent Neural Networks with attention mechanism. They used RNN to encode the information of the protein array and the molecule, structuring the input by concatenating them into one long sequence. This could give more information to the model about the protein arrays and create more meaningful dependences between similar gene symbols.

Additionally it could lead to better performance if more physicochemical properties were added because only three are used in this thesis. It is known that Transformer Neural Networks need a lot of data to show their power; to some extent this is addressed by the use of Transfer Learning. Another experiment that could be done is to train simultaneously on biological activities and physical chemistry properties or even to add other tasks.

6.3 Source Criticism

In this thesis trustworthy sources considered the published papers on journals or conferences, and the published books from researchers and scientists which are the majority of the used references that can be found in Bibliography at the end of this work. Additionally, a number of unpublished pre-prints were cited which was necessary because most of these are new work that has not be reviewed yet. The majority of the pre-prints that are used, have many citation even from published papers, thus they are considered reliable sources. Also, preprints are usually used in computer science field without peer review, as the models are often open sourced, and thus the results are easier to verify for other researchers. Furthermore, all figures are created for the purpose of this thesis following the theory, some others have been adapted from researcher's work which has Creative Commons (CC) licence where the author's name and work are cited below the figure. Additionally, the Figure 2.6 is displayed as it is from its paper where the permission from the author is taken and the paper is cited again below the figure.

6.4 Ethical Considerations

The datasets that are used in this thesis are downloaded from publicly available databases as mentioned in Chapter 2. Moreover, the models that are used are discriminative i.e. regression models, thus there is no concern on generating new molecules that could rise privacy issues for potential company's ownership. All methods and experiments are conducted following the Ethical Guidelines for Statistical Practice[1] of the American Statistical Association.

[1] https://www.amstat.org/ASA/Your-Career/Ethical-Guidelines-for-Statistical-Practice.aspx

7 Conclusion

In this thesis a novel text-based multi-task regression Transformer architecture, inspired by the field of NLP, was introduced for molecular property and activity prediction. After several experiments and trials, the answers to the research questions are presented below.

7.1 Answers to Research Questions

1: **Can a text-base model, with the same parameters and architecture, predict multiple molecular activities/properties and if yes, to which extend?**

The Transformer architecture that is used consists of an embedding layer with positional encoding, encoder blocks and a FNN for regression with one neuron in the end. The different regression tasks are introduced to the model by having a prefix token which corresponds to each task, so the input is the concatenation of the prefix token and the SMILES string as is illustrated on Figure 4.1. This architecture was able to be trained simultaneously on 133 biological activity regression tasks or on three physical chemistry properties. The results indicated that it was able to distinguish between tasks and to have comparable evaluations with the baseline SVR model. More specifically, the best found Transformer achieved test R^2 on the three physicochemical properties 0.7, 0,9 and 0.87 while SVR achieved 0.62, 0.77 and 0.75 respectively. This multi-task approach could be used for other tasks as well, opening other paths of multi-task learning by combining regression tasks with generative and classification tasks.

2: **Can transfer learning improve the performance of the text-base model, by giving prior knowledge on the model about the syntax of chemical compounds?**

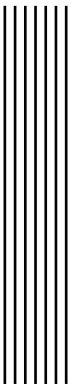
By testing two different pretrained Transformer models, one on 1.5 million molecules from ChEMBL and another larger model on 100 million molecules from ZINC we get a significant improvement on performance by using Transfer Learning. More specifically, on biological activities the pretrained Transformer on ChEMBL did not significant improve the performance, but by using the large model that was pretrained on ZINC, we observed a huge improvement and the pretrained on ZINC model achieved comparable performance to the traditional feature based SVR approach. Additionally, on physical chemistry properties, both pretrained models increased the performance of Transformer networks. Thus, the main outcome is that using large pretrained models on a wide chemical space is the key to improve the performance on property and activity prediction.

3: **How does the text-based model perform compare to a feature-based traditional machine learning algorithm?**

A traditional approach to QSAR/QSPR prediction usually use the well studied ECFPs features, in this work the SVR model with the Tanimoto kernel is used for comparison. As discussed in Section 6.1, on bioactivities datasets the Transformer model with Transfer Learning is comparable with the SVR on the 133 QSAR tasks but overall SVR is statistically significantly better. On the other hand, on physical chemistry properties the Transformer model with Transfer Learning outperforms SVR on all three properties that are tested.

7.2 Future Work

This thesis shown that the multi-task regression Transformer, with the prefix token to distinguish between QSAR/QSPR tasks, achieves competitive results compared to strong baselines, also it shown the huge impact of Transfer Learning on the performance. As continuation of this work, the idea of multi-task Transformer could be extended and different tasks can be added, not only property and activity prediction. More specifically, a model could be trained simultaneously on property/activity prediction, molecular optimization, organic synthesis, de novo generation and other tasks. The architecture of this model could be changed to full Encoder-Decoder Transformer to handle generative and discriminative tasks which would be distinguished by their prefix tokens. Thus, this model will combine information from different aspects of the chemical space; creating complex information from supervised tasks. This might lead to an exciting direction of drug discovery process.



Bibliography

[1] Amina Adadi. "A survey on data-efficient algorithms in big data era." In: *Journal of Big Data* 8.1 (Jan. 2021), p. 24. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00419-9.

[2] Apoorv Agnihotri and Nipun Batra. "Exploring Bayesian Optimization." In: *Distill* (2020). https://distill.pub/2020/bayesian-optimization. DOI: 10.23915/distill.00026.

[3] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. "Optuna: A Next-generation Hyperparameter Optimization Framework." In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[4] Zaid Alyafeai, Maged Saeed AlShaibani, and Irfan Ahmad. *A Survey on Transfer Learning in Natural Language Processing*. 2020. arXiv: 2007.04239 [cs.CL].

[5] Josep Arús-Pous, Simon Viet Johansson, Oleksii Prykhodko, Esben Jannik Bjerrum, Christian Tyrchan, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. "Randomized SMILES strings improve the quality of molecular generative models." In: *Journal of Cheminformatics* 11.1 (Nov. 2019), p. 71. ISSN: 1758-2946. DOI: 10.1186/s13321-019-0393-0.

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization." In: *arXiv preprint arXiv:1607.06450* (2016). arXiv: 1607.06450 [stat.ML].

[7] Dávid Bajusz, Anita Rácz, and Károly Héberger. "Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations?" In: *Journal of Cheminformatics* 7.1 (May 2015), p. 20. ISSN: 1758-2946. DOI: 10.1186/s13321-015-0069-3.

[8] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, 2016. ISBN: 9781493938438.

[9] Esben Jannik Bjerrum. "SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules." In: *arXiv preprint arXiv:1703.07076* (2017). arXiv: 1703.07076 [cs.LG].

[10] David C. Blakemore, Luis Castro, Ian Churcher, David C. Rees, Andrew W. Thomas, David M. Wilson, and Anthony Wood. "Organic synthesis provides opportunities to transform drug discovery." In: *Nature Chemistry* 10.4 (Apr. 2018), pp. 383–394. ISSN: 1755-4349. DOI: 10.1038/s41557-018-0021-z.

[11] Eric Brochu, Vlad M. Cora, and Nando de Freitas. "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning." In: *arXiv preprint arXiv:1012.2599* (2010). arXiv: 1012.2599 [cs.LG].

[12] Matthew P. Brown, Myla Lai-Goldman, and Paul R. Billings. "CHAPTER 31 - Translating Innovation in Diagnostics: Challenges and Opportunities." In: *Genomic and Personalized Medicine*. Ed. by Huntington F. Willard, Ph.D. and Geoffrey S. Ginsburg. New York: Academic Press, 2009, pp. 367–377. ISBN: 978-0-12-369420-1. DOI: https://doi.org/10.1016/B978-0-12-369420-1.00031-7.

[13] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners." In: *arXiv preprint arXiv:2005.14165* (2020). arXiv: 2005.14165 [cs.CL].

[14] Chenjing Cai, Shiwei Wang, Youjun Xu, Weilin Zhang, Ke Tang, Qi Ouyang, Luhua Lai, and Jianfeng Pei. "Transfer Learning for Drug Discovery." In: *Journal of Medicinal Chemistry* 63.16 (2020). PMID: 32672961, pp. 8683–8694. DOI: 10.1021/acs.jmedchem.9b02147.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: Synthetic Minority Over-sampling Technique." In: *Journal of Artificial Intelligence Research* 16 (June 2002), pp. 321–357. ISSN: 1076-9757. DOI: 10.1613/jair.953.

[16] Hongming Chen, Ola Engkvist, Yinhai Wang, Marcus Olivecrona, and Thomas Blaschke. "The rise of deep learning in drug discovery." In: *Drug Discovery Today* 23.6 (2018), pp. 1241–1250. ISSN: 1359-6446. DOI: https://doi.org/10.1016/j.drudis.2018.01.039.

[17] Artem Cherkasov, Eugene N. Muratov, Denis Fourches, Alexandre Varnek, Igor I. Baskin, Mark Cronin, John Dearden, Paola Gramatica, Yvonne C. Martin, Roberto Todeschini, Viviana Consonni, Victor E. Kuz'min, Richard Cramer, Romualdo Benigni, Chihae Yang, James Rathman, Lothar Terfloth, Johann Gasteiger, Ann Richard, and Alexander Tropsha. "QSAR modeling: where have you been? Where are you going to?" In: *Journal of medicinal chemistry* 57.12 (June 2014), pp. 4977–5010. ISSN: 1520-4804. DOI: 10.1021/jm4004285.

[18] Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. "ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction." In: *arXiv preprint arXiv:2010.09885* (2020). arXiv: 2010.09885 [cs.LG].

[19] Connor W. Coley, Regina Barzilay, William H. Green, Tommi S. Jaakkola, and Klavs F. Jensen. "Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction." In: *Journal of Chemical Information and Modeling* 57.8 (2017), pp. 1757–1772. DOI: 10.1021/acs.jcim.6b00601.

[20] G.W. Corder and D.I. Foreman. *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. Wiley, 2009. ISBN: 9780470454619.

[21] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[22] Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. "Molecular representations in AI-driven drug discovery: a review and practical guide." In: *Journal of Cheminformatics* 12.1 (Sept. 2020), p. 56. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00460-5.

[23] John S. Delaney. "ESOL: Estimating Aqueous Solubility Directly from Molecular Structure." In: *Journal of Chemical Information and Computer Sciences* 44.3 (2004). PMID: 15154768, pp. 1000–1005. DOI: 10.1021/ci034243x.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *arXiv preprint arXiv:1810.04805* (2019). arXiv: 1810.04805 [cs.CL].

[25] Ola Engkvist and Paul Wrede. "High-Throughput, In Silico Prediction of Aqueous Solubility Based on One- and Two-Dimensional Descriptors." In: *Journal of Chemical Information and Computer Sciences* 42.5 (2002), pp. 1247–1249. DOI: 10.1021/ci0202685.

[26] Benedek Fabian, Thomas Edlich, Héléna Gaspar, Marwin Segler, Joshua Meyers, Marco Fiscato, and Mohamed Ahmed. "Molecular representation learning with language models and domain-relevant auxiliary tasks." In: *arXiv preprint arXiv:2011.13230* (2020). arXiv: 2011.13230 [cs.LG].

[27] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. "Data Augmentation for Low-Resource Neural Machine Translation." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2017). DOI: 10.18653/v1/p17-2090.

[28] WA et .al Falcon. "PyTorch Lightning." In: *GitHub* 3 (2019). URL: https://github.com/PyTorchLightning/pytorch-lightning.

[29] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. Proceedings of Machine Learning Research. 13–15 May 2010, pp. 249–256.

[30] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[31] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning." In: (2008), pp. 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

[32] Haibo He and Edwardo A. Garcia. "Learning from Imbalanced Data." In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239.

[33] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. 2015. arXiv: 1502.01852 [cs.CV].

[35] Tim Head, MechCoder, Gilles Louppe, Iaroslav Shcherbatyi, fcharras, Zé Vinícius, cmmalone, Christopher Schröder, nel215, Nuno Campos, Todd Young, Stefano Cereda, Thomas Fan, rene-rex, Kejia (KJ) Shi, Justus Schwabedal, carlosdanielcsantos, Hvass-Labs, Mikhail Pak, SoManyUsernamesTaken, Fred Callaway, Loïc Estève, Lilian Besson, Mehdi Cherti, Karlson Pfannschmidt, Fabian Linzberger, Christophe Cauet, Anna Gut, Andreas Mueller, and Alexander Fabisch. *scikit-optimize/scikit-optimize: v0.5.2*. Version v0.5.2. Mar. 2018. DOI: 10.5281/zenodo.1207017.

[36] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification." In: *arXiv preprint arXiv:1801.06146* (2018). arXiv: 1801.06146 [cs.CL].

[37] Benlin Hu, Cheng Lei, Dong Wang, Shu Zhang, and Zhenyu Chen. "A Preliminary Study on Data Augmentation of Deep Learning for Image Classification." In: *arXiv preprint arXiv:1906.11887* (2019). arXiv: 1906.11887 [cs.CV].

[38] J. D. Hunter. "Matplotlib: A 2D graphics environment." In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[39] Sunghwan Kim, Jie Chen, Tiejun Cheng, Asta Gindulyte, Jia He, Siqian He, Qingliang Li, Benjamin A Shoemaker, Paul A Thiessen, Bo Yu, Leonid Zaslavsky, Jian Zhang, and Evan E Bolton. "PubChem in 2021: new data content and improved web interfaces." In: *Nucleic Acids Research* 49.D1 (Nov. 2020), pp. D1388–D1395. ISSN: 0305-1048. DOI: 10.1093/nar/gkaa971.

[40] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *arXiv preprint arXiv:1412.6980* (2017). arXiv: 1412.6980.

[41] Panagiotis-Christos Kotsias, Josep Arús-Pous, Hongming Chen, Ola Engkvist, Christian Tyrchan, and Esben Jannik Bjerrum. "Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks." In: *Nature Machine Intelligence* 2.5 (May 2020), pp. 254–265. ISSN: 2522-5839. DOI: 10.1038/s42256-020-0174-5.

[42] Youngchun Kwon, Dongseon Lee, Youn-Suk Choi, Kyoham Shin, and Seokho Kang. "Compressed graph representation for scalable molecular graph generation." In: *Journal of Cheminformatics* 12.1 (Sept. 2020), p. 58. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00463-2.

[43] Greg Landrum. *RDKit: Open-source cheminformatics*. URL: http://www.rdkit.org.

[44] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." In: *arXiv preprint arXiv:1910.13461* (2019). arXiv: 1910.13461 [cs.CL].

[45] Xinhao Li and Denis Fourches. "Inductive transfer learning for molecular activity prediction: Next - Gen QSAR Models with MolPMoFiT." In: *Journal of Cheminformatics* 12.1 (Apr. 2020), p. 27. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00430-x.

[46] Xinhao Li and Denis Fourches. *SMILES Pair Encoding: A Data-Driven Substructure Tokenization Algorithm for Deep Learning*. May 2020. DOI: 10.26434/chemrxiv.12339368.v1.

[47] Junshui Ma, Robert P. Sheridan, Andy Liaw, George E. Dahl, and Vladimir Svetnik. "Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships." In: *Journal of Chemical Information and Modeling* 55.2 (2015), pp. 263–274. DOI: 10.1021/ci500747n.

[48] Pierre Mahé, Liva Ralaivola, Véronique Stoven, and Jean-Philippe Vert. "The Pharmacophore Kernel for Virtual Screening with Support Vector Machines." In: *Journal of Chemical Information and Modeling* 46.5 (2006), pp. 2003–2014. DOI: 10.1021/ci060138m.

[49] Adam C. Mater and Michelle L. Coote. "Deep Learning in Chemistry." In: *Journal of Chemical Information and Modeling* 59.6 (2019), pp. 2545–2559. DOI: 10.1021/acs.jcim.9b00266.

[50] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. "DeepTox: Toxicity Prediction using Deep Learning." In: *Frontiers in Environmental Science* 3 (2016), p. 80. DOI: 10.3389/fenvs.2015.00080.

[51] David Mendez, Anna Gaulton, A Patrícia Bento, Jon Chambers, Marleen De Veij, Eloy Félix, María Paula Magariños, Juan F Mosquera, Prudence Mutowo, Michał Nowotka, María Gordillo-Marañón, Fiona Hunter, Laura Junco, Grace Mugumbate, Milagros Rodriguez-Lopez, Francis Atkinson, Nicolas Bosc, Chris J Radoux, Aldo Segura-Cabrera, Anne Hersey, and Andrew R Leach. "ChEMBL: towards direct deposition of bioassay data." In: *Nucleic Acids Research* 47.D1 (Nov. 2018), pp. D930–D940. ISSN: 0305-1048. DOI: 10.1093/nar/gky1075.

[52] David L. Mobley and J. Peter Guthrie. "FreeSolv: a database of experimental and calculated hydration free energies, with input files." In: *Journal of Computer-Aided Molecular Design* 28.7 (July 2014), pp. 711–720. ISSN: 1573-4951. DOI: 10.1007/s10822-014-9747-x.

[53] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines." In: (2010), pp. 807–814.

[54] A. Nicholls. "Confidence limits, error bars and method comparison in molecular modeling. Part 1: The calculation of confidence intervals." In: *Journal of Computer-Aided Molecular Design* 28.9 (Sept. 2014), pp. 887–918. ISSN: 1573-4951. DOI: 10.1007/s10822-014-9753-z.

[55] Ali Oskooei, Jannis Born, Matteo Manica, Vigneshwari Subramanian, Julio Sáez-Rodríguez, and María Rodríguez Martínez. *PaccMann: Prediction of anticancer compound sensitivity with multi-modal attention-based neural networks*. 2019. arXiv: 1811.06802 [cs.LG].

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[57] Josh Payne, Mario Srouji, Dian Ang Yap, and Vineet Kosaraju. "BERT Learns (and Teaches) Chemistry." In: *arXiv preprint arXiv:2007.16012* (2020). arXiv: 2007.16012 [q-bio.BM].

[58] K. Pearson. *Mathematical Contributions to the Theory of Evolution. XIII. On the Theory of Contingency and Its Relation to Association and Normal Correlation, Etc.* 1904.

[59] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[60] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." In: *arXiv preprint arXiv:1910.10683* (2020). arXiv: 1910.10683 [cs.LG].

[61] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN: 026218253X.

[62] Raquel Rodríguez-Pérez and Jürgen Bajorath. "Evaluation of multi-target deep neural network models for compound potency prediction under increasingly challenging test conditions." In: *Journal of Computer-Aided Molecular Design* 35.3 (Mar. 2021), pp. 285–295. ISSN: 1573-4951. DOI: 10.1007/s10822-021-00376-8.

[63] Raquel Rodríguez-Pérez, Martin Vogt, and Jürgen Bajorath. "Support Vector Machine Classification and Regression Prioritize Different Structural Features for Binary Compound Activity and Potency Value Prediction." In: *ACS Omega* 2.10 (2017), pp. 6371–6379. DOI: 10.1021/acsomega.7b01079.

[64] Raquel Rodríguez-Pérez, Martin Vogt, and Jürgen Bajorath. "Support Vector Machine Classification and Regression Prioritize Different Structural Features for Binary Compound Activity and Potency Value Prediction." In: *ACS Omega* 2.10 (2017), pp. 6371–6379. DOI: 10.1021/acsomega.7b01079.

[65] David Rogers and Mathew Hahn. "Extended-Connectivity Fingerprints." In: *Journal of Chemical Information and Modeling* 50.5 (2010), pp. 742–754. DOI: 10.1021/ci100050t.

[66] F. Rosenblatt. "Principles of Neurodynamics: Perceptions and the Theory of Brain Mechanisms." In: *American Journal of Psychology* 76 (1963), p. 705.

[67] Sebastian Ruder. "An overview of gradient descent optimization algorithms." In: *arXiv preprint arXiv:1609.04747* (2017). arXiv: 1609.04747.

[68] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." In: *Nature* 323.6088 (1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

[69] María Virginia Sabando, Ignacio Ponzoni, Evangelos E. Milios, and Axel J. Soto. *Using Molecular Embeddings in QSAR Modeling: Does it Make a Difference?* 2021. arXiv: 2104.02604 [q-bio.BM].

[70] Philippe Schwaller, Théophile Gaudin, Dávid Lányi, Costas Bekas, and Teodoro Laino. ""Found in Translation": predicting outcomes of complex organic chemistry reactions using neural sequence-to-sequence models." In: *Chem. Sci.* 9 (28 2018), pp. 6091–6098. DOI: 10.1039/C8SC02339E.

[71] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network." In: *arXiv preprint arXiv:1808.03314* (2021). arXiv: 1808.03314 [cs.LG].

[72] Leslie N. Smith. "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay." In: *arXiv preprint arXiv:1803.09820* (2018). arXiv: 1803.09820 [cs.LG].

[73] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms." In: *arXiv preprint arXiv:1206.2944* (2012). arXiv: 1206.2944 [stat.ML].

[74] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[75] Teague Sterling and John J. Irwin. "ZINC 15 – Ligand Discovery for Everyone." In: *Journal of Chemical Information and Modeling* 55.11 (2015). PMID: 26479676, pp. 2324–2337. DOI: 10.1021/acs.jcim.5b00559.

[76] Noé Sturm, Andreas Mayr, Thanh Le Van, Vladimir Chupakhin, Hugo Ceulemans, Joerg Wegner, Jose-Felipe Golib-Dzib, Nina Jeliazkova, Yves Vandriessche, Stanislav Böhm, Vojtech Cima, Jan Martinovic, Nigel Greene, Tom Vander Aa, Thomas J. Ashby, Sepp Hochreiter, Ola Engkvist, Günter Klambauer, and Hongming Chen. "Industry-scale application and evaluation of deep learning for drug target prediction." In: *Journal of Cheminformatics* 12.1 (Apr. 2020), p. 26. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00428-5.

[77] Jiangming Sun, Nina Jeliazkova, Vladimir Chupakhin, Jose-Felipe Golib-Dzib, Ola Engkvist, Lars Carlsson, Jörg Wegner, Hugo Ceulemans, Ivan Georgiev, Vedrin Jeliazkov, Nikolay Kochev, Thomas J. Ashby, and Hongming Chen. "ExCAPE-DB: an integrated large scale dataset facilitating Big Data analysis in chemogenomics." In: *Journal of Cheminformatics* 9.1 (Mar. 2017), p. 17. ISSN: 1758-2946. DOI: 10.1186/s13321-017-0203-5.

[78] Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. "Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling." In: *Journal of Chemical Information and Computer Sciences* 43.6 (2003), pp. 1947–1958. DOI: 10.1021/ci034160g.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need." In: *arXiv preprint arXiv:1706.03762* (2017). arXiv: 1706.03762 [cs.CL].

[80] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[81] Dennis D. Wackerly, William Mendenhall III, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. sixth edition. Duxbury Advanced Series, 2002.

[82] Sheng Wang, Yuzhi Guo, Yuhong Wang, Hongmao Sun, and Junzhou Huang. "SMILES-BERT: Large Scale Unsupervised Pre-Training for Molecular Property Prediction." In: (2019), pp. 429–436. DOI: 10.1145/3307339.3342186.

[83] Michael L. Waskom. "seaborn: statistical data visualization." In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021.

[84] Jonathan J. Webster and Chunyu Kit. "Tokenization as the Initial Phase in NLP." In: *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*. USA: Association for Computational Linguistics, 1992, pp. 1106–1110. DOI: 10.3115/992424.992434. URL: https://doi.org/10.3115/992424.992434.

[85] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules." In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), pp. 31–36. DOI: 10.1021/ci00057a005.

[86] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules." In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), pp. 31–36. DOI: 10.1021/ci00057a005.

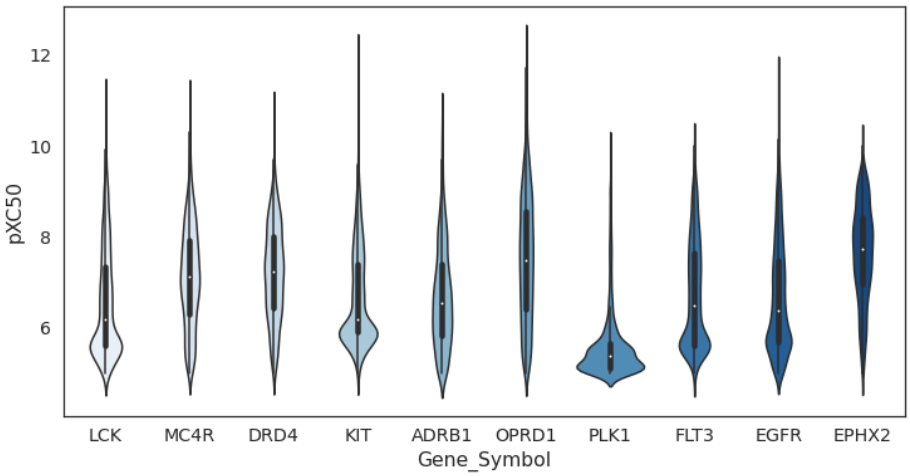
[87] Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. "MoleculeNet: A Benchmark for Molecular Machine Learning." In: *arXiv preprint arXiv:1703.00564* (2018). arXiv: 1703.00564 [cs.LG].

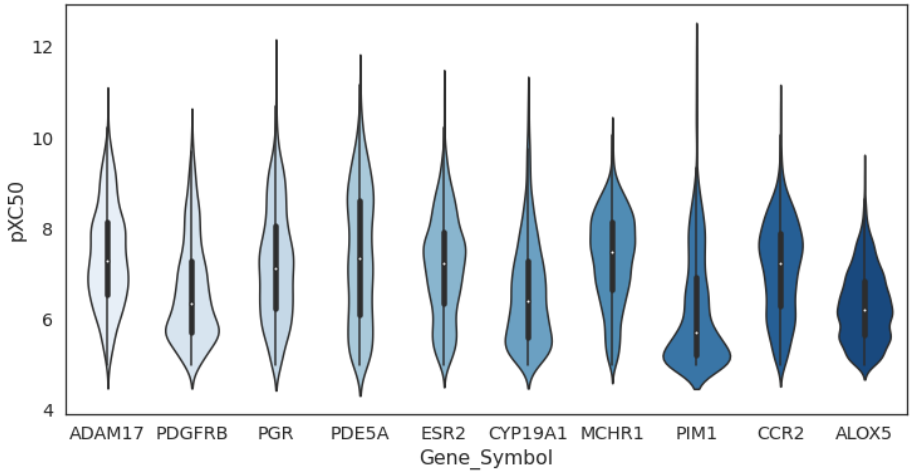
[88] Yadi Zhou, Suntara Cahya, Steven A. Combs, Christos A. Nicolaou, Jibo Wang, Prashant V. Desai, and Jie Shen. "Exploring Tunable Hyperparameters for Deep Neural Networks with Industrial ADME Data Sets." In: *Journal of Chemical Information and Modeling* 59.3 (2019), pp. 1005–1016. DOI: 10.1021/acs.jcim.8b00671.

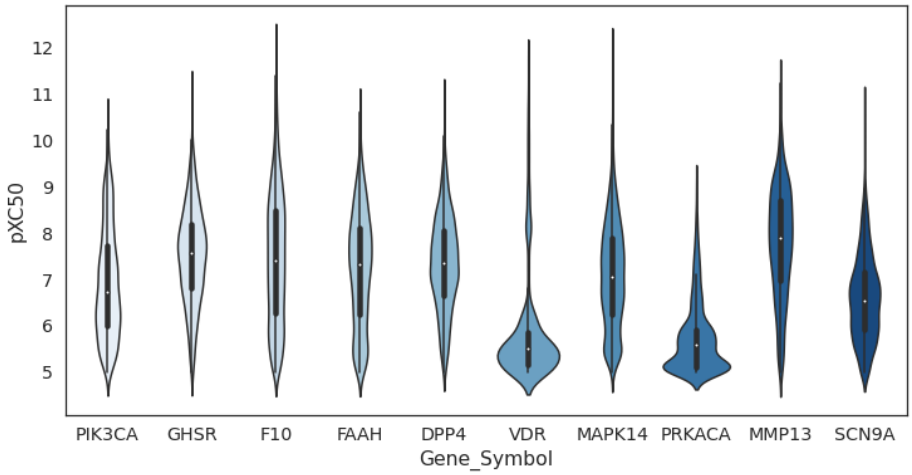
[89] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. "A Comprehensive Survey on Transfer Learning." In: *arXiv preprint arXiv:1911.02685* (2020). arXiv: 1911.02685 [cs.LG].

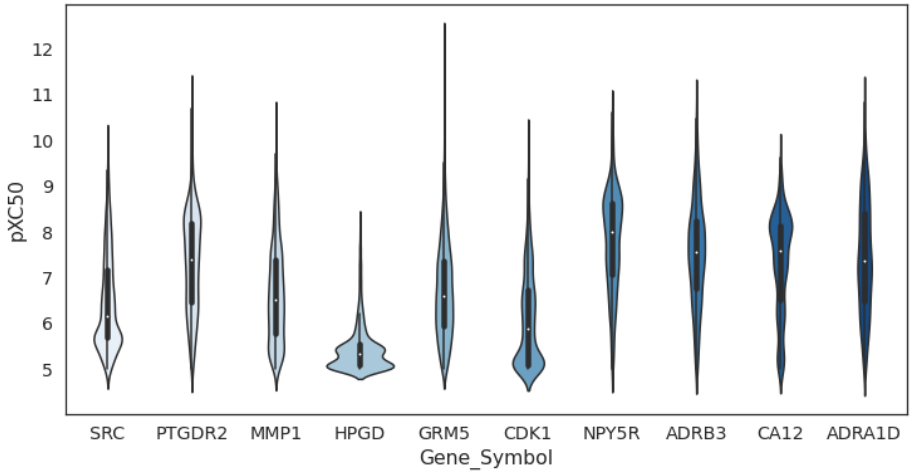
A Appendix

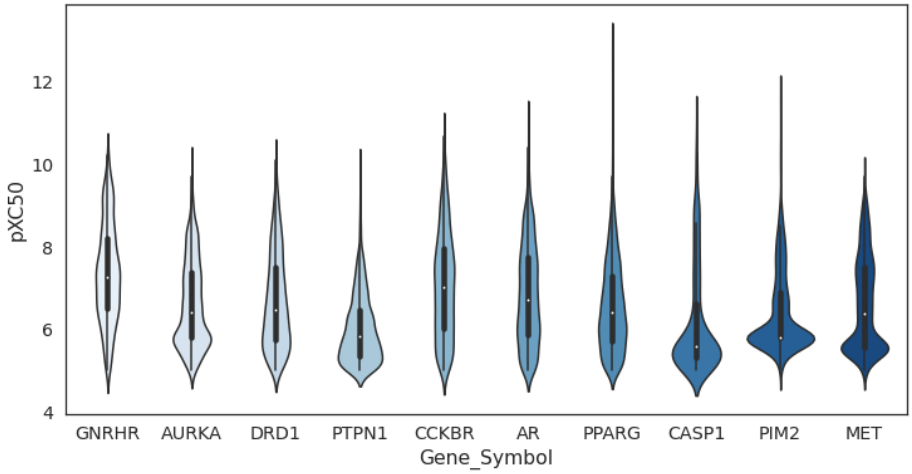
A.1 ExCAPE activity values distributions

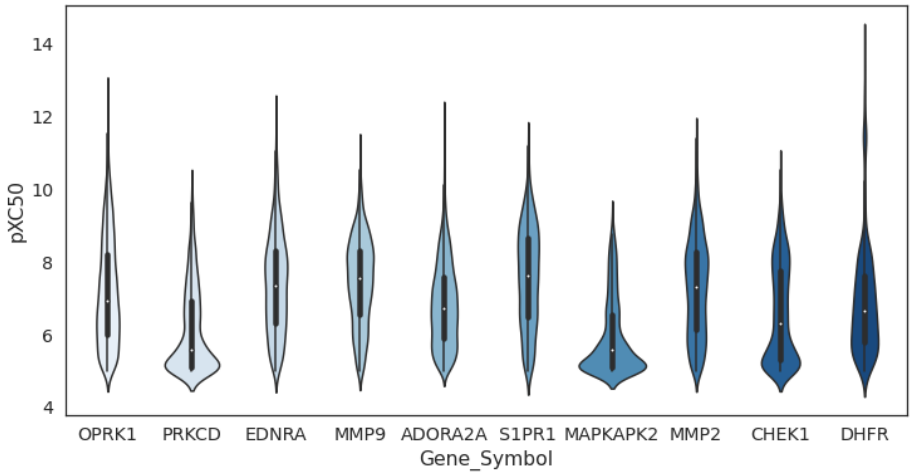


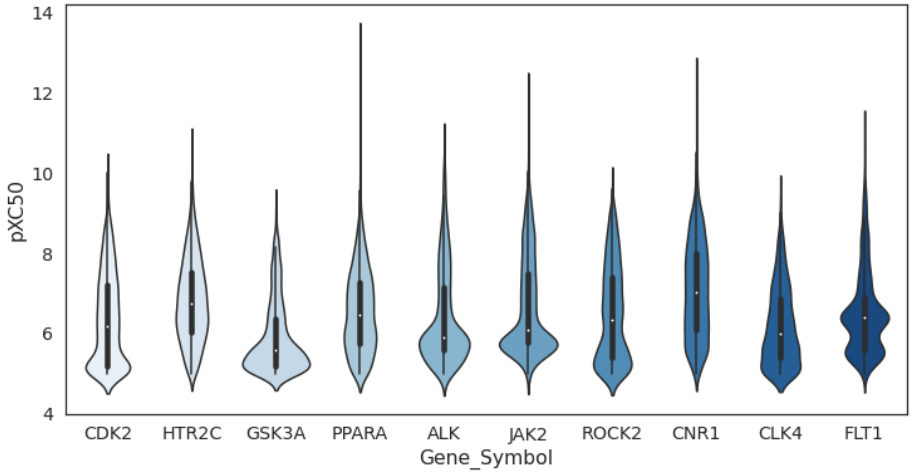

pXC50
12
10
8
6
4
ADAM17
PDGFRB
PGR
PDE5A
ESR2
CYP19A1
MCHR1
PIM1
CCR2
ALOX5
Gene_Symbol

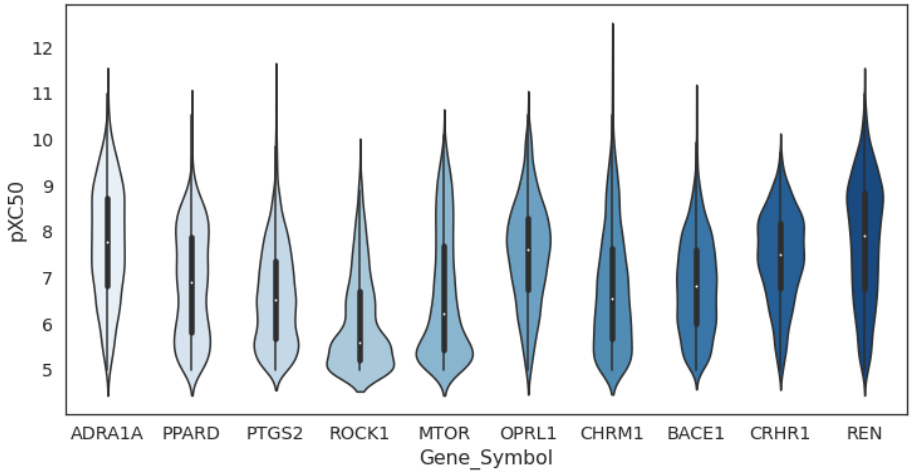

pXC50
12
11
10
9
8
7
6
5
PIK3CA
GHSR
F10
FAAH
DPP4
VDR
MAPK14
PRKACA
MMP13
SCN9A
Gene_Symbol

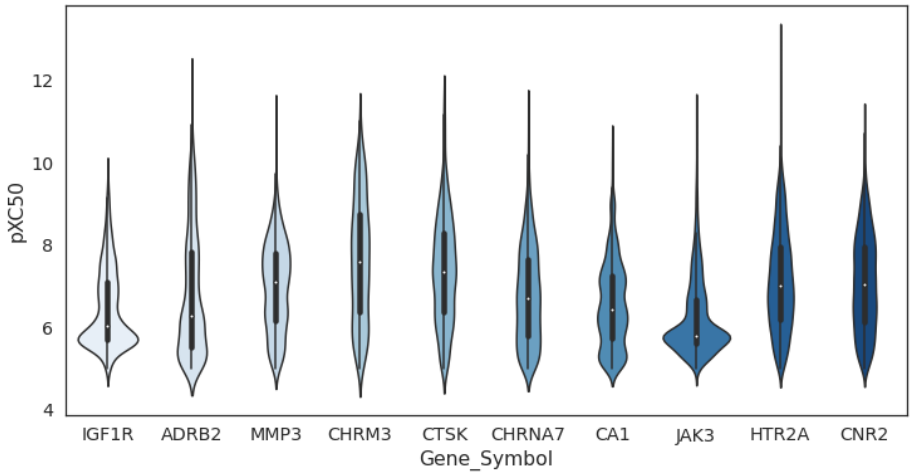

pXC50
12
11
10
9
8
7
6
5
SRC
PTGDR2
MMP1
HPGD
GRM5
CDK1
NPY5R
ADRB3
CA12
ADRA1D
Gene_Symbol

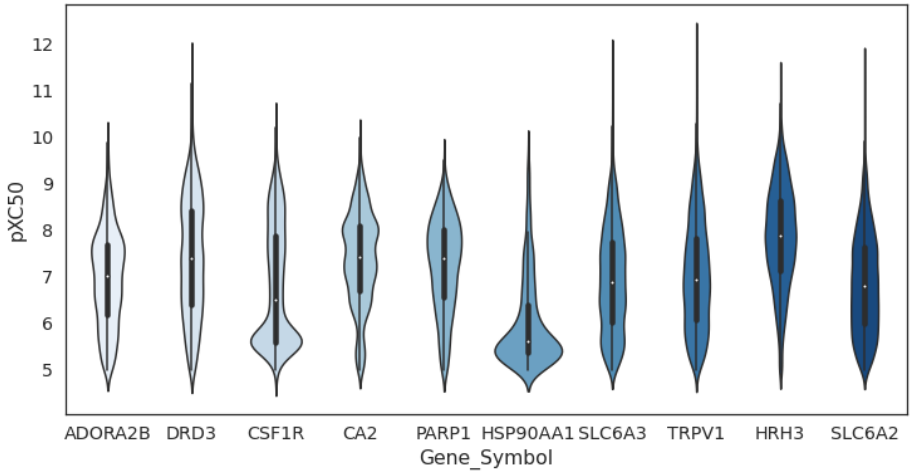

pXC50
12
10
8
6
4
GNRHR
AURKA
DRD1
PTPN1
CCKBR
AR
PPARG
CASP1
PIM2
MET
Gene_Symbol

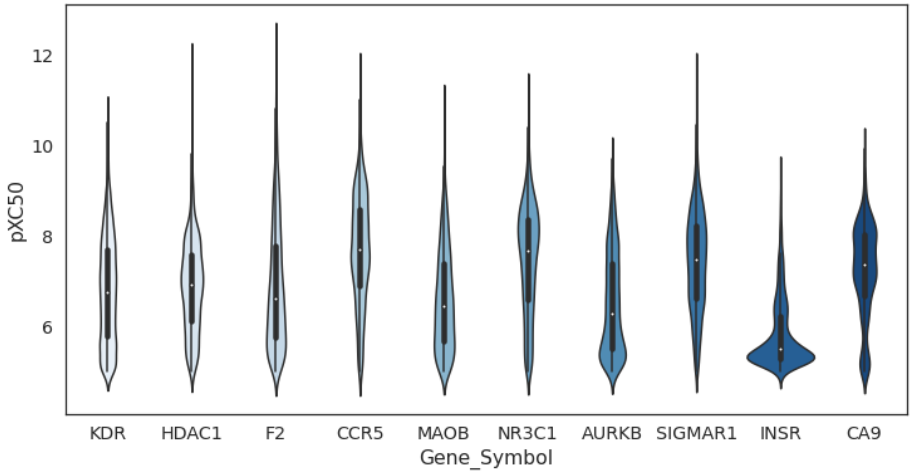

pXC50
14
12
10
8
6
4
OPRK1
PRKCD
EDNRA
MMP9
ADORA2A
S1PR1
MAPKAPK2
MMP2
CHEK1
DHFR
Gene_Symbol

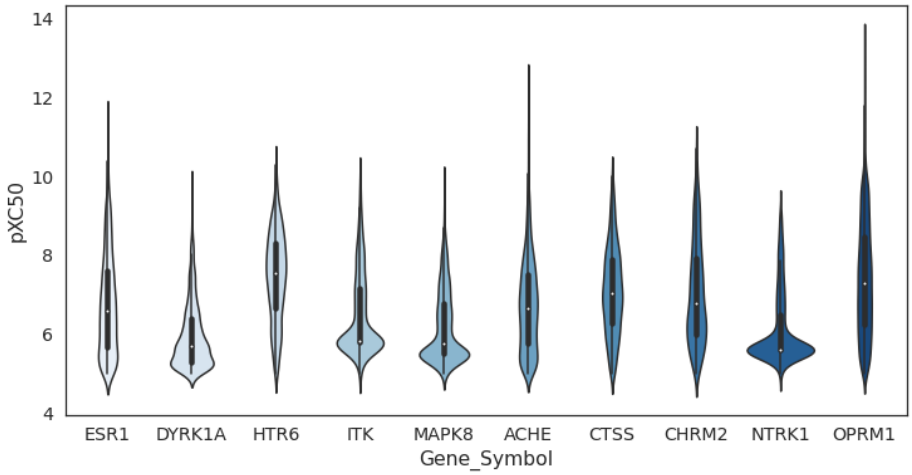

pXC50
14
12
10
8
6
4
CDK2
HTR2C
GSK3A
PPARA
ALK
JAK2
ROCK2
CNR1
CLK4
FLT1
Gene_Symbol

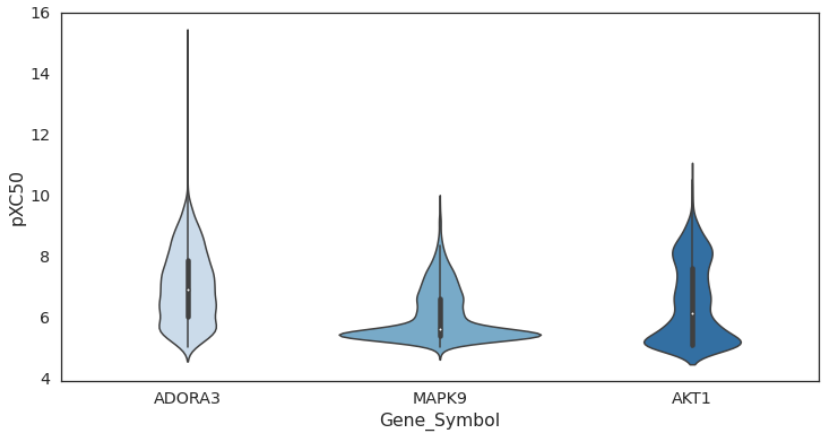

pXC50
12
11
10
9
8
7
6
5
ADRA1A
PPARD
PTGS2
ROCK1
MTOR
OPRL1
CHRM1
BACE1
CRHR1
REN
Gene_Symbol


pXC50
12
10
8
6
4
IGF1R
ADRB2
MMP3
CHRM3
CTSK
CHRNA7
CA1
JAK3
HTR2A
CNR2
Gene_Symbol


pXC50
12
11
10
9
8
7
6
5
ADORA2B
DRD3
CSF1R
CA2
PARP1
HSP90AA1
SLC6A3
TRPV1
HRH3
SLC6A2
Gene_Symbol


pXC50
12
10
8
6
KDR
HDAC1
F2
CCR5
MAOB
NR3C1
AURKB
SIGMAR1
INSR
CA9
Gene_Symbol


pXC50
14
12
10
8
6
4
ESR1
DYRK1A
HTR6
ITK
MAPK8
ACHE
CTSS
CHRM2
NTRK1
OPRM1
Gene_Symbol


pXC50
16
14
12
10
8
6
4
ADORA3
MAPK9
AKT1
Gene_Symbol

A.2 Python 3.7 used packages.

package	version
SciPy	1.4.1
matplotlib	3.3.4
seaborn	0.11.1
numpy	1.19.2
pandas	1.1.5
RDKit	2020.09.1
scikit-learn	0.22.2
scikit-optimize	0.8.1
PyTorch	1.8.0
PyTorch Lightning	1.2.3
cudatoolkit	10.1
tensorboard	2.4.1
Optuna	0.2.3

A.3 Hyperparameters.

A.3.1 Transformer models on Biological Activities, using multi-task simultaneous learning.

EncRegr model = {`d_model: 512, num_enc_layers: 2, num_heads: 32, d_FNN: 1024, dropout: 0.2, activation: 'relu', h_FNN: [1024, 512, 1], dropout: 0.7, lr: 0.0003, weight_decay: 0.000003`}

EncRegrTL ChEMBL = {`d_model: 256, num_enc_layers: 4, num_heads: 8, d_FNN: 2048, dropout: 0.15, activation: 'relu', h_FNN: [1024, 512, 1], dropout: 0.5, lr: 0.0003, weight_decay: 0.0`}

EncRegrTL ZINC = {`d_model: 512, num_enc_layers: 6, num_heads: 8, d_FNN: 2048, dropout: 0.13, activation: 'relu', h_FNN: [2048,1024, 1], dropout: 0.5, lr: 0.0006, weight_decay: 0.000004`}

A.3.2 SVR models on Biological Activities, using single task learning.

Gene Symbol	C	epsilon	gamma	Gene Symbol	C	epsilon	gamma
ABL1	5.803	0.143	0.859	DPP4	5.742	0.063	0.0
ACHE	5.236	0.094	1.0	DRD1	5.029	0.233	1.0
ADAM17	5.112	0.006	0.738	DRD3	5.761	0.109	0.619
ADORA2A	5.136	0.187	0.0	DRD4	5.619	0.069	0.0
ADORA2B	8.172	0.001	0.069	DYRK1A	7.136	0.134	0.414
ADORA3	5.713	0.062	0.982	EDNRA	5.037	0.081	0.077
ADRA1A	7.331	0.001	1.0	EGFR	8.742	0.099	0.463
ADRA1D	45.169	0.214	0.0	EPHX2	4.778	0.04	0.0
ADRB1	5.392	0.154	1.0	ERBB2	5.202	0.228	1.0
ADRB2	4.726	0.134	1.0	ESR1	5.505	0.052	1.0
ADRB3	5.727	0.001	0.099	ESR2	5.129	0.305	0.0
AKT1	5.144	0.001	0.0	F10	5.086	0.104	0.0
AKT2	5.728	0.001	1.0	F2	6.994	0.059	1.0
ALK	8.742	0.099	0.463	FAAH	4.962	0.201	0.0
ALOX5	5.62	0.124	1.0	FGFR1	4.063	0.017	1.0
AR	7.184	0.078	1.0	FLT1	5.296	0.065	0.908
AURKA	5.626	0.047	0.994	FLT3	5.66	0.001	1.0
AURKB	5.613	0.142	1.0	GHSR	4.395	0.001	1.0
BACE1	5.266	0.147	1.0	GNRHR	4.268	0.386	0.751
CA1	5.567	0.03	0.847	GRM5	5.832	0.096	0.0
CA12	6.427	0.001	1.0	GSK3A	20.519	0.103	1.0
CA2	5.711	0.004	0.414	GSK3B	5.483	0.014	0.0
CA9	6.109	0.001	0.0	HDAC1	1.752	0.001	1.0
CASP1	5.083	0.142	0.508	HPGD	13.944	0.13	0.0
CCKBR	4.778	0.173	0.853	HRH3	5.21	0.216	0.0
CCR2	5.01	0.251	0.0	HSD11B1	3.241	0.319	0.319
CCR5	5.356	0.107	0.993	HSP90AA1	47.185	0.059	0.617
CDK1	8.457	0.001	0.382	HTR2A	5.667	0.091	1.0
CDK2	5.986	0.096	0.976	HTR2C	4.291	0.093	0.0
CHEK1	6.979	0.001	0.534	HTR6	5.128	0.094	1.0
CHRM1	5.228	0.21	0.0	HTR7	8.742	0.099	0.463
CHRM2	5.804	0.029	1.0	IGF1R	5.617	0.14	0.0
CHRM3	4.232	0.001	1.0	INSR	8.139	0.001	0.755
CHRNA7	4.419	0.001	0.949	ITK	9.659	0.001	1.0
CLK4	5.491	0.187	0.071	JAK2	6.021	0.001	0.612
CNR1	5.312	0.048	1.0	JAK3	10.228	0.001	0.0
CNR2	5.045	0.067	1.0	KCNH2	8.742	0.099	0.463
CRHR1	5.416	0.001	0.0	KDR	5.307	0.001	0.883
CSF1R	5.271	0.125	1.0	KIT	7.08	0.001	0.0
CTSK	5.051	0.19	0.0	LCK	5.693	0.108	0.0
CTSS	5.196	0.107	1.0	MAOB	5.428	0.14	1.0
CYP19A1	6.207	0.186	1.0	MAPK14	5.884	0.066	1.0
DHFR	8.742	0.099	0.463	MAPK8	7.784	0.001	0.0

Gene Symbol	C	epsilon	gamma	Gene Symbol	C	epsilon	gamma
MAPK9	24.049	0.04	1.0	PIK3CA	5.401	0.097	1.0
MAPKAPK2	6.008	0.001	0.615	PIM1	12.667	0.051	1.0
MC4R	4.699	0.001	1.0	PIM2	10.887	0.001	1.0
MCHR1	8.742	0.099	0.463	PLK1	1.802	0.05	0.347
MET	4.772	0.001	1.0	PPARA	11.816	0.14	1.0
MMP1	4.924	0.001	0.0	PPARD	4.803	0.247	0.882
MMP13	3.166	0.001	0.0	PPARG	18.158	0.18	1.0
MMP2	5.953	0.056	1.0	PRKACA	3.267	0.097	0.0
MMP3	5.432	0.09	0.728	PRKCD	12.641	0.049	1.0
MMP9	5.003	0.107	1.0	PTGDR2	6.202	0.001	1.0
MTOR	5.068	0.149	1.0	PTGS2	5.822	0.14	0.0
NPY5R	5.033	0.001	1.0	PTPN1	10.713	0.023	0.518
NR3C1	4.762	0.108	1.0	REN	4.82	0.204	0.948
NTRK1	14.697	0.035	0.0	ROCK1	6.779	0.109	1.0
OPRD1	4.852	0.174	0.719	ROCK2	4.586	0.001	0.256
OPRK1	4.902	0.161	0.0	S1PR1	5.61	0.001	0.0
OPRL1	4.99	0.001	0.421	SCN9A	5.046	0.15	0.0
OPRM1	4.904	0.165	0.772	SIGMAR1	5.788	0.089	1.0
P2RX7	5.513	0.089	0.835	SLC6A2	5.232	0.205	0.0
PARP1	5.969	0.18	1.0	SLC6A3	6.534	0.174	1.0
PDE5A	5.224	0.001	0.874	SRC	3.912	0.097	0.359
PDGFRB	7.056	0.001	1.0	TACR1	5.702	0.123	0.0
PGR	9.205	0.001	0.0	TRPV1	4.973	0.001	0.0

A.3.3 Transformer models on Physical chemistry properties, using multi-task simultaneous learning.

EncRegr model = {`d_model: 256, num_enc_layers: 2, num_heads: 8, d_FNN: 1024, dropout: 0.15, activation: 'relu', h_FNN: [1024, 512, 1], dropout: 0.5, lr: 0.0003, weight_decay: 0.0`}

EncRegrTL ChEMBL = {`d_model: 256, num_enc_layers: 4, num_heads: 8, d_FNN: 1024, dropout: 0.15, activation: 'relu', h_FNN: [1024, 512, 1], dropout: 0.5, lr: 0.0003, weight_decay: 0.0`}

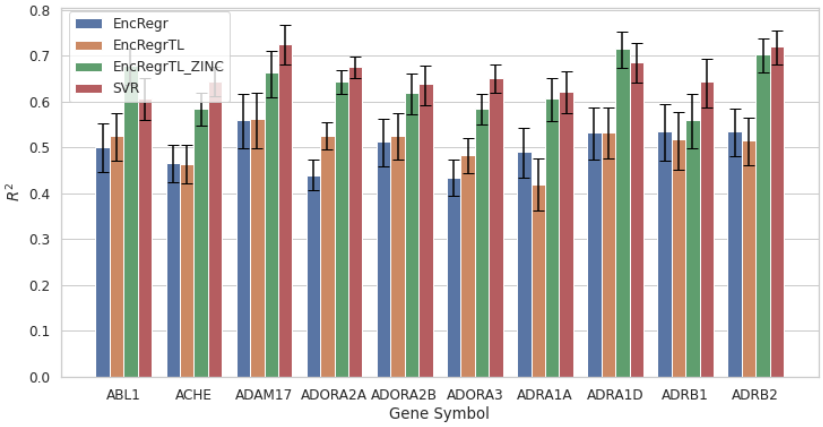
EncRegrTL ZINC = {`d_model: 512, num_enc_layers: 6, num_heads: 8, d_FNN: 2048, dropout: 0.13, activation: 'relu', h_FNN: [2048, 1024, 1], dropout: 0.5, lr: 0.0006, weight_decay: 0.000004`}

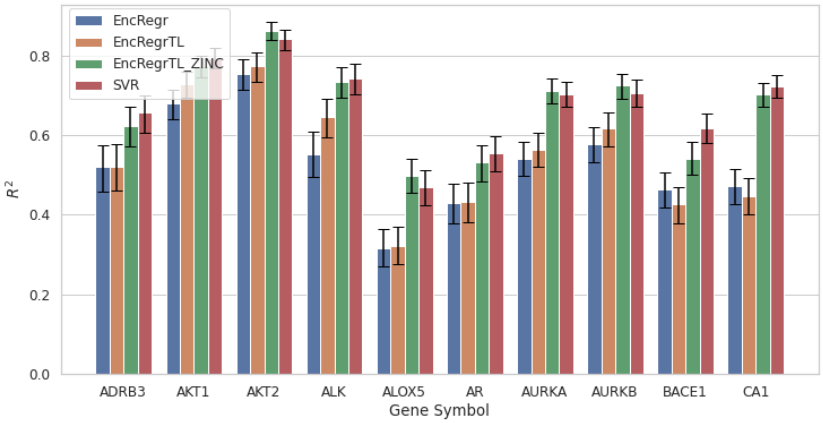
A.3.4 SVR models on Physical chemistry properties, using single task learning.

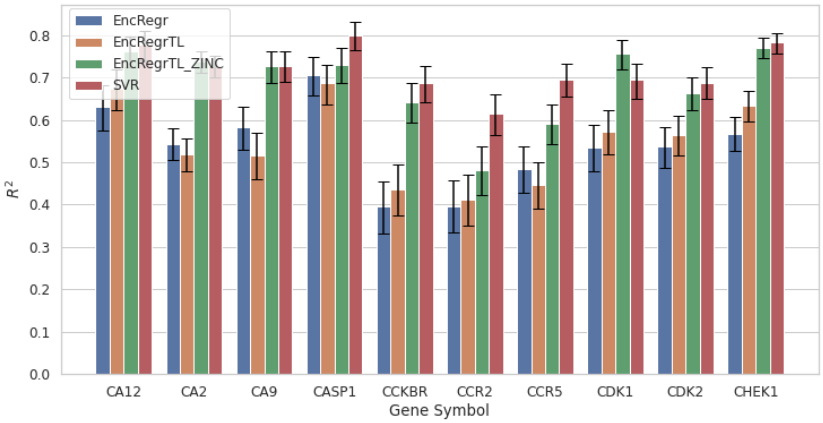
SVR on Lipophilicity = {`C: 58.2757, epsilon: 0.0016, gamma: 0.0792`}
SVR on ESOL = {`C: 0.5053, epsilon: 0.0017, gamma: 0.9089`}
SVR on FreeSolvation = {`C: 100.0, epsilon: 0.001, gamma: 1.0`}

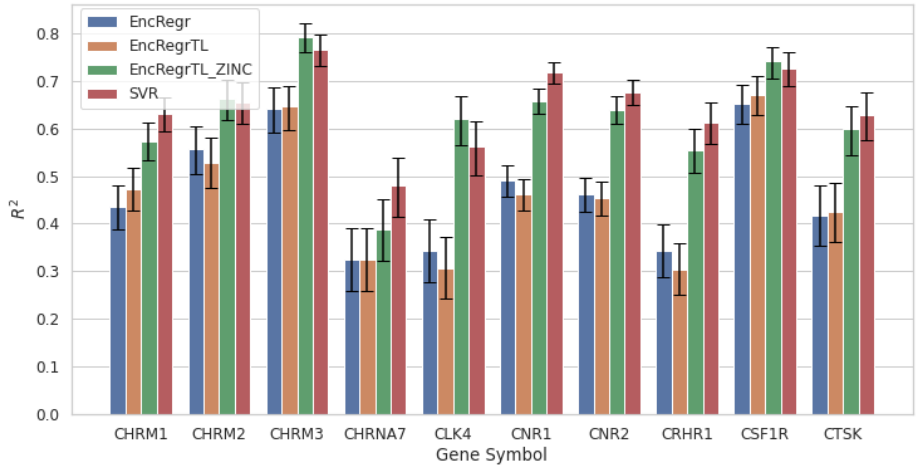
A.4 Biological Activities Additional Results.

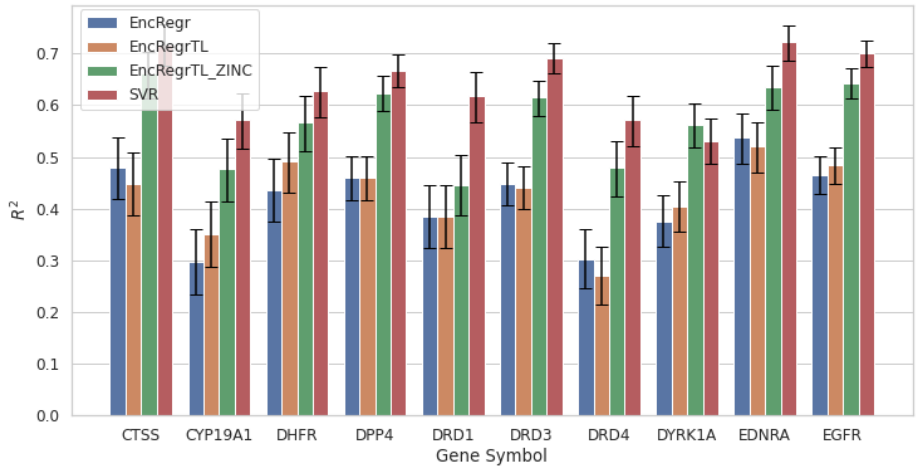
Bar charts with error bars of R^2 on test sets (using Fisher transformation on Pearson correlation for the error bars).

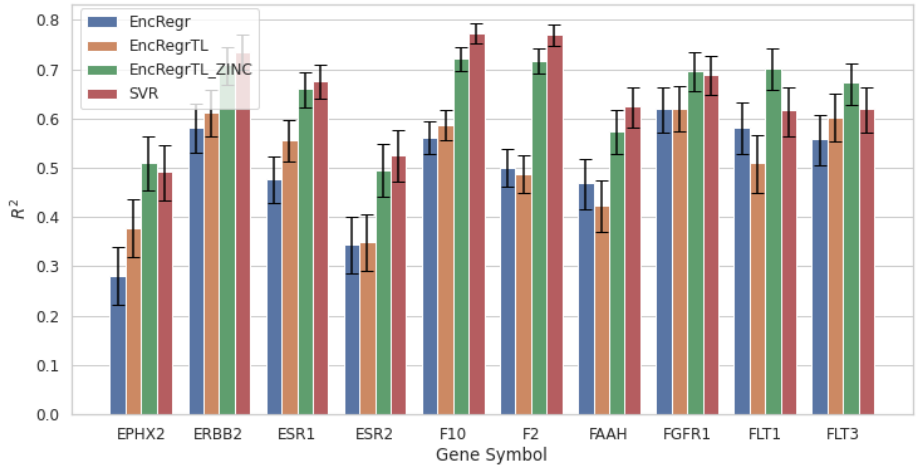


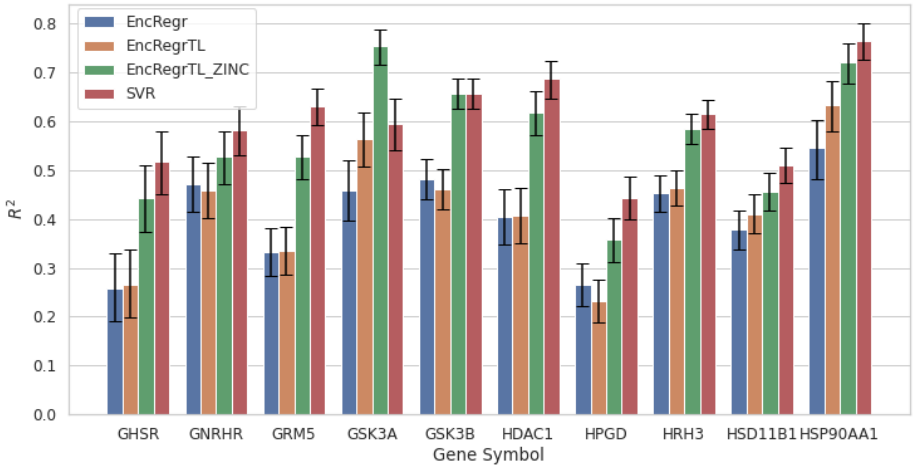


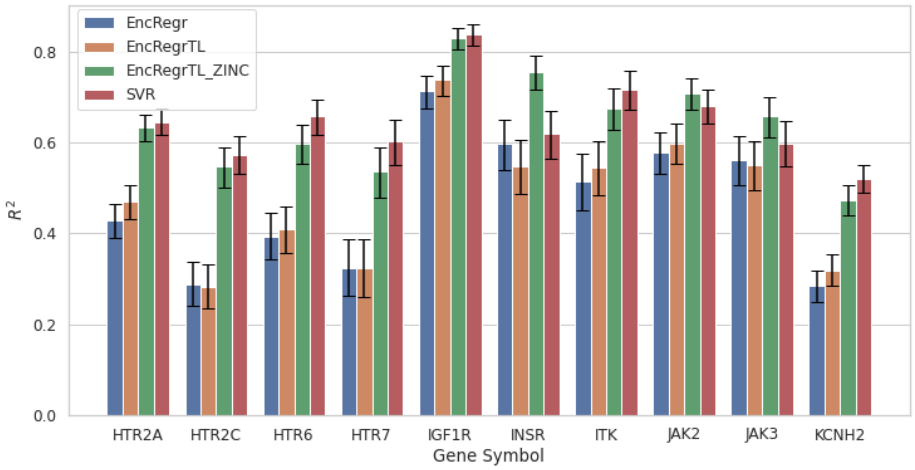


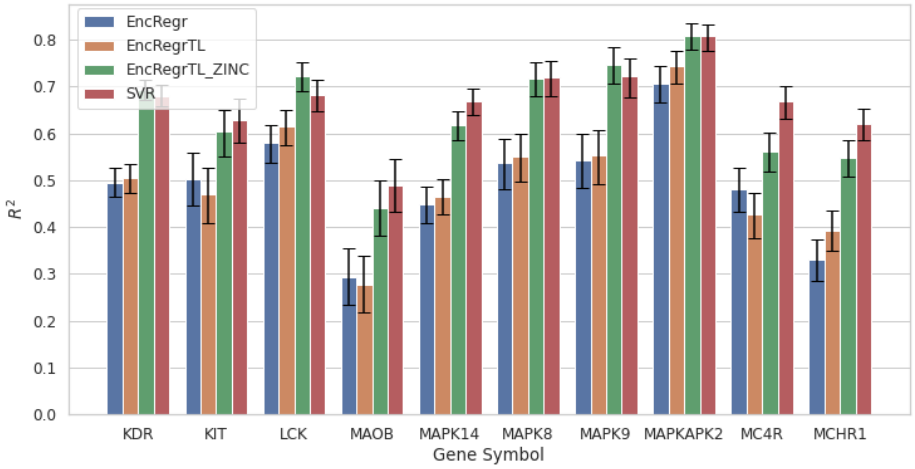

EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
CHRM1
CHRM2
CHRM3
CHRNA7
CLK4
CNR1
CNR2
CRHR1
CSF1R
CTSK
Gene Symbol

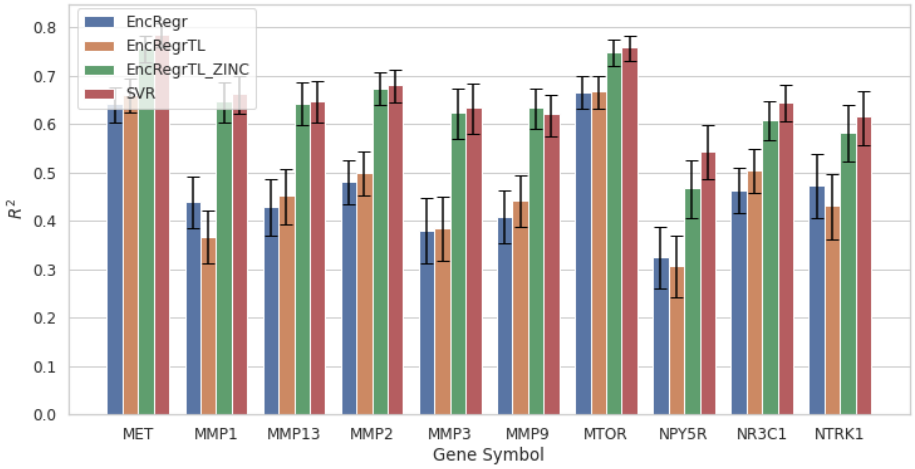

EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
CTSS
CYP19A1
DHFR
DPP4
DRD1
DRD3
DRD4
DYRK1A
EDNRA
EGFR
Gene Symbol

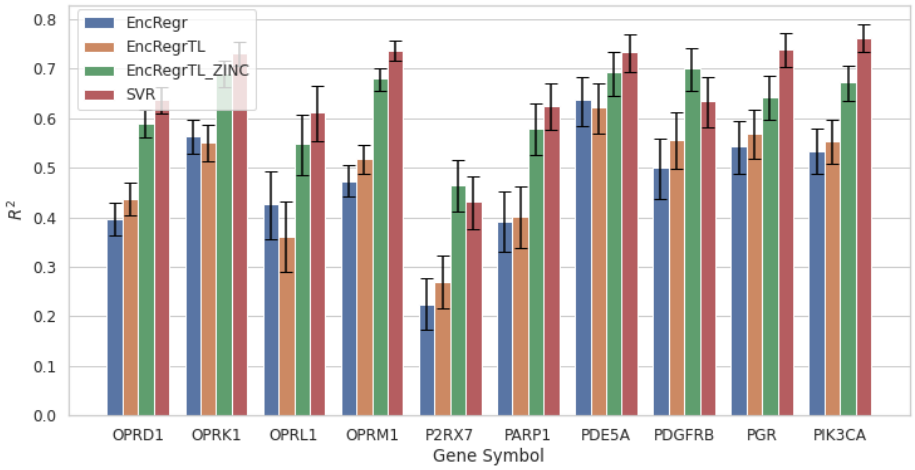

EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
EPHX2
ERBB2
ESR1
ESR2
F10
F2
FAAH
FGFR1
FLT1
FLT3
Gene Symbol

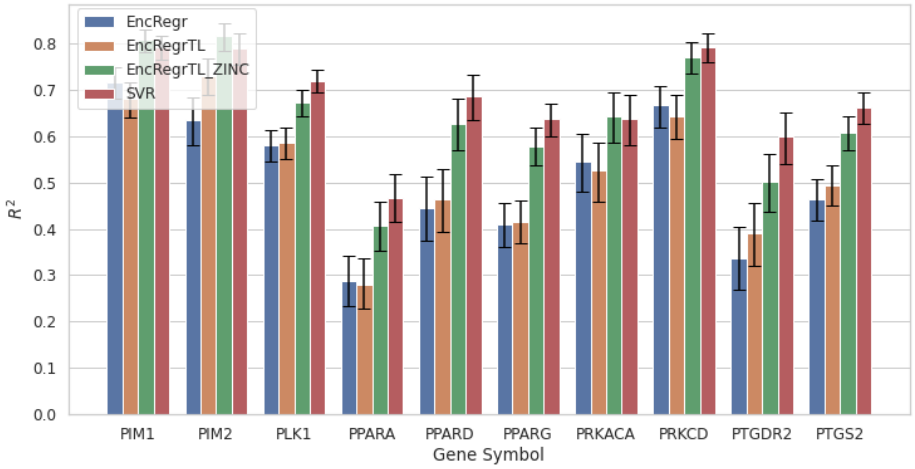

EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
GHSR
GNRHR
GRM5
GSK3A
GSK3B
HDAC1
HPGD
HRH3
HSD11B1
HSP90AA1
Gene Symbol

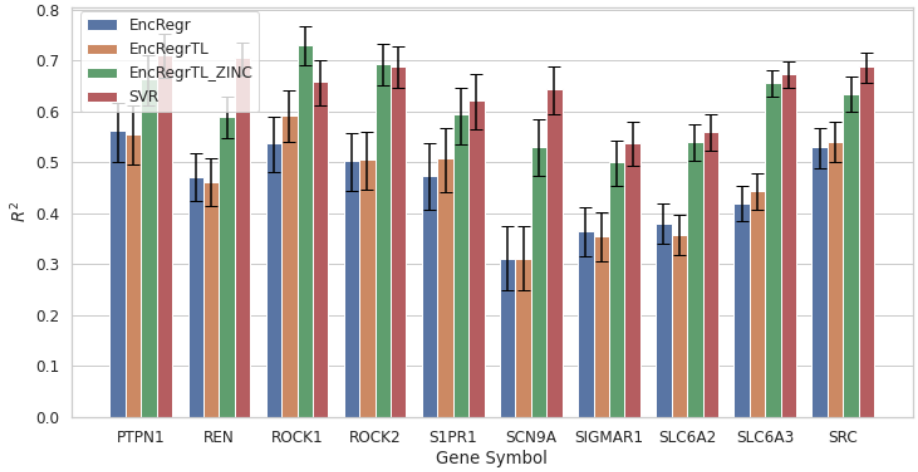

EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.2
0.4
0.6
0.8
HTR2A
HTR2C
HTR6
HTR7
IGF1R
INSR
ITK
JAK2
JAK3
KCNH2
Gene Symbol

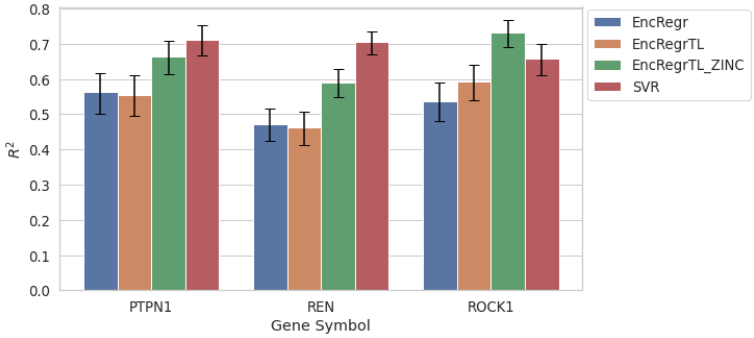

EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
KDR
KIT
LCK
MAOB
MAPK14
MAPK8
MAPK9
MAPKAPK2
MC4R
MCHR1
Gene Symbol


EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
MET
MMP1
MMP13
MMP2
MMP3
MMP9
MTOR
NPY5R
NR3C1
NTRK1
Gene Symbol


EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
OPRD1
OPRK1
OPRL1
OPRM1
P2RX7
PARP1
PDE5A
PDGFRB
PGR
PIK3CA
Gene Symbol


EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
PIM1
PIM2
PLK1
PPARA
PPARD
PPARG
PRKACA
PRKCD
PTGDR2
PTGS2
Gene Symbol


EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
PTPN1
REN
ROCK1
ROCK2
S1PR1
SCN9A
SIGMAR1
SLC6A2
SLC6A3
SRC
Gene Symbol


EncRegr
EncRegrTL
EncRegrTL_ZINC
SVR
R^2
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
PTPN1
REN
ROCK1
Gene Symbol

Results of the 133 SVR models on all 133 Gene Symbols.

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
ABL1	1858	0.1852	0.6146	0.9656	0.6066
ACHE	3151	0.3291	0.7002	0.9197	0.6442
ADAM17	1371	0.1695	0.5942	0.9764	0.7259
ADORA2A	5269	0.2966	0.6589	0.9337	0.6758
ADORA2B	2072	0.1861	0.6101	0.9667	0.6378
ADORA3	3810	0.3284	0.7	0.9241	0.6512
ADRA1A	1779	0.2648	0.747	0.9558	0.6224
ADRA1D	1458	0.2999	0.713	0.9399	0.6861
ADRB1	1297	0.2826	0.6773	0.9374	0.6428
ADRB2	1940	0.2869	0.7417	0.9643	0.7195
ADRB3	1455	0.2459	0.6632	0.9531	0.6566
AKT1	2229	0.1886	0.5967	0.9783	0.7947
AKT2	1368	0.1076	0.4103	0.9901	0.8424
ALK	1434	0.1166	0.5779	0.991	0.7434
ALOX5	2891	0.2667	0.5762	0.8914	0.47
AR	2571	0.3227	0.8262	0.9273	0.556
AURKA	2733	0.1302	0.5801	0.9846	0.7036
AURKB	2199	0.1773	0.6218	0.9779	0.7068
BACE1	2951	0.2891	0.6174	0.9268	0.6185
CA1	2858	0.223	0.5263	0.9556	0.7242
CA12	1299	0.087	0.5363	0.9935	0.7777
CA2	3652	0.1716	0.5224	0.9703	0.7279
CA9	1744	0.1496	0.5418	0.9791	0.7277
CASP1	1378	0.2056	0.5541	0.976	0.8007
CCKBR	1611	0.4226	0.7316	0.889	0.6876
CCR2	1689	0.3536	0.6664	0.8946	0.6138
CCR5	2016	0.2949	0.635	0.9415	0.6956
CDK1	1593	0.0667	0.5739	0.9955	0.6942
CDK2	2088	0.1715	0.6413	0.9788	0.6886
CHEK1	2718	0.125	0.6204	0.9914	0.7824
CHRM1	2757	0.3389	0.8009	0.9367	0.6308
CHRM2	1787	0.2897	0.7601	0.951	0.6555
CHRM3	1672	0.3544	0.7432	0.9427	0.7661
CHRNA7	1443	0.4417	0.8155	0.8607	0.479
CLK4	1492	0.1904	0.6718	0.9698	0.561
CNR1	5279	0.3031	0.6429	0.9346	0.7177
CNR2	4476	0.2832	0.6669	0.9425	0.6754
CRHR1	1999	0.1644	0.5968	0.9729	0.6124
CSF1R	1986	0.2566	0.6493	0.9595	0.7262
CTSK	1603	0.401	0.7755	0.9105	0.6273
CTSS	1570	0.2094	0.5904	0.9667	0.7191
CYP19A1	1622	0.3048	0.7492	0.9337	0.5717
DHFR	1516	0.4167	0.9038	0.9268	0.6282
DPP4	3293	0.2283	0.6093	0.9486	0.6681
DRD1	1732	0.389	0.7504	0.8851	0.6176
DRD3	3491	0.2955	0.6814	0.9454	0.6923
DRD4	1956	0.2363	0.6688	0.9515	0.571

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
DYRK1A	2568	0.1773	0.5582	0.9593	0.5313
EDNRA	2085	0.3109	0.6907	0.9477	0.7222
EGFR	4314	0.1976	0.6685	0.9738	0.7002
EPHX2	1792	0.3475	0.7486	0.8963	0.4914
ERBB2	1924	0.2292	0.5098	0.9499	0.736
ESR1	2591	0.1948	0.7234	0.9765	0.6757
ESR2	1946	0.3345	0.7635	0.9251	0.5254
F10	4637	0.2899	0.6779	0.957	0.7734
F2	3896	0.2484	0.6661	0.9667	0.7701
FAAH	2267	0.3829	0.7365	0.9051	0.6237
FGFR1	1807	0.1143	0.5209	0.9851	0.6887
FLT1	1643	0.1513	0.6398	0.9792	0.6163
FLT3	2000	0.1716	0.715	0.9782	0.6192
GHSR	1241	0.2923	0.6981	0.9217	0.5168
GNRHR	1720	0.5737	0.7623	0.7784	0.583
GRM5	2534	0.1785	0.6279	0.9695	0.6307
GSK3A	1606	0.1169	0.5604	0.9866	0.5957
GSK3B	3330	0.1373	0.607	0.9836	0.6573
HDAC1	1936	0.3255	0.5567	0.8949	0.6869
HPGD	3061	0.1203	0.4004	0.9534	0.4443
HRH3	4662	0.3555	0.6805	0.8945	0.6146
HSD11B1	3906	0.4581	0.7254	0.7961	0.5104
HSP90AA1	1368	0.0606	0.4612	0.9966	0.7659
HTR2A	4134	0.2854	0.7058	0.945	0.6456
HTR2C	2751	0.2539	0.6535	0.9427	0.5729
HTR6	2356	0.2144	0.6429	0.9659	0.6574
HTR7	1525	0.1948	0.6571	0.966	0.6025
IGF1R	2014	0.1628	0.3887	0.973	0.838
INSR	1381	0.0658	0.4552	0.9928	0.6195
ITK	1381	0.1108	0.5182	0.989	0.7175
JAK2	2146	0.1598	0.6664	0.9819	0.6805
JAK3	1665	0.1148	0.5436	0.9838	0.5985
KCNH2	5275	0.1389	0.5554	0.9726	0.5202
KDR	5749	0.1661	0.6634	0.9788	0.6806
KIT	1622	0.1394	0.6038	0.9821	0.6292
LCK	2652	0.1704	0.6464	0.9807	0.6816
MAOB	1688	0.32	0.7684	0.9203	0.4893
MAPK14	3947	0.1635	0.6232	0.9794	0.6694
MAPK8	1848	0.0992	0.5059	0.9875	0.7188
MAPK9	1503	0.0804	0.4418	0.9913	0.7212
MAPKAPK2	1620	0.0917	0.4549	0.9923	0.8071
MC4R	2516	0.3547	0.6331	0.8967	0.6674
MCHR1	3198	0.2151	0.6421	0.9558	0.6197
MET	2811	0.1365	0.5384	0.9852	0.7847
MMP1	2071	0.2423	0.5936	0.9468	0.6621
MMP13	1856	0.3288	0.696	0.9239	0.648
MMP2	2437	0.2466	0.7556	0.9666	0.68
MMP3	1566	0.2633	0.6584	0.9399	0.6335
MMP9	1999	0.2661	0.7403	0.9519	0.6201
MTOR	2907	0.204	0.69	0.9799	0.7582
NPY5R	1516	0.2445	0.7189	0.9481	0.5444
NR3C1	2522	0.3096	0.706	0.9329	0.6453
NTRK1	1296	0.0966	0.5234	0.9895	0.6148
OPRD1	5337	0.481	0.834	0.8773	0.6365
OPRK1	3667	0.3609	0.7419	0.9386	0.7308
OPRL1	1344	0.2575	0.7221	0.9462	0.6116

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
OPRM1	5830	0.4531	0.7321	0.8995	0.7366
P2RX7	2093	0.2596	0.6502	0.9046	0.4306
PARP1	1583	0.2391	0.6258	0.9454	0.6255
PDE5A	1467	0.3133	0.7719	0.9534	0.7336
PDGFRB	1509	0.1375	0.6976	0.9847	0.6348
PGR	1636	0.1287	0.6393	0.9891	0.7402
PIK3CA	2308	0.1645	0.5628	0.9822	0.7625
PIM1	2126	0.0736	0.5762	0.9965	0.7925
PIM2	1336	0.0536	0.4748	0.9969	0.7906
PLK1	3837	0.1505	0.3955	0.9589	0.7188
PPARA	2086	0.3282	0.7807	0.9059	0.4667
PPARD	1295	0.3077	0.6436	0.9361	0.6875
PPARG	2702	0.2262	0.6597	0.959	0.6362
PRKACA	1271	0.162	0.4709	0.9611	0.6382
PRKCD	1525	0.1204	0.5122	0.9897	0.793
PTGDR2	1375	0.2136	0.6618	0.96	0.5984
PTGS2	2861	0.2786	0.6385	0.9352	0.6621
PTPN1	1453	0.0491	0.4214	0.9966	0.7117
REN	2498	0.4253	0.7265	0.8993	0.705
ROCK1	1629	0.158	0.5947	0.9783	0.6584
ROCK2	1760	0.2421	0.6553	0.9594	0.6894
S1PR1	1361	0.2674	0.8355	0.9622	0.6216
SCN9A	1654	0.1791	0.5518	0.9644	0.6445
SIGMAR1	2886	0.347	0.7475	0.8992	0.5386
SLC6A2	3865	0.3369	0.7065	0.9008	0.5599
SLC6A3	5006	0.3152	0.6256	0.9217	0.6728
SRC	3086	0.2443	0.5726	0.9468	0.6875
TACR1	2412	0.3628	0.7191	0.9254	0.6925
TRPV1	2980	0.2943	0.6868	0.9348	0.6477
VDR	2585	0.2433	0.6164	0.958	0.7104

Results of the EncRegr model on 133 Gene Symbols.

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
ABL1	1858	0.4624	0.7596	0.6979	0.5006
ACHE	3151	0.6311	0.7054	0.8751	0.4655
ADAM17	1371	0.5681	0.7238	0.7549	0.5599
ADORA2A	5269	0.6326	0.6803	0.8797	0.4398
ADORA2B	2072	0.593	0.6577	0.7072	0.512
ADORA3	3810	0.6897	0.6596	0.9086	0.4352
ADRA1A	1779	0.6407	0.7458	0.8793	0.4906
ADRA1D	1458	0.5691	0.776	0.8852	0.5318
ADRB1	1297	0.5687	0.7416	0.7921	0.5354
ADRB2	1940	0.6236	0.8267	0.9912	0.5348
ADRB3	1455	0.5712	0.7505	0.8025	0.5197
AKT1	2229	0.529	0.8268	0.7503	0.6799
AKT2	1368	0.4053	0.861	0.513	0.7553
ALK	1434	0.4522	0.8608	0.7824	0.5533
ALOX5	2891	0.5262	0.5568	0.673	0.3171
AR	2571	0.6342	0.7146	0.9534	0.4285
AURKA	2733	0.5058	0.7625	0.7373	0.5418
AURKB	2199	0.5018	0.7925	0.7563	0.5784
BACE1	2951	0.5999	0.6722	0.7487	0.4636
CA1	2858	0.5966	0.6781	0.7428	0.4722
CA12	1299	0.5405	0.74	0.6894	0.6312
CA2	3652	0.5854	0.654	0.6844	0.5442
CA9	1744	0.5695	0.7024	0.6773	0.5818
CASP1	1378	0.4765	0.8677	0.6862	0.7058
CCKBR	1611	0.726	0.6645	1.0433	0.3945
CCR2	1689	0.6406	0.6291	0.8525	0.3967
CCR5	2016	0.6204	0.731	0.8462	0.4841
CDK1	1593	0.4832	0.7603	0.7126	0.5354
CDK2	2088	0.5168	0.7914	0.7975	0.5365
CHEK1	2718	0.5613	0.8268	0.8975	0.5681
CHRM1	2757	0.6491	0.7445	1.0197	0.4354
CHRM2	1787	0.6141	0.7798	0.8773	0.5557
CHRM3	1672	0.6168	0.8254	0.9366	0.6406
CHRNA7	1443	0.6825	0.6593	0.9392	0.3235
CLK4	1492	0.5585	0.6584	0.8471	0.3433
CNR1	5279	0.6641	0.687	0.8722	0.4905
CNR2	4476	0.6346	0.7059	0.8803	0.4615
CRHR1	1999	0.6443	0.5742	0.7901	0.343
CSF1R	1986	0.4874	0.8485	0.7435	0.6527
CTSK	1603	0.7177	0.7079	0.9955	0.4177
CTSS	1570	0.5602	0.7457	0.8204	0.4794
CYP19A1	1622	0.6466	0.674	1.0015	0.2963
DHFR	1516	0.8331	0.7078	1.1469	0.4364
DPP4	3293	0.61	0.6364	0.7886	0.4602
DRD1	1732	0.6692	0.6505	0.9629	0.3844
DRD3	3491	0.658	0.7228	0.9391	0.4479
DRD4	1956	0.6494	0.6131	0.8841	0.3017

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
DYRK1A	2568	0.5203	0.6042	0.6545	0.3759
EDNRA	2085	0.6998	0.731	0.9015	0.537
EGFR	4314	0.6436	0.7183	0.9094	0.4656
EPHX2	1792	0.6762	0.5984	0.8983	0.2811
ERBB2	1924	0.4618	0.7767	0.6485	0.582
ESR1	2591	0.64	0.7333	0.9378	0.4763
ESR2	1946	0.6423	0.6756	0.9361	0.3439
F10	4637	0.6557	0.776	0.9551	0.5619
F2	3896	0.683	0.7455	1.001	0.4998
FAAH	2267	0.6654	0.7048	0.9001	0.4684
FGFR1	1807	0.4094	0.8077	0.5882	0.6191
FLT1	1643	0.5051	0.7516	0.6713	0.5822
FLT3	2000	0.5149	0.8029	0.7823	0.5575
GHSR	1241	0.6456	0.6249	0.8973	0.2587
GNRHR	1720	0.7244	0.6351	0.8683	0.4717
GRM5	2534	0.6482	0.5757	0.8629	0.3325
GSK3A	1606	0.4354	0.7704	0.6779	0.4595
GSK3B	3330	0.5583	0.7222	0.7658	0.4821
HDAC1	1936	0.5423	0.702	0.8002	0.405
HPGD	3061	0.3884	0.3507	0.4656	0.2653
HRH3	4662	0.6421	0.645	0.822	0.4526
HSD11B1	3906	0.6356	0.5982	0.8339	0.3774
HSP90AA1	1368	0.4176	0.8143	0.6616	0.5452
HTR2A	4134	0.6565	0.6997	0.9287	0.4279
HTR2C	2751	0.6323	0.6214	0.883	0.288
HTR6	2356	0.6589	0.6624	0.8987	0.394
HTR7	1525	0.6264	0.6233	0.878	0.3245
IGF1R	2014	0.4442	0.792	0.527	0.7131
INSR	1381	0.3918	0.7535	0.4794	0.5961
ITK	1381	0.4595	0.8153	0.6985	0.5152
JAK2	2146	0.4972	0.8284	0.775	0.578
JAK3	1665	0.4245	0.7807	0.5758	0.5623
KCNH2	5275	0.5002	0.6052	0.6941	0.2842
KDR	5749	0.5665	0.7491	0.8459	0.4955
KIT	1622	0.5065	0.7669	0.7261	0.5027
LCK	2652	0.4961	0.8169	0.7547	0.5794
MAOB	1688	0.7038	0.5939	0.9372	0.2934
MAPK14	3947	0.5783	0.7302	0.8306	0.448
MAPK8	1848	0.4266	0.7676	0.6676	0.5362
MAPK9	1503	0.4069	0.7727	0.5784	0.5428
MAPKAPK2	1620	0.4161	0.8398	0.5644	0.7071
MC4R	2516	0.6757	0.6221	0.7975	0.4806
MCHR1	3198	0.6061	0.6367	0.8753	0.3291
MET	2811	0.4803	0.8159	0.7043	0.6408
MMP1	2071	0.595	0.6822	0.7757	0.4399
MMP13	1856	0.6192	0.7317	0.9152	0.428
MMP2	2437	0.6587	0.7595	0.9769	0.4805
MMP3	1566	0.6117	0.6653	0.8713	0.3807
MMP9	1999	0.656	0.6973	0.9512	0.4091
MTOR	2907	0.6247	0.7841	0.8173	0.6661
NPY5R	1516	0.6573	0.6227	0.8963	0.3241
NR3C1	2522	0.6311	0.7185	0.8867	0.4632
NTRK1	1296	0.3979	0.8178	0.6304	0.4735
OPRD1	5337	0.7962	0.6613	1.0986	0.3962
OPRK1	3667	0.6833	0.7708	0.9606	0.5631
OPRL1	1344	0.6291	0.6845	0.8901	0.4262

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
OPRM1	5830	0.7585	0.7145	1.057	0.4734
P2RX7	2093	0.5741	0.5233	0.7849	0.2233
PARP1	1583	0.5584	0.6844	0.8272	0.392
PDE5A	1467	0.6301	0.8144	0.9057	0.6365
PDGFRB	1509	0.497	0.7967	0.8259	0.4992
PGR	1636	0.591	0.7687	0.8636	0.5427
PIK3CA	2308	0.6006	0.7491	0.8017	0.5338
PIM1	2126	0.4666	0.8547	0.6777	0.7162
PIM2	1336	0.3625	0.8581	0.6377	0.6352
PLK1	3837	0.3671	0.7602	0.4931	0.5805
PPARA	2086	0.6653	0.5842	0.9102	0.2874
PPARD	1295	0.5936	0.7364	0.8858	0.4449
PPARG	2702	0.5983	0.696	0.857	0.4087
PRKACA	1271	0.4189	0.717	0.5396	0.5457
PRKCD	1525	0.4627	0.8411	0.6499	0.6666
PTGDR2	1375	0.6506	0.6207	0.8848	0.3372
PTGS2	2861	0.6587	0.6184	0.8156	0.4626
PTPN1	1453	0.406	0.7596	0.5288	0.5625
REN	2498	0.7527	0.6753	0.9775	0.4716
ROCK1	1629	0.4985	0.7682	0.6981	0.5375
ROCK2	1760	0.5551	0.7857	0.8508	0.5023
S1PR1	1361	0.6796	0.7544	1.0073	0.4738
SCN9A	1654	0.6598	0.4717	0.7861	0.3117
SIGMAR1	2886	0.6886	0.5955	0.8975	0.3647
SLC6A2	3865	0.6586	0.6127	0.8565	0.3803
SLC6A3	5006	0.66	0.6508	0.8496	0.4197
SRC	3086	0.5471	0.725	0.7118	0.5295
TACR1	2412	0.6687	0.7395	0.9236	0.5127
TRPV1	2980	0.6495	0.6804	0.883	0.4359
VDR	2585	0.5144	0.8011	0.6542	0.6832

Results of the EncRegr with TL (pretrained on 100M molecules from ZINC) on 133 Gene Symbols.

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
ABL1	1858	0.2063	0.952	0.5586	0.6784
ACHE	3151	0.3166	0.9258	0.7773	0.5844
ADAM17	1371	0.2488	0.9469	0.6635	0.6633
ADORA2A	5269	0.2866	0.9343	0.6943	0.6439
ADORA2B	2072	0.2602	0.9339	0.6364	0.6187
ADORA3	3810	0.3268	0.9234	0.7792	0.5843
ADRA1A	1779	0.3105	0.9401	0.7819	0.6055
ADRA1D	1458	0.3326	0.9235	0.6809	0.7158
ADRB1	1297	0.2824	0.9363	0.7721	0.5604
ADRB2	1940	0.28	0.965	0.7797	0.7021
ADRB3	1455	0.2696	0.9443	0.7056	0.6237
AKT1	2229	0.2511	0.961	0.6325	0.774
AKT2	1368	0.2077	0.9636	0.381	0.8629
ALK	1434	0.2097	0.97	0.5915	0.7342
ALOX5	2891	0.2727	0.8809	0.5675	0.4987
AR	2571	0.327	0.9241	0.8733	0.531
AURKA	2733	0.2219	0.9542	0.577	0.7127
AURKB	2199	0.2138	0.9623	0.6043	0.7247
BACE1	2951	0.3096	0.9126	0.6999	0.5421
CA1	2858	0.2417	0.9471	0.5514	0.7033
CA12	1299	0.2149	0.9589	0.5526	0.7631
CA2	3652	0.2304	0.9463	0.5186	0.7372
CA9	1744	0.2255	0.9535	0.5483	0.7262
CASP1	1378	0.2359	0.9676	0.6521	0.7311
CCKBR	1611	0.435	0.8797	0.7896	0.643
CCR2	1689	0.3346	0.8987	0.7948	0.4827
CCR5	2016	0.2999	0.937	0.7561	0.592
CDK1	1593	0.1991	0.9591	0.5129	0.7558
CDK2	2088	0.2086	0.9661	0.6765	0.6627
CHEK1	2718	0.2267	0.9717	0.6401	0.7713
CHRM1	2757	0.3008	0.9451	0.8832	0.5738
CHRM2	1787	0.3125	0.9429	0.7655	0.6624
CHRM3	1672	0.3177	0.9537	0.7067	0.7915
CHRNA7	1443	0.3923	0.8874	0.9307	0.3875
CLK4	1492	0.2252	0.9445	0.6304	0.6192
CNR1	5279	0.3295	0.9231	0.7186	0.6582
CNR2	4476	0.2994	0.9345	0.7148	0.6391
CRHR1	1999	0.2687	0.9259	0.6531	0.5548
CSF1R	1986	0.2659	0.9548	0.6408	0.7403
CTSK	1603	0.3592	0.926	0.8172	0.5977
CTSS	1570	0.2671	0.9421	0.6568	0.6608
CYP19A1	1622	0.2714	0.9426	0.8426	0.4764
DHFR	1516	0.4093	0.9294	1.0143	0.5678
DPP4	3293	0.2853	0.9201	0.6539	0.6239
DRD1	1732	0.3509	0.9036	0.9256	0.4463
DRD3	3491	0.3131	0.9373	0.771	0.6154
DRD4	1956	0.2875	0.9243	0.7613	0.4785

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
DYRK1A	2568	0.2263	0.9249	0.5517	0.5613
EDNRA	2085	0.3431	0.9353	0.8107	0.6359
EGFR	4314	0.265	0.9522	0.7409	0.6432
EPHX2	1792	0.3429	0.8964	0.745	0.5109
ERBB2	1924	0.2348	0.9419	0.5388	0.7088
ESR1	2591	0.255	0.9576	0.7497	0.6597
ESR2	1946	0.2841	0.9363	0.8109	0.4958
F10	4637	0.3125	0.9491	0.7595	0.721
F2	3896	0.3054	0.9491	0.7438	0.718
FAAH	2267	0.3541	0.9163	0.8117	0.5742
FGFR1	1807	0.1918	0.9579	0.5226	0.6968
FLT1	1643	0.2144	0.9552	0.5639	0.7027
FLT3	2000	0.2237	0.9627	0.671	0.6724
GHSR	1241	0.3402	0.8955	0.7569	0.4431
GNRHR	1720	0.4954	0.8293	0.8374	0.5271
GRM5	2534	0.2727	0.9249	0.7277	0.528
GSK3A	1606	0.1825	0.9596	0.4546	0.7541
GSK3B	3330	0.2254	0.9546	0.6145	0.6576
HDAC1	1936	0.2418	0.9405	0.6266	0.6191
HPGD	3061	0.1983	0.8299	0.4451	0.3569
HRH3	4662	0.3626	0.8866	0.7202	0.5852
HSD11B1	3906	0.401	0.8394	0.7826	0.4566
HSP90AA1	1368	0.1863	0.963	0.5098	0.7213
HTR2A	4134	0.2906	0.941	0.7356	0.6321
HTR2C	2751	0.2987	0.9153	0.6882	0.5463
HTR6	2356	0.298	0.9308	0.7115	0.5978
HTR7	1525	0.2486	0.9404	0.7191	0.5362
IGF1R	2014	0.208	0.9542	0.4028	0.8307
INSR	1381	0.1754	0.9507	0.3675	0.7558
ITK	1381	0.193	0.9672	0.5656	0.6759
JAK2	2146	0.2348	0.9616	0.6371	0.7075
JAK3	1665	0.2137	0.9444	0.5064	0.6576
KCNH2	5275	0.2212	0.9227	0.5966	0.4733
KDR	5749	0.2418	0.9543	0.6526	0.694
KIT	1622	0.1917	0.9664	0.6488	0.6031
LCK	2652	0.2149	0.9656	0.6034	0.7226
MAOB	1688	0.286	0.9329	0.8341	0.4411
MAPK14	3947	0.2584	0.946	0.6859	0.6173
MAPK8	1848	0.1865	0.9553	0.509	0.7175
MAPK9	1503	0.1804	0.9549	0.4244	0.7474
MAPKAPK2	1620	0.1991	0.9635	0.4682	0.8082
MC4R	2516	0.3799	0.8803	0.7412	0.5607
MCHR1	3198	0.2837	0.9204	0.7138	0.547
MET	2811	0.2112	0.9642	0.5778	0.7558
MMP1	2071	0.2567	0.9409	0.6203	0.6461
MMP13	1856	0.2862	0.9426	0.7232	0.6432
MMP2	2437	0.2699	0.9596	0.7768	0.6742
MMP3	1566	0.2664	0.9364	0.6704	0.6231
MMP9	1999	0.2633	0.9512	0.7417	0.6329
MTOR	2907	0.23	0.9708	0.711	0.7483
NPY5R	1516	0.261	0.9405	0.811	0.4672
NR3C1	2522	0.3353	0.9204	0.7504	0.608
NTRK1	1296	0.159	0.9709	0.5607	0.5833
OPRD1	5337	0.4608	0.8865	0.9038	0.5898
OPRK1	3667	0.3267	0.9476	0.8054	0.6908
OPRL1	1344	0.2875	0.9339	0.7836	0.5494

Gene Symbol	N mols	RMSE train	RMSE test	R2 train	R2 test
OPRM1	5830	0.4355	0.9058	0.8162	0.6796
P2RX7	2093	0.3049	0.865	0.6369	0.4639
PARP1	1583	0.2828	0.9184	0.6726	0.5791
PDE5A	1467	0.3415	0.9455	0.8448	0.6922
PDGFRB	1509	0.2192	0.9606	0.6339	0.7017
PGR	1636	0.2735	0.9504	0.7654	0.6422
PIK3CA	2308	0.2344	0.9617	0.6683	0.6724
PIM1	2126	0.2163	0.969	0.5556	0.8075
PIM2	1336	0.172	0.9677	0.4495	0.8153
PLK1	3837	0.1943	0.9326	0.4346	0.6726
PPARA	2086	0.3519	0.8836	0.84	0.4066
PPARD	1295	0.2918	0.9363	0.7132	0.6276
PPARG	2702	0.273	0.9367	0.7296	0.5789
PRKACA	1271	0.1906	0.9407	0.4713	0.6436
PRKCD	1525	0.2151	0.9656	0.5438	0.7712
PTGDR2	1375	0.2813	0.929	0.7575	0.5017
PTGS2	2861	0.288	0.9269	0.7096	0.6077
PTPN1	1453	0.2032	0.9401	0.4695	0.6639
REN	2498	0.4272	0.8955	0.8677	0.5906
ROCK1	1629	0.2151	0.9574	0.5255	0.7316
ROCK2	1760	0.2759	0.947	0.6592	0.6936
S1PR1	1361	0.309	0.9491	0.8775	0.5937
SCN9A	1654	0.2642	0.9148	0.6478	0.5317
SIGMAR1	2886	0.3403	0.9009	0.8045	0.5005
SLC6A2	3865	0.3042	0.9171	0.7355	0.5404
SLC6A3	5006	0.3007	0.9274	0.6486	0.6568
SRC	3086	0.2731	0.9313	0.6273	0.635
TACR1	2412	0.3533	0.9273	0.8349	0.6047
TRPV1	2980	0.315	0.9248	0.8002	0.552
VDR	2585	0.2433	0.9556	0.6491	0.6929