Diploma Thesis

Prediction of pKa values of small molecules via graph neural networks

submitted in satisfaction of the requirements for the degree of
Diplom-Ingenieur
of the TU Wien, Faculty of Chemistry

Diplomarbeit

Vorhersage der pKa-Werte organischer Moleküle mittels Graph Neuronaler Netze

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Diplom-Ingenieurs
eingereicht an der Technischen Universität Wien, Fakultät für Chemie

von

Fritz Mayr, BSc

Matr.Nr.: 01425153

unter der Anleitung von

[1]Univ.-Prof. Mag. Dr. **Thierry Langer**
[1]Dr. **Marcus Wieder**, MSc., MSc.
[2]Univ.Prof. Dipl.-Ing. Dr.techn. **Ruth Birner-Grünberger**

[1] Institut für Pharmaceutische Wissenschaften,
Universität Wien
Althanstrasse 14 (UZA II)
1090 Wien, Österreich

[2] Institut für Chemische Technologien und Analytik,
Technische Universität Wien
Getreidemarkt 9, 1060 Wien, Österreich
1060 Wien, Österreich

Wien, im Juli 2021

Danksagungen

Ich danke Herrn Prof. Thierry Langer und meinem Betreuer, Herrn Dr. Marcus Wieder, für die erstklassige Unterstützung während dieser Arbeit. Stellvertretend für die Forschungsgruppe möchte ich mich ganz besonders bei Steffen Hirte bedanken, welcher mir vom ersten Tag an bei unzähligen informatischen Fragestellungen mit Rat und Tat zur Seite stand. Einen speziellen Dank schulde ich Frau Prof. Birner-Grünberger, welche mir einen großen Gefallen tat, indem sie die Betreuung und Benotung dieser Arbeit seitens der TU übernahm.
Meiner Freundin Veronika bin ich sehr dankbar für den liebevollen Beistand, vor allem wenn die Programmierungen einfach nicht klappen wollten und zu frustrieren begannen.
Der größte Dank gebührt meinen Eltern, Elfi und Thomas, die mich durch ihre Liebe und Unterstützung dahin brachten, wo ich heute stehe und mir ein wirklich sorgenfreies Studium ermöglichten.

Kurzfassung

pK_a-Werte spielen im Bereich der molekularen Modellierung eine große Rolle, da sie die Ladung, Tautomer-Konfiguration und allgemeine 3D-Struktur eines Moleküls in der physiologischen Umgebung beeinflussen. All diese Faktoren prägen weiters die Mobilität, Permeabilität, Stabilität und Wirkweise der Substanzen im Körper. Bei unzureichenden bzw. fehlenden empirischen Messdaten ist die korrekte Bestimmung von pK_a-Werten somit essentiell, um die genannten Moleküleigenschaften korrekt vorhersagen zu können. Die vorliegende Arbeit geht von den Datensätzen und Modellen der Publikation *Machine learning meets pKa* von Baltruschat et al.[1] aus, deren relevante Ergebnisse reproduziert und mittels auf Graph neuronalen Netzen basierenden Modellen sogar substanziell verbessert wurden. Die Arbeit wurde in der Programmiersprache *Python* verfasst und die Funktionen bzw. Prozessskripten wurden in Form des eigens erstellten Paketes *pkasolver* (`https://github.com/MayrF/pkasolver`) veröffentlicht und frei zugänglich gemacht.

Abstract

pK_a values play a major role in the field of molecular modelling, as they influence the charge, tautomer configuration, and overall 3D structure of a molecule in the physiological environment. All these factors further shape the mobility, permeability, stability and mode of action of substances in the body. In case of insufficient or missing empirical data, the correct determination of pK_a values is thus essential to correctly predict the aforementioned molecular properties. The present work is based on the data sets and models of the publication *Machine learning meets pKa* by Baltruschat et al.[1], the relevant results of which were reproduced and even substantially improved upon by using models based on graph neural networks. The work was written in the programming language *Python*, the functions and process scripts have been published and made freely available in the form of the custom package *pkasolver* (`https://github.com/MayrF/pkasolver`).

Contents

Chapter 1

Theory

1.1 Introduction to pK_a

1.1.1 Definition of pK_a

Acid-base reactions in their most general form are described by the equation $HA+Y \rightleftarrows A^- + HY^+$, where HA represents the Brønsted-Acid and A^- its conjugate base, while Y and HY^+ stand for any protonatable solvent and its protonated form, respectively. In case of the most common solvent, water, the equation becomes $HA + H_2O \rightleftarrows A^- + H_3O^+$. The distributions of the four different species, involved in any acid-base reaction in water are described by the law of mass action, which states that

$$K = \prod_{i=1}^{z} a_i^{\nu_i} \tag{1.1}$$

where a_i is the activity of every reactant i and v_i its respective stoichiometric coefficient. For acid-base reactions in water then results in the following equation:

$$K = \frac{{a_{A^-}}^{\nu_{A^-}} \cdot {a_{H_3O^+}}^{\nu_{H_3O^+}}}{{a_{HA}}^{\nu_{HA}} \cdot {a_{H_2O}}^{\nu_{H_2O}}} \tag{1.2}$$

K is a dimensionless thermodynamic constant which is described by

$$K = \exp\left[-\frac{\Delta_R G^\circ}{R \cdot T}\right] \tag{1.3}$$

where $\Delta_R G^\circ$ is the molar Gibbs free energy of the reaction at a given temperature and pressure (usually *Standard Ambient Temperature and Pressure*, $T = 298.15\,\mathrm{K}$ or $25\,^\circ\mathrm{C}$, $\mathrm{p} = 101\,300\,\mathrm{Pa} = 1013\,\mathrm{hPa} = 101.3\,\mathrm{kPa} = 1.013\,\mathrm{bar}$), R is the gas constant and T the temperature. We see therefore that the reaction constant K for any reaction is independent of the distributions of compounds and only dependent on temperature and pressure. For sufficiently diluted solutions the activities can be approximated as concentration. Also, the concentration of water can be considered constant and can therefore be multiplied with each side, resulting in the formula for the acid dissociation constant K_a:

$$K_a = \frac{[A^-]\,[H_3O^+]}{[HA]} \tag{1.4}$$

to which applying the negative common logarithm yields the formula for the pK_a:

$$pK_a = -\log_{10} K_a = \log_{10} \frac{[HA]}{[A^-]\,[H_3O^+]} \tag{1.5}$$

We can see that the further the equilibrium lies on the side of the conjugated anion and H_3O^+, the lower the pK_a becomes. Also, when substituting $-\log_{10}[H_3O^+]$ with the pH, we get:

$$pH = pK_a + \log_{10}\frac{[A^-]}{[HA]} \tag{1.6}$$

It is observable that pK_a represents the pH at which protonated and deprotonated forms have the same concentration because $\log_{10} 1 = 0$. A pH $> pK_a$, therefore, leads to the deprotonated form dominating, while pH $< pK_a$ will result in the protonated form representing the majority of the molecules present in the described system.

1.1.2 Molecular influences on the pK_a

As indicated above, the pK_a value and hence the resulting acidity of any acid-base reaction depends on the difference in $\Delta_R G°$, the Gibbs free energy, between the protonated and the deprotonated form of the reactant. The pK_a decreases the higher the free energy of the acid and the lower the free energy of the conjugate base, while free energy trending in opposite directions increases the pK_a. When the free energy of the acid is lower than that of the conjugate base, the pK_a becomes greater than 7 and vice versa. The free energies of acid and conjugate base for the most part are determined by the molecular structure and thus the molecular effects on the dissociation centre exerted by its neighbourhood. An overview of the molecular effects on pK_a is given in the following. For a more detailed description see Perrin et al. [2].

1.1.2.1 Inductive Effect

This effect occurs in molecules wherever different elements with different electronegativities are σ-bonded to each other. As the more electronegative atom will draw more electronic density towards itself, it will gain a partially negative charge and induce a partial positive charge on the other atom. These partial charges can then be propagated through the carbon backbone of the molecule as partially positively or negatively charged carbon atoms will draw electronic density from their neighbouring carbon atoms, respectively donate electronic density to them. The impact on any particular atom will decrease rapidly with the distance to the origin of the inductive effect. However, this attenuation will occur less rapidly in unsaturated groups like −C=R−, −C≡R− or −C=N− [2]. Examples for elements and functional groups that draw electrons from neighboring carbon atoms (-I-Effect) are, in order of increasing effect, H $<$ $C_6H_5$$<$ −OR $<$ I $<$ Cl $<$ $CONH_2$ $<$ COOH $<$ CHO $<$ CN $<$ NH_3^+. The presence and proximity of these groups to the reaction centre lowers the pK_a as it stabilises the deprotonated conjugate base. On the other hand, examples of groups that donate electronic density are H $<$ CH_2R $<$ $CH(R)_2$ $<$ $C(R)_3$, where the higher the number of neighboring carbon rests (R), the greater the electronic density being donated. This results in a destabilisation of charged conjugate bases respectively lowering the urge for the acid to part with a proton and therefore increases to pK_a towards less acidic respectively towards basic values.

Although there is thought to be a difference between charge inducing effects through chemical bonds and effects contributed via electrostatic fields (e.g. variable proximity of groups in the cis and trans configuration, leading to different pK_a values for these isomers), they are usually summarized as *Inductive effects* as they tend to operate in the same directions and most of the time can't be separated. Generally, the characterisation of the inductive effect in terms of electron density is merely descriptive and, from the point of view of rigorous quantum chemistry, unacceptable [3].

1.1.2.2 Mesomeric Effect

This effect (±M) is attributed to delocalised pi - electrons in conjugated systems and contributes significantly to the degree with which the strength of an acid or a base is modified by remote substituents, especially in aromatic or heteroaromatic systems that contain ortho or para substituents (substituents in meta position have negligible mesomeric effects). Mesomeric effects may enhance or oppose inductive effects, as can be seen in Table 1.1 which lists a number of functional groups contributing +I, -I, +M and -M effects, respectively.

Tab. 1.1: Inductive and resonance effects of substituents

+I (acid-weakening)	$-CO_2^-$, $-O^-$, $-NH^-$, −alkyl
-I (acid-strengthening[1])	$-NH_3^+$, $-NR_3^+$, $-NO_2$, $-SO_2R$, −CN, −F, −Cl, −Br, −I, $-CF_3$, $-COOH$, $-CONH_2$, −COOR, −CHO, −COR, −OR, −SR, $-NH_2$, $-C_6H_5$
+M (acid-weakening)	−F, −CI, −Br, −I, −OH, −OR, $-NH_2$, $-NR_2$, −NHCOR, $-O^-$, $-NH^-$, −alkyl
-M (acid-strengthening)	$-NO_2$, −CN, $-CO_2H$, $-CO_2R$, $-CONH_2$, $-C_6H_5$, −COR, $-SO_2R$

[1] Approximately in decreasing order.

1.1.2.3 Steric effects

Steric interactions of the dissociation site with other parts of a molecule can lead to changes in the otherwise expected pK_a values. Examples include destabilisation of ions by hindering hydration, increased difficulty of protonating amino groups by neighbouring alkyl groups or stabilisation via internal hydrogen bonding.

1.1.2.4 Entropic effects

Entropic effects can occur when a molecule has multiple groups n, each with an equal probability to lose or gain a proton. The fact that butanoic acid (figure 1.1) has two equivalent ways of losing a proton but only one site to which the proton can be restored, is lowering the pK_a by log n (0.3 = log 2) from the expected pK_a value. On the other hand, the monoanionic form has only one ionizable proton whereas the dianion has two identical sites for protonation. Therefore, the pK_a of the second deprotonation is weakened and appears increased by log 2 compared to the otherwise expected pK_a.

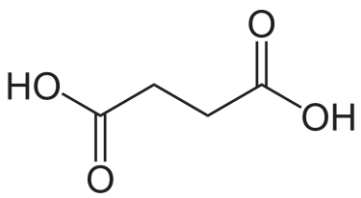


Fig. 1.1: butanedioic acid

1.1.2.5 Tautomers

Tautomers like keton-enols, amide–imidic acids and many others usually have different pK_a values depending on their tautomeric forms. When measuring the pK_a of a component, the measurement is macroscopic in nature and will be a combination of the different microscopic pK_a values of the tautomers. Only if one tautomer is predominantly present, the corresponding microscopic pK_a can be approximated to be equal to the measured macroscopic pK_a.

1.2 Importance of pK_a in drug discovery

In pharmaceutical sciences and drug discovery the impact of the pK_a, which can also be referred to as the ionization constant, will be examined by tracing its influence on the 5 areas of the absorption, distribution, metabolism, excretion, and toxicity (ADMET) scheme and its effect on the mode and strength of interactions of drugs with their targets. Additionally, we will have a look at the considerations in which the pK_a plays a role for drug formulation, the process of combining different chemical substances, including the active drug, to produce a final medicinal product. The subsequent remarks largely follow the work of Manallack et al. [4], which we refer to for a more granular description of the impact of pK_a in drug discovery.

1.2.1 Absorption

Several studies suggest that the rate of absorption is much higher for uncharged than for charged molecules and absorption rates can therefore differ widely between similar compounds when these differ in their pK_a values [5, 6, 7, 8]. Generally, the findings of Schanker et al. [5] suggest that rates of absorption of a wide variety of drugs are much faster if pK_a values are above 3 for acids and below 8 for bases. Considering the lipophilic nature of membranes these findings regarding the favoured transportation of uncharged species are to be expected. Also, the findings of Gleeson [9], suggesting that permeability of bases tend to be higher than that of zwitterions and acids (neutrals > bases > zwitterions > acids) can be explained by the anionic nature of lipid headgroups, rendering the membrane negatively charged, which favour interactions with positively charged bases, instead of acids and zwitterions containing negatively charged groups. Although, when it comes to bioavailability, Gleeson [9] observed that acids are found to be more bioavailable than bases, as the low pH in the gastrointestinal tract inclines to lead to protonation of the bases, thus reducing lipophilicity, increasing polarity and therefore limiting passive absorption. Obviously simple classification of molecules into bases and acids does not suffice for explaining qualitative or even quantitative observations of permeability and bioavailability and must be complemented by the consideration of pK_a values. Despite their findings that higher ratios of uncharged species increase the absorption remarkably, Palm et al. [6]. emphasized the significant contribution of the ionised form in situations where the unionized form makes up less than 10%.

1.2.2 Distribution

The theoretical property, volume of distribution (V_d), can be used to assess the distribution of a drug throughout the target system. Larger values of V_d indicate that the compound is widely distributed, while smaller values imply that the drug mostly persists in the systemic circulation [4]. In conjunction with information about clearance, the biological half-life of a particular drug can be calculated (half-life $= 0.693 \cdot V_d/clearance$). We can therefore see that at comparable clearance values, a higher V_d leads to increased half-life and persistence of a drug in the system. The findings of Gleeson [9] suggest that V_d tends to be the highest for basic and the lowest for acidic molecules. This can be explained as acids generally tend to be constrained to plasma due to

their strong affinity to bind to human serum albumin (HSA), the most abundant protein in human blood. Bases on the other hand do not bind to HSA as strongly and additionally, due to their affinity to negatively charged membranes and tissue, will distribute more widely into other compartments of the body. For bases and neutral compounds, an increase in the logarithmic partition coefficient (clogP) tends to increase V_d while the same effect is not observable for zwitterions and acids [9], which further underscores the high impact of plasma protein binding on the Distribution.

1.2.3 Metabolism and Excretion

Excretion may take place via hepatic, renal, and biliary processes and is regarded to be the most difficult to predict of the ADMET processes due to the dependence on structural aspects being higher than that on physicochemical properties [9]. Molecules with good solubility may have higher clearance while compounds with a positive logarithmic distribution constant ($\log D_{pH=7.4}$) will generally be reabsorbed in the kidneys and are required to be metabolised in order to become water-soluble. Drugs and other exogenous chemicals are often metabolised by cytochrome P450 enzymes, whereas the different isoforms of cytochrome P450 vary drastically in their sensibility to charge states. Furthermore, it must be considered that cytochrome P450 enzymes as well as other drug metabolising enzymes can yield biologically active metabolites, through processes of bioactivation [4]. The anionic forms of acids tend to have much lower clearance rates than basic, neural or zwitterionic species, which can be explained by their high affinity to plasma proteins and thus much lower susceptibility to clearance in the liver [9].

1.2.4 Toxicity

Although toxicity is difficult to reduce to individual molecular properties and depends on a variety of factors, trends for ionization states indicating the likeness for certain complications have been identified. For instance, basic compounds are more likely to inhibit hERG-channels and additionally with increased clogP, more likely to cause phospholipidosis. On the other hand, drugs causing mitochondrial dysfunction by acutely reducing ATP production, have been shown to be more likely acidic or lipophilic in nature. Lastly, off-target activity which can lead to unwanted and toxic effects, although again being very complex, has been found to occur more frequently with basic drugs that tend to be more promiscuous than acids, neutrals and zwitterions [4].

1.2.5 Drug–receptor interactions

The binding pose of a small molecule within its target macro-molecule may involve a variety of electrostatic interactions including hydrogen bonding, ionic bonding, dipole-dipole interactions (London dispersion), ion-dipole interactions and cation–π interactions. During lead optimization, medicinal chemists are often tempted to improve docking potency by adding lipophilicity or groups that make interactions. This approach often leads to a phenomenon called *molecular obesity* and frequently results in problems involving ADMET processes or solubility. More specific interactions like hydrogen bonding require precise placement of functional groups and are often punished with entropically disfavored desolvation processes. Several metrics like *ligand efficiency* (binding free energy per heavy atom) and similar measures are developed to counteract molecular obesity. Naturally, there is no universal rule of thumb for drug-pK_a for ligand-protein interactions, but evidently, the pK_a and the resulting ionic state of a compound in the pH environment at the target is a crucial factor for the strength of an interaction or even the possibility of interaction.

1.2.6 Formulation

When it comes to pharmaceutical formulation, apart from the presence of unionizable polar groups, the pK_a of a molecule has a tremendous effect on its solubility, as ionised species are much more soluble in water. As Drugs generally need a certain degree of lipophilicity in order to reach the site of action (beginning with absorption in the gastrointestinal tract) and for interacting with the target receptor, efforts to enhance lipophilicity may thwart the desired solubility. One way to counteract this is to modulate the degree of ionisation by the pH of the solution formulation. It has to be ensured that the modulation does not significantly impede the receptor interaction. Generally, the pH of the injectable solution should stay within pH 4-9 to avoid pain or tissue damage, although some poorly soluble drugs still require more extreme pH values. The osmotic effect, which is influenced crucially by the pK_a of a drug and the solution pH, especially when the drug concentration is high and it thus becomes the main osmotic determinant, is another issue particularly important in injectable and ophthalmic drugs. Stability may also be affected by pK_a and therefore poses another factor dictating the adequate range of solution pH.

1.3 Analytical determination of pK_a

1.3.1 Overview of analytical methods

Methods for determining the pK_a values of organic compounds include potentiometric, conductometric, voltammetric, calorimetric, spectrometric, fluorimetric and polarimetric techniques, NMR titration, liquid chromatography, capillary electrophoresis and approaches that derive pK_a from solubility experiments or kinetic measurements of reactions and combinations of some of the listed processes. The potentially most accurate and precise are conductometric methods (± 0.0001 pK_a units or better), while the most widely used methods tend to be based on pH measurements, which limits the theoretical accuracy to ± 0.02 pK_a units at best [10]. For a detailed introduction to these methods and a discussion of their advantages and limitations, the reader is referred elsewhere [11, 12, 13].

1.3.2 Data quality and precision

The International Union of Pure and Applied Chemistry (IUPAC) has sponsored compilations of pK_a values which not only assess the precision of the empirical data but also examine the reproducibility of the measurements, the purity and stability of the materials being investigated and of the solvents used, the consistency of the temperature, etc [14]. The precision of the pK_a values described in these compilations are classified as *very reliable* (pK_a error < 0.005), *reliable* (pK_a error 0.005 to 0.02), *approximate* (pK_a error 0.02 to 0.04), and *uncertain* (pK_a error > 0.04), which might seem very strict, but becomes evident, when considering that a pK_a difference of 0.04 translates to a difference in Ka values of close to 10%. Small changes like that may have a great impact on predicted ionization ratios, solubility and therefore overall physicochemical behaviour at pH values close to the alleged pK_a values. In his pK_a compilation, Prankerd [10] critically reviews the sadly very common, sloppy practices used when compiling physicochemical constants, such as pK_a into secondary literature. He lists pitfalls such as failing to include experimental conditions and errors, sometimes neglecting to mention when solvents other than pure water are used or the also very common practice of compiling data from other secondary literature, which makes it very difficult and sometimes even impossible to find the original experimental literature and frequently introduces various errors. These include the omittance of values when multiple different pK_a values were measured for the same molecule, plainly copying the wrong pK_a values from the original source, confusing pK_b for pK_a or, when

pK_a values were calculated from pK_b values (via $pK_a + pK_b = pK_w$), using the value for pK_w at 25 °C (14.008), even when the temperature for the pK_b measurements ($\neq$ 25 °C) was available. The most questionable mistakes include listing a value that is more than a hundred years old and differs by more than 1.4 pK_a units from the more recent measurement and the confusion of the ionization constant with values of the negative log of the affinity of a drug to a receptor, which in pharmaceutical literature sometimes is also called pK_a.

1.4 pK_a prediction

There are a number of different computational approaches to predict pK_a values e.g. based on Linear Free Energy Relationships (LFER), Quantitative Structure-Property Relations (QSPR) or Quantum Mechanical and Continuum Electrostatic Methods. For a broader review of these methods, we refer to the reviews by Lee et al. [15]. and Shields et al. [16]. The remarks of this section will be confined to the in-silico methods that were used in the work.

1.4.1 Extended connectivity fingerprint (ECFP)

Most machine learning models can only handle a fixed number of inputs. Molecules vary significantly in their size and structural complexity and therefore in the amount of representative information that can be used as input for models. In order to be able to predict any characteristics from the chemical structure of a compound, it has to be transformed into a standardised, constant-sized form. One of the most popular ways to achieve this are extended connectivity fingerprints (ECFPs) [17], also called Morgan fingerprints. ECFPs are circular, topological fingerprints that present molecular substructures by means of circular atom neighbourhoods. The process of generating these fingerprints is outlined in figure 1.2 and the following explanation is taken from the documentation of ChemAxon[1].

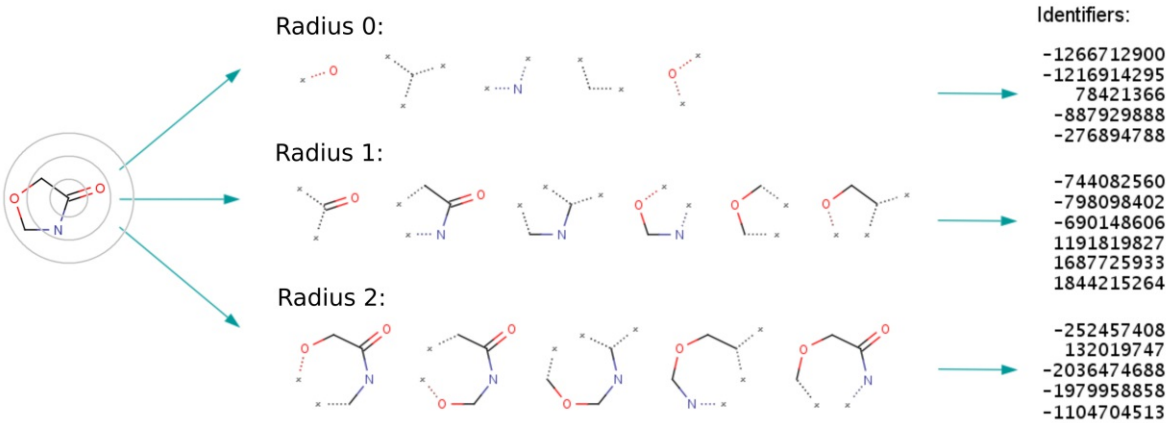


Fig. 1.2: Generation of ECFPs on the example of 4-Oxazolidinone. The illustration shows how all substructures in a specified range of radii are calculated for a given molecule and get assigned an identifier value.

It begins with the assignment of an initial integer identifier to each non-hydrogen atom of the input molecule. This identifier captures some local information about the corresponding atom

[1] https://docs.chemaxon.com/display/docs/extended-connectivity-fingerprint-ecfp.md#src-1806333-extendedconnectivityfingerprintecfp-fig-2

in such a way that various atom properties (e.g., atomic number, connection count, etc.) are packed into a single integer value using a hash function.

After that, a number of iterations are performed to combine the initial atom identifiers with identifiers of neighbouring atoms until a specified radius is reached. Each iteration captures larger and larger circular neighbourhoods around each atom, which are then encoded into single integer values using a suitable hashing method and these identifiers are collected into a list.

This iterative updating process is based on the Morgan algorithm [18], which is why ECFPs are often referred to as Morgan Fingerprints.

The final step of the generation process is the removal of multiple identifier representations of equivalent atom neighbourhoods. Two neighbourhoods are considered to be equivalent if they contain exactly the same set of bonds or their hashed integer identifiers are the same. If the identifier counts should be kept, then this step is modified to store each integer identifier as many times as the corresponding substructural feature occurs in the molecule.

This process yields a set of unique identifiers, the number of which is different for each molecule. In order to convert the data to a constant-sized input, it is compressed into a bit vector of user-defined size (e.g. 4096) by applying a mod function (e.g. mod 4096) and setting the bits to 1 at all the vector indices corresponding to the results of the mod function applied to each identifier. These bit vectors of constant size can be used as inputs for the machine-learning models.

1.4.2 Random forest

The main baseline model used in this work is random forest regression. Decision trees form the basis of this method, but they often suffer from overfitting through over-complexity and tend to be non-robust, meaning small changes in the training data can result in larger changes in the tree model. Random forests improve this by creating multiple randomized decision trees. The randomness is introduced on one hand by bootstrapping the data for every tree, meaning random sampling with replacement for a number of samples equal to the number of elements in the training data. On the other hand, when creating each tree, for every node/split, the decision variable is chosen only from a certain number of randomly selected variables, instead of all variables. When predicting, results for all trees are calculated and averaged to yield the final model result. This method has been shown to be much more robust and generalizable [19].

1.4.3 Graph neural networks

1.4.3.1 Overview neural networks

The general idea of artificial neural networks (also simply called neural networks) is to use a collection of interconnected nodes that loosely resemble our idea of how neurons in the human brain work. Each neuron has inputs and outputs and uses some kind of non-linear activation function, to determine how to convert the first into the latter. The most popular activation function used is rectified linear unit ($f(x) = max(0, x)$), which outputs 0 for every input x <= 0 and x for every input x > 0.

Figure 1.3 [2] shows the structure of one of the most basic neural networks, called Multilayer perceptron. It is organised in layers of neurons where each neuron is connected with all neurons of the layer before and after and each connection is assigned a weight and optionally a bias which are the parameters that will be optimized during the training of the model. These kinds of layers are called *fully connected feed-forward layers* or *linear layers*. There at least two layers, an

[2] https://www.kdnuggets.com/2019/07/convolutional-neural-networks-python-tutorial-tensorflow-keras.html

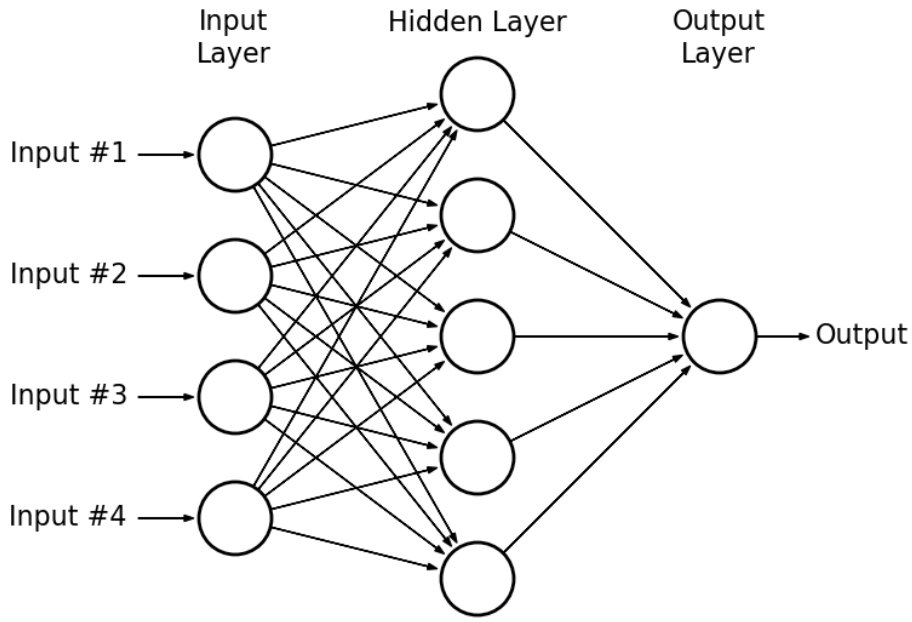


Fig. 1.3: Scheme of a fully connected multilayer perceptron. All nodes of every layer are connected to all nodes of the layer before and after. There is one input and one output layer with the number of input and output nodes corresponding to the number of input and output variables, while the number of hidden layers and their number of nodes can be chosen arbitrarily.

input and an output layer, with an arbitrary number of so-called hidden layers in between. The number of neurons in the input layer corresponds to the number of input variables used and the number of neurons in the output layer reflects the number of target variables, while the number of neurons of the hidden layers can be chosen freely and may be subject to hyperparameter optimisation. For every prediction, the model takes the values of all the input variables and feeds them to the neurons in the next layer modified by the respective weights and biases of each connection. Every neuron in the next layer takes the sum of all the outputs from the neurons from the previous layer, connected to it as inputs to its activation function and passes the output along with the weighed connections to the neurons in the next layer. This process is done for all neurons in all hidden layers and ends with all outputs of the last hidden layer (or input layer, if there are no hidden layers) feeding into the neurons of the output layer. For classification tasks, the output neurons represent the predicted likelihood for each class assignment, with the condition of all output neurons' values always adding up to one. For regression tasks, like pK_a prediction, a single neuron that represents the prediction value, is used.

When initializing a neural network, all weights and biases are assigned randomly and therefore also the predictions of the model are random. Fortunately, when comparing the output of the network with the true values, a gradient that indicates how to adjust all weights and biases in order to match the output with the true values, can be calculated. For a more detailed explanation of how this gradient is calculated we refer elsewhere [20]. As neural networks often have a vast number of adjustable parameters, ranging from tens of thousands to many millions, adjusting the parameters to perfectly fit the outputs to the true values of a dataset leads to massive overfitting.

A first step to avoid overfitting is therefore to only adjust the parameters by a fractional amount (e.g. 0.001) called the learning rate. Training the model can then be done by either predicting one sample at the time, calculating the gradient and adjusting the parameters or by predicting multiple samples, calculating theirs gradients and then adjusting the parameters by the sum or mean of the gradients, also called batching. Batching further helps the model to mainly learn patterns that correspond to multiple samples instead of overfitting sample-specific characteristics. As the parameter adjustment after every batch of samples depends on the learning rate and tends to be quite small, the training of a neural network usually involves repeating the steps listed above for every batch in a training set and further repeating the training on every batch for a certain number of times (epochs). Each epoch represents a circle of repetition in which the models sees all training data once. Model performance and training progress are evaluated by a loss function, which quantifies the difference between predicted and true values. Furthermore, to evaluate the generalizability of the model, its performance is tested on a separate validation set, on which it is not trained. Usually, it can be observed, that after a certain number of epochs the performance of training and validation loss discontinue to further improve, which marks the end of the training.

Using ECFP bit-vectors a multilayer perceptron can be utilised to predict pK_a values. Another way to use data in machine learning models is to represent molecules as graphs with the nodes and their connecting edges representing atoms and bonds, respectively. Each node and edge can be assigned features (like atomic number, charge, hybridisation, total number of attached hydrogen, bond type, etc.) that are considered relevant for the prediction of the desired property. These graph representations can then be used in special forms of artificial neural networks, called graph convolutional networks.

One special property of such graph convolutional networks is that they can take graphs of any size as their inputs. Using all nodes and edges of a graph as inputs is not possible as the architecture of the model needs to be constant and cannot change the number of input neurons for the different graphs in the dataset. Instead graph convolutional networks (GCNs) simple use one node and its corresponding features at a time. This leads to a constant number of inputs for all nodes in all graphs corresponding to the number of node features used in the data.

The principle graph convolutional layer used in this work is the graph convolutional operator, described by Kipf and Welling [21] and is shown in Equation 1.7.[3]

$$\mathbf{X}' = \sigma(\hat{\mathbf{D}}^{-1/2}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-1/2}\mathbf{X}\mathbf{\Theta}) \tag{1.7}$$

σ represents an activation function, $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ denotes the adjacency matrix with inserted self-loops and $\hat{D}_{ii} = \sum_{j=0} \hat{A}_{ij}$ its diagonal degree matrix. The dimensions of both of these matrices are [number of nodes, number of nodes]. X is the node matrix with the dimension [number of nodes, number of features] and Θ a linear layer that maps the features of every node to set of new features, represented by a matrix of shape [number of input features, number of output features]. The equation describes a process wherein a first step, all features of each node are put through a linear layer and mapped onto a new set of features and then, message passing is conducted, summing the normalised feature vectors of each node and its neighbouring node to constitute the new node vector. This process works, regardless of the number of nodes in a graph as the only trainable parameters are contained in the linear layer matrix Θ which is only determined by the number of input and output (node-) features but independent of the number of nodes. Multiple GCN layers can be used consecutively, representing multiple steps of message passing, thereby informing each node about an ever-larger radius of neighbouring nodes. In order

[3] (`https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#torch_geometric.nn.conv.GCNConv`)

to be able to use the graphs resulting from the message passing for classification or regression tasks, they must be transformed into feature vectors whose dimensions are independent of the number of nodes of the graphs. This can be done via a pooling layer that takes the node matrix [nodes, features] and aggregates the features of for example by summing, averaging or selecting the maximal value of all nodes for every feature. The resulting feature vector can then be passed through a multilayer perceptron to yield the final prediction variable. Figure 1.4 [4] shows an overview of such a graph neural network.

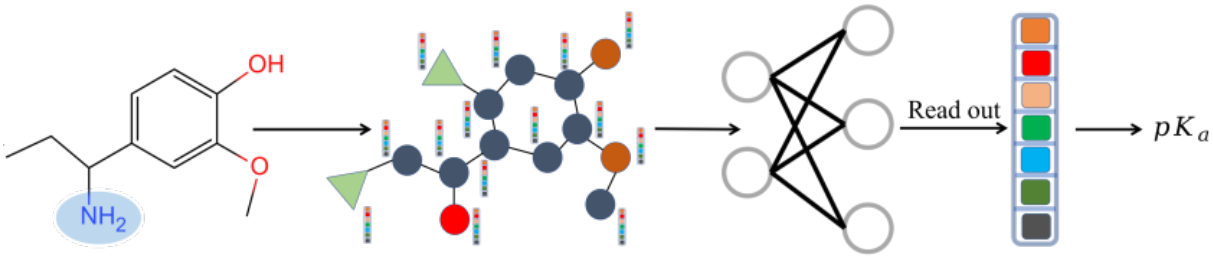


Fig. 1.4: Scheme of the steps involved in a graph convolutional network. The illustration indicates the transformation of the input molecule into a graph representation with node and, optionally, edge features which then enters the message passing phase (node convolution), whereby information about each atom and its neighbours is encoded into a new representation of the molecule. The last step (readout) uses the output of the convolutional layers to predict pK_a with a multilayer perceptron.

[4] http://xundrug.cn/molgpka/about

Chapter 2

Methods

2.1 Description of the *pkasolver* package

The entirety of the workflow presented in this chapter is conducted via the python [22] programming language as it provides adequate tools for handling and manipulating data tables with pandas [23], numerical data using NumPy [24] and chemical data utilising RDKit [25], as well as packages for using machine learning models - scikit-learn [26], neural networks - PyTorch [27] and graph neural networks - PyTorch-geometric [28]). Furthermore, the packages matplotlib [29] and seaborn [30] are used for plotting.

The code of this work is public and open-source (MIT License) and can be accessed via GitHub[1]. It consists of a juypter notebook *thesis_pipeline.ipynb* that contains the entire workflow except for the cross-validation training and a script *train_script.py* that enables parallel computation of all train-test-split and cross-validation models on multiple machines. The global variables are stored and called from the file *config.py* and the code for the neural network architectures as well as the training functions from the file *architecture.py*. All of these files, as well as the directories storing saved data and models, are found within the *examples* folder of the repository where each run (multiple runs with different configurations may be conducted) is stored in separate subfolders. Most of the functions used in the code are stored and called from a custom made python package called *pkasolver*. The notebook is intended to enable going through data preparation processes, model training and evaluation, observing the individual steps, their inputs and outputs, and experimenting with new ways to use the package and the pipeline. The script on the other hand can be used for preprocessing data and training models of the cross-validation in a more automated fashion.

2.2 Data

2.2.1 Datasets

The datasets used in this work were curated by Baltruschat et al. [1] and acquired from their GitHub repository[2]. The training set consists of a file from DataWorrior [31] called *pKaInWater.dwar* merged with the result of an extensive query of all assays in the ChEMBL [32] database. The criteria for this query were:

- type: physicochemical assay,
- source: scientific literature
- organism taxonomy: “N/A”
- format: “small-molecule physicochemical format”

[1] `https://github.com/MayrF/pkasolver`
[2] `https://github.com/czodrowskilab/Machine-learning-meets-pKa`

Furthermore, the results from the query were filtered for measurements specified as exact (*standard_relation* equals "=") and for which the *standard_type* was specified to "pka", "pka value", "pka1", "pka2", "pka3" or "pka4" (case-insensitive), resulting in a dataset with 8111 pK_a values from the ChEMBL query and 7911 entries with valid molecular structures from the DataWorrier Data. This concatenated dataset was processed further via the following steps:

- Removal of all salts from molecules
- Removal of molecules containing nitro groups, Boron, Selenium or Silicium
- Filtering by Lipinski's rule of five (one violation allowed)
- Keeping only pK_a data points between 2 and 12
- Tautomer standardization of all molecules
- protonation of all molecules at pH 7.4
- Keeping only monoprotic molecules regarding the specified pK_a range
- Combination of data points from duplicated structures while removing outliers

The completely preprocessed training dataset consists of 5994 unique molecules. Further details regarding the data preparations are found in the publication of Baltruschat et al. [1] Two more datasets named *Literature* and *Novartis* were provided by Baltruschat et al. They were processed via the same steps described above for the training set and used for testing purposes only. *Literature* is a set of 123 compounds collected by manual curation of literature [33, 34, 35, 36, 13] and *Novartis* is a dataset of 280 molecules provided by Novartis [37]. All three datasets were provided preprocessed and stored in sdf files by Baltruschat et al. [1].

2.2.2 Conjugate creation algorithm

One goal of this work was to assess whether the performance of predictive models for pK_a is enhanced when using both the protonated and the deprotonated species involved in an acid-base reaction. Therefore, an algorithm was written, to create both of these species by calculating the respective conjugate from a single molecule of each data point provided in the datasets. During the process of developing the algorithm, it was observed that a significant portion of molecules in the provided datasets are actually not representing the protonation state at pH 7.4, as they were supposed to, according to the preprocessing steps described by Baltruschat et al. [1]. The way this was detected was on one hand the observation, that some basic molecules with a $pK_a > 7.4$ that are supposed to be in their protonated form have no spare hydrogen molecules attached to their reaction centre (usually a nitrogen atom) and can therefore not be deprotonated to obtain its conjugate. On the other hand, some acidic molecules with $pK_a < 7.4$ were found to have a reaction centre with a positive charge, the protonation of which would lead to obtaining a conjugate containing a reaction centre with a charge of +2. In a pK_a range of 2–12, this is highly unlikely to represent reality.

The algorithm developed to correct for these errors, calculate the conjugate molecules and sort the resulting and existing molecules into protonated and deprotonated species does so by checking which of the three conditions are met and acting accordingly:

1. The experimental pK_a is higher than the supposed protonation state pH (pH = 7.4 in our case) and the number of Hydrogens at the reaction centre is higher than 0 **or** the reaction centre charge is higher than 0: the molecule is considered protonated and therefore, the

deprotonated conjugate is created by decreasing the formal charge and the number of hydrogens at the reaction site by 1.

2. The experimental pK_a is lower than the supposed protonation state pH (pH = 7.4) and the reaction centre charge is lower than 1: the molecule is considered deprotonated and therefore the protonated conjugate is created by increasing the formal charge and the number of hydrogens at the reaction site by 1.

3. The experimental pK_a is higher than the supposed protonation state pH (pH = 7.4) but the number of hydrogens at the reaction site is 0: the molecule is supposed to be in its protonated form, but as it is not possible to deprotonate an atom with no protons, the given molecule is assumed to be in its deprotonated form and the protonated conjugate is created by increasing the formal charge and the number of hydrogens at the reaction site by 1.

A total of 581 errors in the training set (581/5994), 23 errors in the *Novartis* set (23/280) and 6 errors in the *Literature* set (6/123) can be found and correct using the algorithm described in this section.

2.2.3 Featurization

For the baseline machine learning models that require constant sized inputs, ECFPs are created via the function `GetMorganFingerprintAsBitVect`, provided by the RDKit library [25]. The same settings as Baltruschat et al. [1] are used, applying a bit length of 4096 and a radius of 3.

The atom and edge features of the input graph data for the GCNs are obtained via the corresponding RDKit functions. The features and their respective range of unique values across all datasets are shown in table 2.1.

Tab. 2.1: List of atom and edge features used for the graph representations of molecules. It also includes all respective unique values yielded for all molecules of the datasets used in this work

Atom features	
atomic number:	1, 6, 7, 8, 9, 15, 16, 17, 33, 35, 53
formal charge	-1, 0, 1
chiral tag:	0, 1, 2 (None, R, S)
hybridization:	1, 2, 3, 4
total number of Hs:	0, 1, 2, 3
explicit number of Hs:	0, 1, 2, 3
aromatic tag:	True, False
total valence:	1, 2, 3, 4, 5, 6
total degree:	1, 2, 3, 4
ring:	True, False
amide center atom:	True, False

Bond features	
bond type:	1.0, 1.5, 2.0, 3.0
conjugated:	True, False
rotatable:	True, False

2.2.4 Data preprocessing

The dataset sdf files are loaded into a Pandas [23] Dataframe object via the PandasTools subpackage of RDKit. Then the conjugates of all molecules are calculated via the aforementioned conjugate creation algorithm (subsection: 2.2.2) and added to the DataFrame objects in their respective columns named *protonated* and *deprotonated*. The *Training* dataset is split randomly into a *train* and a *validation* set consisting of respectively 80 % and 20% of the *Training* samples. All the DataFrames are combined into a dictionary and saved via python's pickle module for later use.

For the fixed-sized inputs, the Morgan-fingerprint bit-vectors are calculated for both the protonated and deprotonated molecule and stored in a Numpy array object with a shape [number of molecules, 2 × number of vector bits] for each dataset. The resulting array, together with the target vector (shape: [number of molecules]) containing the experimental pK_a values, are ready to be used to train and test the baseline models.

For the creation of the molecular graphs, all-atom features, bonds and bond features are calculated for protonated and deprotonated versions of the molecules and stored in objects of a customised version of the PyTorch Geometric `Data` class called `PairData`. These objects are then combined to a list containing all data of a dataset, where they are then ready to be used for the batching, training and testing steps of GCN modelling.

All the fingerprint and graph data are combined into a dictionary and saved via the pickle module for later use. All above steps are also done to prepare the datasets for the 5-fold-cross-validation experiments, where the *Training* dataset is split into 5 random parts of equal size, of which, for each cross-validation (CV) one part is chosen to be the validation set and 4 parts to be combined to the train set. All the preprocessing steps listed above were also conducted for the cv datasets.

2.3 Models

2.3.1 Baseline models

When developing a new predictive model, it is important to have reference models that can serve as a baseline and indicate to what extent the new, usually more complex model outperforms other simpler or already established types of models. From several models tried in various configurations by Baltruschat et al. [1] (Random forest, support vector regression, multilayer perceptron and XGradientBoost) the random Forest regressor (RFR) models turned out to be the best performers and are therefore chosen to be used as the baseline models for this work. Also, partial least squares (PLS) models are used additionally, to see how linear models compare to the non-linear random forest and GCN models.

All the baseline models are implemented in the form of their respective Regressor objects, provided by the scikit-learn library [26]. For PLS, the default parameters of Scikit-Learn (version 0.24.2) are used. For the RF-Models the number of estimators (= number of trees in the forest) is set to 100. This parameter was set to 1000 by Baltruschat et al. [1] but preliminary testing revealed that the difference between the performance of 1000 and 100 estimators is marginal for our datasets. For all model types, three kinds of molecular data, using either the fingerprint vectors of only protonated (1x4096 bits) - *prot*, only deprotonated (1x4096 bits) - *deprot* or combined, referred to as *pair* (2x4096), are used for training and testing.

2.3.2 GCNs

For the implementation of the graph-based neural network models, the python libraries PyTorch [27] and PyTorch Geometric [28] are used.

2.3.2.1 Architecture

Four different types of architectures are used, two of them taking only graphs of either the protonated - *prot* or the deprotonated - *deprot* molecules as input and two that use the graphs of both the protonated and deprotonated - *pair* species as input data. For each data configuration there is one model useing only node features and one model using node and bond features (marked with the *edge* - keyword) of the input graphs for training and testing.

The architectures used in this work have been determined by confined empirical testing and were found to be suitable for the task at hand, but have not been subject to thorough, systematic hyperparameter testing.

2.3.2.2 Single input

- Convolutional Layers:
  1. `GCNConv / NNConv`(11, 96)
  2. `GCNConv / NNConv`(96, 96)
  3. `GCNConv / NNConv`(96, 96)
  4. `GCNConv / NNConv`(96, 96)
- `global_max_pool` function (number of atoms $\times$ 96, 96)
- Linear layers:
  1. `Linear`(96, 96)
  2. `Linear`(96, 1)

Fig. 2.1: GCN architecture for single-molecule inputs (*prot*, *deprot*). The `GCNConv` layers are used for atom features only, while for models utilizing edge features additionally, the `NNConv` layers are used instead.

The basic input structure is depicted in figure 2.1. First, we have 4 layers of the type `GCNConv`, with the first layer having an input size according to the number of node features (11) and 96 outputs. The subsequent layers all have 96 in- and output features. All the atoms of a molecule get passed through the convolutional layers separately and then get combined by the `global_max_pool` function, which takes the maximum value for every feature of all of the graphs atoms. This results in a vector of length 96 which is then fed through two fully connected `Linear` layers that result in a single output that can then be trained to estimate the pK_a values.

The structure for the model incorporating the edge features differs from this basic model by using the `NNConv` layer for all convolutional layers, which factors in the edge features in form of a sequence of two `Linear` layers.

- graph layers (protonated)
  - Convolutional Layers:
    1. `GCNConv / NNConv`(11, 96)
    2. `GCNConv / NNConv`(96, 96)
    3. `GCNConv / NNConv`(96, 96)
    4. `GCNConv / NNConv`(96, 96)
  - `global_max_pool` function (number of atoms × 96, 96)
- graph layers (deprotonated)
  - Convolutional Layers:
    1. `GCNConv / NNConv`(11, 96)
    2. `GCNConv / NNConv`(96, 96)
    3. `GCNConv / NNConv`(96, 96)
    4. `GCNConv / NNConv`(96, 96)
  - `global_max_pool` function (number of atoms × 96, 96)
- concatenate pooled feature vectors (2 × 96, 192)
- Linear layers:
  1. `Linear`(192, 192)
  2. `Linear`(192, 1)

Fig. 2.2: GCN architecture for *pair* molecule inputs. The `GCNConv` layers are used for atom features only, while for the model utilizing edge features, additionally, the `NNConv` layers are used instead. The graphs of the protonated and deprotonated representations of any molecule are passed through their respective convolutional layers, separately, and are then combined for the interpretation in the readout layer.

2.3.2.3 Paired input

The GCN models using paired graphs with and without edge features as input consist of two separate blocks of the same convolutional layers and pooling functions as the single graph models described above. The resulting vectors of length 96 from each block get concatenated and fed through two linear layers with the first having 192 in- and outputs and the second having 192 inputs and one final output.

2.3.2.4 Training

6 different models with varying permutations of graph data and edge feature inclusions (protonated, deprotonated, paired each with and without edge features) using the corresponding architectures, are trained. Training for each of these 6 models is done for 2000 Epochs and a batch size of 64. The learning rate is set to 0.001 and adjusted by a scheduler, called `ReduceLROnPlateau`, with patience=5 and verbose=True. The loss criterion is chosen to be mean squared error (MSE) and parameter adjustment is done using the `Adam` optimizer. For the training, a dropout layer with a probability of 0.5 is used. A dropout layer introduces an

additional source of randomness into the training process as for every value in every molecule feature vector passing through it during training it has a chance (50% in our case) to set it to 0. This prevents the model from memorizing the batches and is a further measure to prevent overfitting.

The same training process was done for the 5-fold cross-validations of all 6 different models.

2.3.2.5 Interpretation

The individual impacts of the node and edge parameters on the predictions of the models is estimated using the `IntegratedGradients` class of the PyTorch Captum library [38]. The `IntegratedGradients` object can take a trained molecule and a sample input and is then able to compute the impact of every input feature on the prediction of the sample. This was done for each of the 6 types of models on a random selection of 100 molecules from the training set and plotted using the seaborn `boxplot` function for qualitative interpretation.

Chapter 3

Results and Discussion

3.1 Baseline models

The results for the 5-fold cross-validation, conducted for the two baseline models, the random forest regressor and partial least squares regressor are shown in table 3.1.

Tab. 3.1: Cross validation training results for baseline models

	R^2 (mean ± std)	$RMSE$ (mean ± std)	MAE (mean ± std)
RFR_prot	0.791 ± 0.007	1.110 ± 0.025	0.732 ± 0.014
RFR_deprot	0.753 ± 0.015	1.209 ± 0.048	0.799 ± 0.023
RFR_pair	0.798 ± 0.009	1.093 ± 0.036	0.727 ± 0.022
PLS_prot	0.633 ± 0.012	1.473 ± 0.016	1.132 ± 0.016
PLS_deprot	0.595 ± 0.010	1.546 ± 0.042	1.189 ± 0.038
PLS_pair	0.645 ± 0.007	1.448 ± 0.025	1.110 ± 0.029

The RFR models are quite clearly outperforming the PLS models by a difference in RMSE of about 0.34 while the differences within each model type between the 3 different data types were less notable. Although the PLS models clearly performed worse, than the RFR models, it must be noted, that despite the assumed non-linearity of the problem of pK_a prediction, the results of the linear PLS models are surprisingly good. The respective model types using *pair* input data are both performing best in terms of the mean values of the metrics, but considering standard deviation, they cannot be regarded as significantly superior to the models of the same type that use protonated input data. The models using *deprot* input data are under-performing significantly compared to the models using the other two data types. This suggests that there is, at least in our dataset, a significant difference between the information relevant to pK_a contained in the protonated and the deprotonated molecular fingerprint data.

Table 3.2 shows the performance of the baseline models of the two test sets. The best performing baseline model on the *Novartis* test set is the RFR_pair model and on the *Literature* set the RFR_prot model. Just as it was the case for the training results, all Random Forest models performed better than the PLS models. Also, the models using *deprot* input data underperformed again, compared to those using *prot* or *pair* input data, expect for the PLS_deprot model on the *Literature* data set, which actually performed better than the other PLS models.

The best model of Baltruschat et al. [1], a Random Forest Regressor with a number of estimeters=1000 and ECFPs with radius=3 and 4096 bits calculated from molecules, allegedly in a protonation state at pH=7.4, has been outperformed on the Novartis test set by both the RFR_prot and RFR_pair models with a difference of about 0.06 in RMSE but was still significantly better than any of the random forest models of this work when applied to the *Literature* test set. As the model of Baltruschat et al. [1] used the same fingerprint and model configurations (apart from our model using 100 instead of 1000 estimators) as our random

Tab. 3.2: Test results for Baseline models

	Novartis			Literature		
	R^2	RMSE	MAE	R^2	RMSE	MAE
RFR_prot	0.587	1.479	1.169	**0.844**	**0.932**	**0.612**
RFR_deprot	0.423	1.749	1.379	0.756	1.166	0.767
RFR_pair	**0.604**	**1.45**	**1.139**	0.833	0.965	0.657
PLS_prot	0.423	1.749	1.379	0.66	1.376	1.049
PLS_deprot	0.361	1.842	1.473	0.727	1.232	0.935
PLS_pair	0.458	1.696	1.341	0.711	1.269	0.941
Baltr.: RFR /FCFP6 (4096 bits)	0.569	1.513	1.147	**0.889**	**0.785**	**0.532**
ChemAxon Marvin (V20.1.0)	**0.744**	**1.166**	**0.856**	0.866	0.865	0.566

forest models, the enhanced performance of our models can be assumed to originate from the protonation state errors, we found and fixed via the conjugate creation algorithm described in subsection 2.2.2. The worse performance of our models on the Literature dataset might be attributable to the 10 times greater number of estimators used by Baltruschat et al. [1]. Baltruschat et al. also predicted the test datasets with the commercial tool `Marvin`[1] by the company ChemAxon. The results that show the `Marvin` model outperforming all of our baseline models both on the *Novartis* and the *Literature* test set can be seen in the last row of table 3.2. Table 3.3 shows the cross-validation results of the baseline models on the test sets.

Tab. 3.3: Results of the cross-validation of the baseline models on the test sets

	Novartis			Literature		
	R^2	RMSE	MAE	R^2	RMSE	MAE
RFR_prot	0.625 ± 0.032	1.409 ± 0.060	1.118 ± 0.044	0.839 ± 0.009	0.946 ± 0.026	0.615 ± 0.014
RFR_deprot	0.425 ± 0.014	1.746 ± 0.021	1.365 ± 0.017	0.761 ± 0.005	1.154 ± 0.011	0.761 ± 0.017
RFR_pair	0.617 ± 0.020	1.425 ± 0.038	1.120 ± 0.029	0.832 ± 0.012	0.968 ± 0.035	0.638 ± 0.020
PLS_prot	0.422 ± 0.005	1.751 ± 0.008	1.384 ± 0.016	0.662 ± 0.019	1.372 ± 0.038	1.044 ± 0.021
PLS_deprot	0.358 ± 0.006	1.846 ± 0.009	1.479 ± 0.012	0.727 ± 0.018	1.233 ± 0.041	0.947 ± 0.026
PLS_pair	0.458 ± 0.005	1.696 ± 0.007	1.351 ± 0.015	0.710 ± 0.015	1.269 ± 0.032	0.958 ± 0.021

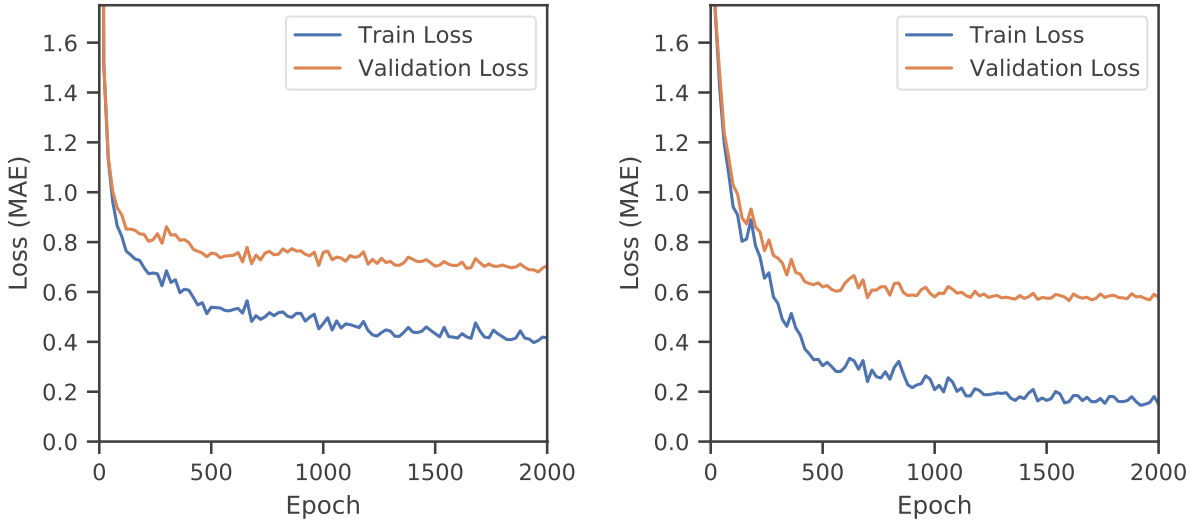
3.2 GCN Models

3.2.1 Training

All the GCN models were trained for 2000 epochs. Figures 3.1a and 3.1b show the loss on the train and the validation set plotted against the number of training epochs for the models GCN_prot_no-edge and GCN_paired_edge, respectively. For both models, the losses drop rapidly during the first 500 epochs. Afterwards, the train losses continue decreasing slightly, ending up at about 0.4 and 0.15, respectively, while the validation losses plateau after about 1000 epochs at just under 0.80 and around 0.60. One might be tempted to assume that continuing training when the training loss is going down, whilst the validation loss only barely decreases, would lead to overfitting the model and thus diminished generalizability. But as long as the validation test loss is the only measure, by which the performance of the model on data that it

[1] `https://chemaxon.com/products/marvin`

is not trained on can be assessed, it must be assumed that as long as the validation loss keeps decreasing, no matter how negligibly, subsequent training further improves the general prediction ability of the model.



(a) Training progression of GCN_prot_no-edge **(b)** Training progression of GCN_pair_edge

Fig. 3.1: Training progression exemplified by GCN_prot_no-edge and GCN_paired_edge.

For calculating the final prediction results of the GCN models, we would typically want to use the best models, according to their performance on the validation set. These models are usually obtained at epochs close to the end of the training (1500-2000). For reasons, which we will go into in the next section, we chose to only use the best models from the first 500 training epochs.

The results for the 5-fold cross-validation, conducted for the 6 different GCN models are shown in table 3.4. In terms of the mean values, we can see, that using edge features generally improves training performance and that the models using *pair* input data are slightly in the lead against models using the same edge feature configuration, but other data inputs.

Tab. 3.4: Cross validation training results for GCN models

	R^2 (mean ± std)	$RMSE$ (mean ± std)	MAE (mean ± std)
GCN_prot_no-edge	0.804 ± 0.013	1.077 ± 0.023	0.740 ± 0.020
GCN_prot_edge	0.837 ± 0.007	0.982 ± 0.036	0.663 ± 0.024
GCN_deprot_no-edge	0.796 ± 0.004	1.097 ± 0.012	0.770 ± 0.010
GCN_deprot_edge	0.835 ± 0.008	0.989 ± 0.036	0.683 ± 0.016
GCN_pair_no-edge	0.824 ± 0.011	1.021 ± 0.043	0.699 ± 0.024
GCN_pair_edge	0.850 ± 0.008	0.942 ± 0.025	0.638 ± 0.007

3.2.2 Test results

As we saw in figure 3.1, the performance of the models on the validation set does not significantly increase after training for more than 500 epochs. Looking at the prediction results on the test sets in tables 3.5 and 3.6, we actually best models chosen only from the models trained for a maximum of 500 epochs, perform close to as good or sometimes even better than best models chosen from

Tab. 3.5: Results of the cross-validation of the GCN models on the test sets (best of 2000ep)

	Novartis			Literature		
	R^2	RMSE	MAE	R^2	RMSE	MAE
GCN_prot_no-edge	0.668 ± 0.028	1.326 ± 0.055	0.971 ± 0.050	0.883 ± 0.023	0.803 ± 0.079	0.572 ± 0.058
GCN_prot_edge	0.685 ± 0.015	1.292 ± 0.032	0.981 ± 0.028	0.876 ± 0.016	0.828 ± 0.054	0.589 ± 0.034
GCN_deprot_no-edge	0.610 ± 0.029	1.437 ± 0.053	1.123 ± 0.036	0.869 ± 0.030	0.850 ± 0.094	0.611 ± 0.057
GCN_deprot_edge	0.697 ± 0.017	1.268 ± 0.035	0.984 ± 0.014	0.896 ± 0.010	0.758 ± 0.036	0.550 ± 0.043
GCN_pair_no-edge	0.702 ± 0.017	1.257 ± 0.037	0.978 ± 0.033	0.903 ± 0.012	0.734 ± 0.046	0.536 ± 0.028
GCN_pair_edge	0.728 ± 0.023	1.199 ± 0.050	0.922 ± 0.032	0.888 ± 0.010	0.788 ± 0.035	0.553 ± 0.028

Tab. 3.6: Results of the cross-validation of the GCN models on the test sets (best of 500ep)

	Novartis			Literature		
	R^2	RMSE	MAE	R^2	RMSE	MAE
GCN_prot_no-edge	0.674 ± 0.026	1.314 ± 0.052	0.997 ± 0.049	0.893 ± 0.017	0.770 ± 0.063	0.557 ± 0.051
GCN_prot_edge	0.709 ± 0.036	1.241 ± 0.077	0.947 ± 0.064	0.888 ± 0.013	0.789 ± 0.045	0.577 ± 0.033
GCN_deprot_no-edge	0.602 ± 0.048	1.451 ± 0.087	1.119 ± 0.065	0.901 ± 0.014	0.740 ± 0.052	0.560 ± 0.051
GCN_deprot_edge	0.685 ± 0.023	1.291 ± 0.048	0.985 ± 0.033	0.894 ± 0.014	0.768 ± 0.049	0.570 ± 0.035
GCN_pair_no-edge	0.700 ± 0.019	1.261 ± 0.040	0.986 ± 0.033	0.890 ± 0.012	0.781 ± 0.045	0.578 ± 0.046
GCN_pair_edge	0.729 ± 0.021	1.198 ± 0.047	0.937 ± 0.039	0.890 ± 0.017	0.780 ± 0.061	0.578 ± 0.048

all 2000 epochs. We can thus see that the model training can be considered finished after 500 epochs and that further training does not generally improve the models. We therefore decided to only use the best models from the first 500 training epochs for the subsequent performance analysis.

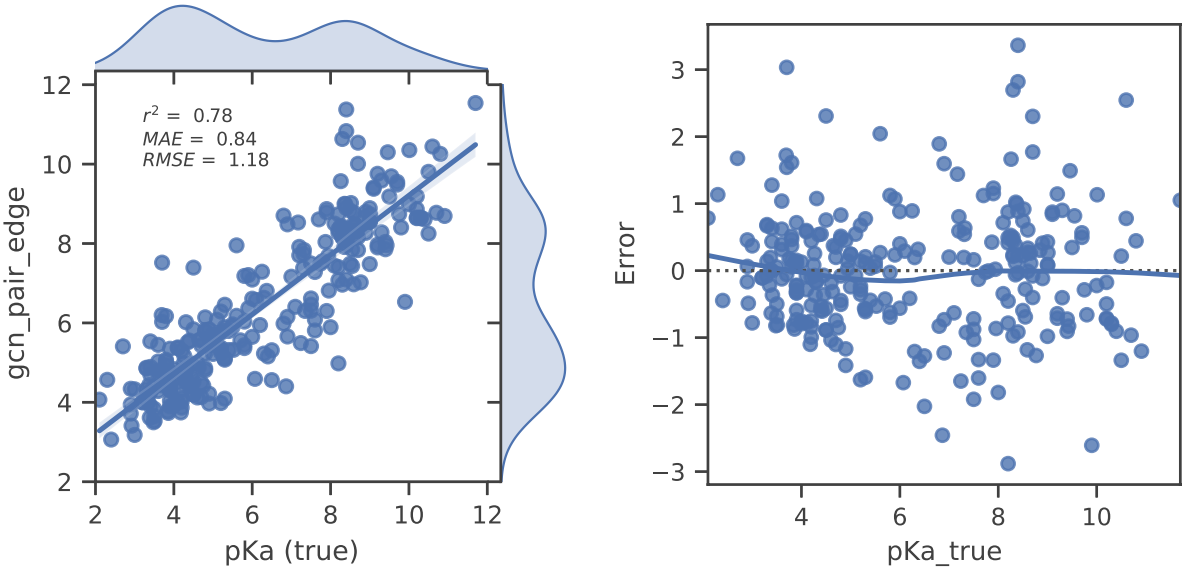
In table 3.7 the performances of the GCN models on the two test sets *Novartis* and *Literature* are shown. The table also includes the results of the best baseline models and the best models of Baltruschat et al. [1] for further comparisons. It should be noted that some of the results obtained from the models trained on the train-test split data are outside the confidence interval that resulted from the cross validation on the test data.

Tab. 3.7: Test results for GCN models

	Novartis			Literature		
	R^2	RMSE	MAE	R^2	RMSE	MAE
GCN_prot_no-edge	0.705	1.25	0.975	0.882	0.811	0.614
GCN_prot_edge	0.688	1.287	0.971	0.862	0.877	0.667
GCN_deprot_no-edge	0.61	1.439	1.102	**0.929**	**0.628**	**0.496**
GCN_deprot_edge	0.7	1.261	0.985	0.903	0.734	0.558
GCN_pair_no-edge	0.644	1.374	1.045	0.899	0.75	0.571
GCN_pair_edge	**0.751**	**1.15**	**0.873**	0.895	0.766	0.57
RFR_prot	0.616	1.428	1.126	**0.853**	**0.903**	**0.589**
RFR_pair	**0.625**	**1.411**	**1.100**	0.837	0.952	0.632
Baltr.: RFR /FCFP6 (4096 bits)	0.569	1.513	1.147	**0.889**	**0.785**	**0.532**
ChemAxon Marvin (V20.1.0)	**0.744**	**1.166**	**0.856**	0.866	0.865	0.566

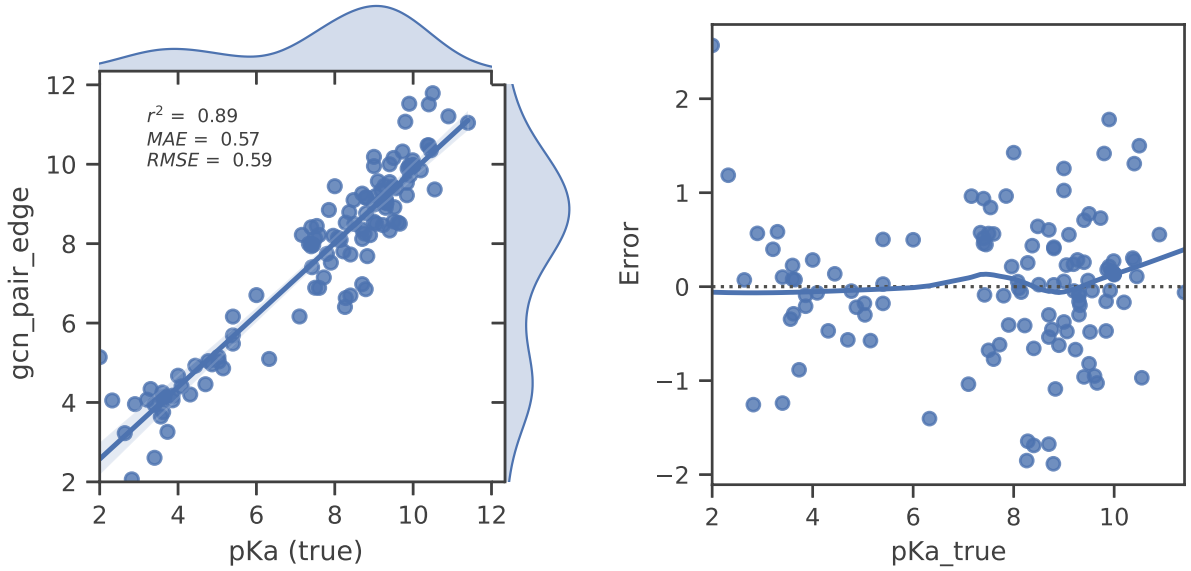
We can see that the GCN model using edge feature and pair data is the best performing model on the *Novartis* test set, outperforming the random forest pair model by a difference in RMSE of 0.26 and even slightly improving on the results of the ChemAxon's `Marvin` model with an RSME lower by 0.01 units. The same model also beats all baseline and reference models on the

Literature set, but is itself outperformed by the models, GCN_pair_no-edge, GCN_deprot_edge and GCN_deprot_no-edge, with the latter being the best performing model on this dataset. Regression and Residual plots of the results of the GCN_pair_edge model on the test sets can be seen in figure 3.2a and 3.2b for the *Novartis* test set and in figure 3.3a and 3.3b for the *Literature* test set.



(a) Regression plot **(b)** Residual plot

Fig. 3.2: Regression and Residual plots of the predictions of the GCN_pair_edge model on the *Novartis* testset. The prediction errors are distributed quite uniformly along the pK_a spectrum, with a minor overestimation at lower pK_a values.

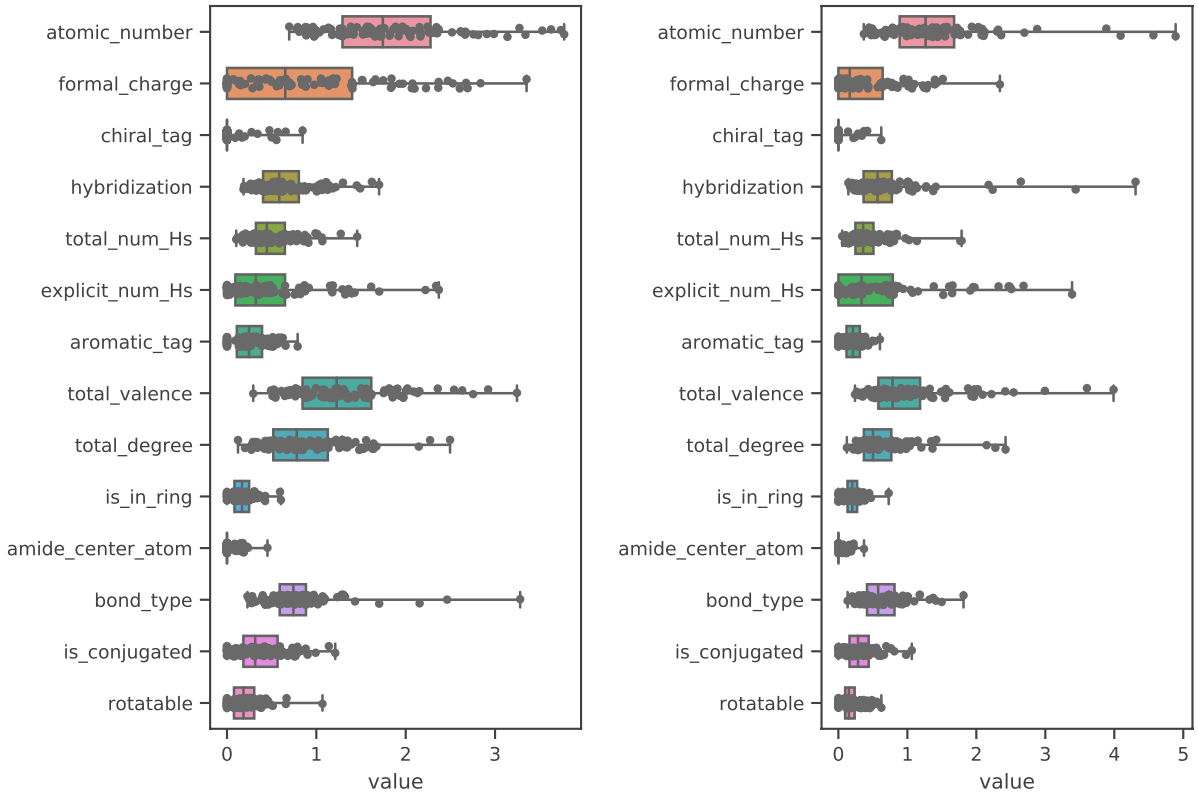


(a) Regression plot **(b)** Residual plot

Fig. 3.3: Regression and Residual plots of GCN_prot_edge on the Literature testset. The residual plot show significant overestimation for pK_a values > 9.5.

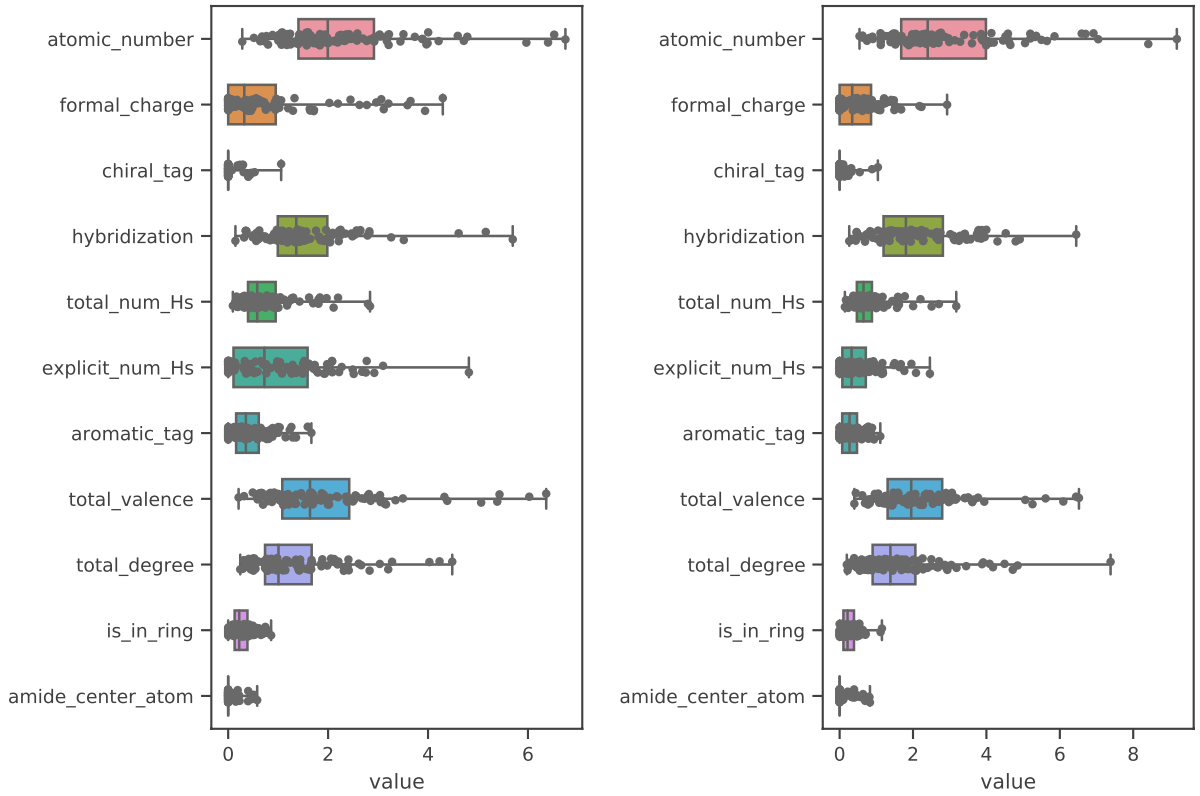
3.2.3 Model interpretation

Machine Learning models, especially neural networks, although very powerful in their ability to model and predict complex systems, tend to be very difficult to interpret. We know that the models are learning properties and characteristics about the data they are being trained on, but more often than not, the learned parameters and patterns are hard to reconcile with how humans might expect the information in the data could be learned. To at least shine a bit of light into how the GCNs used in this work function, respectively what impact the individual node and edge features have on the predictions, we used the `IntegratedGradients` method provided by the PyTorch Captum library [38]. For a random sample of 100 molecules from the train set, the importance of the features, meaning the extent of the effect on the prediction value of the model, of each individual node and edge (atom and bond) were calculated. Then for each molecule, the maximal absolute value for every feature, found in any of the nodes or edges of the said molecule, were taken and combined to a distribution of 100 values per feature. This has been done for all 6 types of GCN models and the results in form of box plots are displayed, including their discussions, in figures 3.4, 3.5, 3.6 and 3.7.



(a) attributions for *prot* data **(b)** attributions for *deprot* data

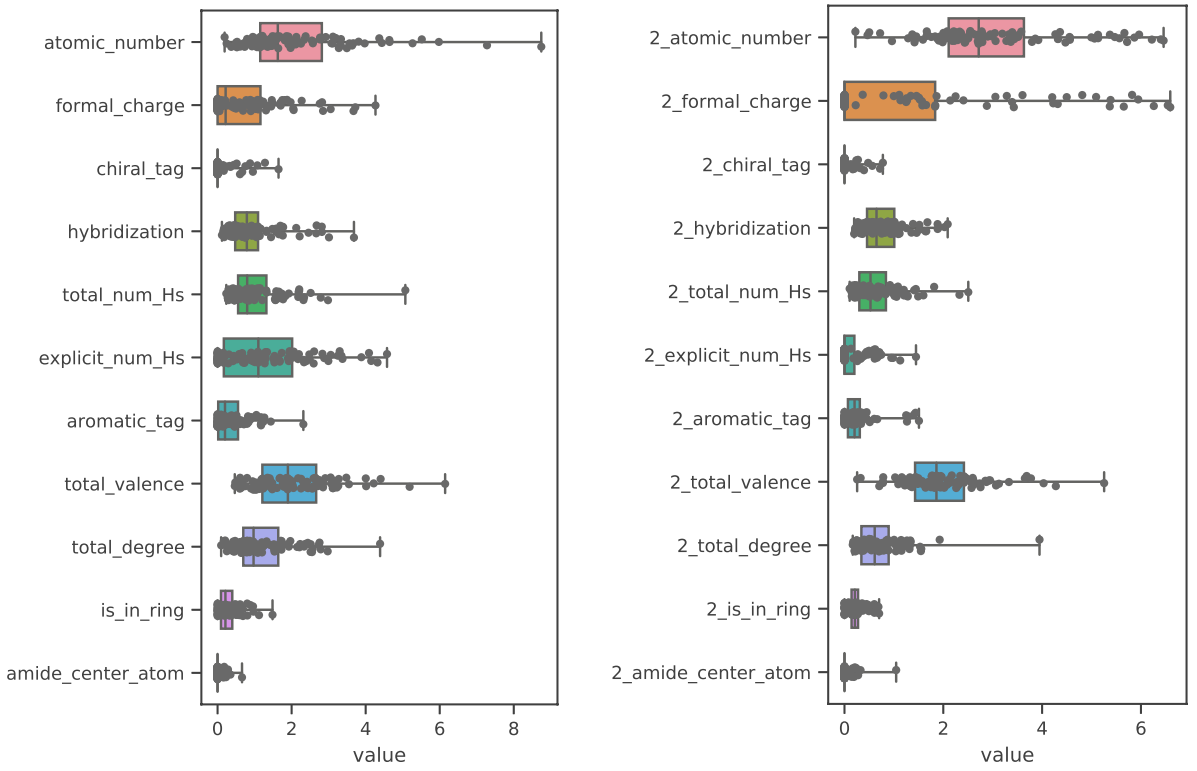
Fig. 3.4: Importances of GCN_prot_edge and GCN_deprot_edge models. Feature importances for all nodes and edges of a random sample of 100 molecules from the train set were calculated and filtered by the maximal value for each feature and molecule. We can see that the atomic number and the total valence have the most impact for both models. The impact of most features is scattered around medians between 0.5 to 1.5. chiral tag, amide centre atom and rotatability have the least impact with their distribution centring close to 0 impacts. Generally, it can be seen that the importance of the features of the deprotonated molecules scatter within a larger range, while the medians of both models are very similar.



(a) attributions for *prot* data

(b) attributions for *deprot* data

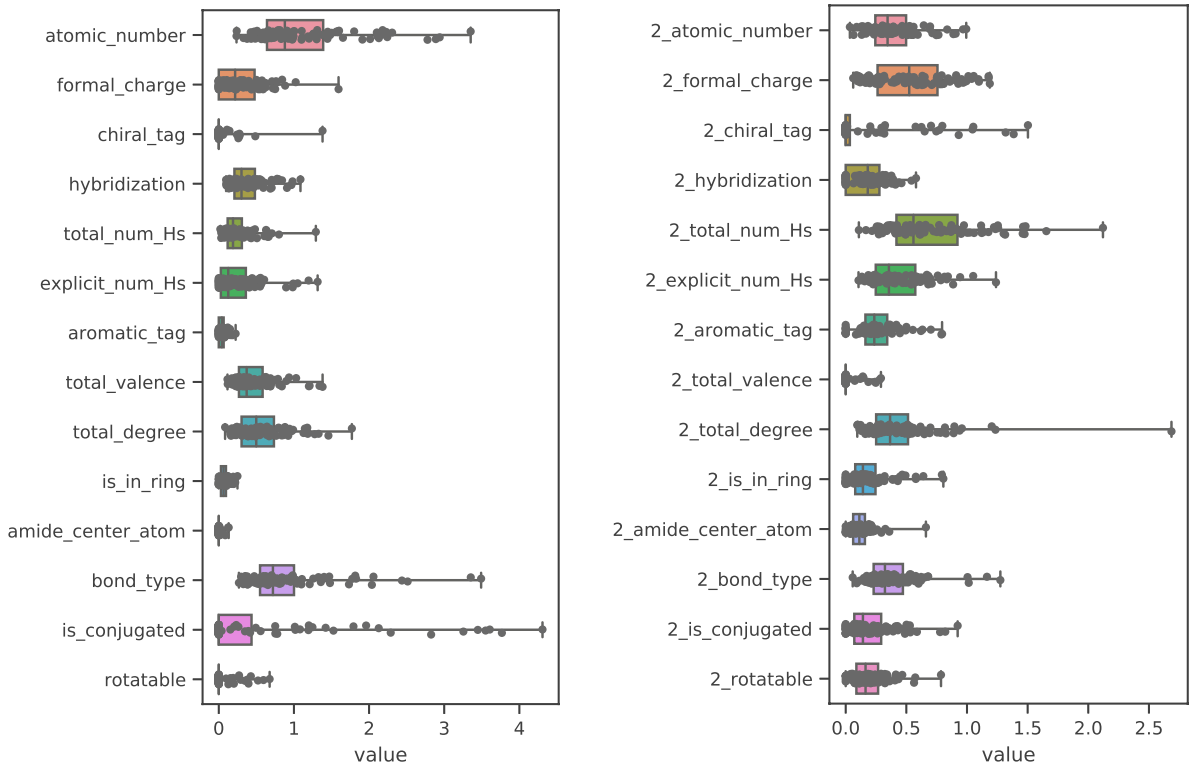
Fig. 3.5: Importances of GCN_prot_no-edge and GCN_deprot_no-edge models models. Feature importances for all nodes and edges of a random sample of 100 molecules from the train set were calculated and filtered by the maximal value for each feature and molecule. For these models with single-molecule inputs without edge features, we can observe, that the features atomic number, total valence, hybridization and total degree have the most impact. Again, the impacts of chiral tag and amide centre atom have the least impact, mainly distributing around 0. The range of feature impacts of the models using deprotonated data is higher than that of the models using protonated data.



(a) attributions for *prot* data

(b) attributions for *deprot* data

Fig. 3.6: Importances of GCN_pair_no-edge models. Feature importances for all nodes and edges of a random sample of 100 molecules from the train set were calculated and filtered by the maximal value for each feature and molecule. For both molecule inputs of the pair_no-edge models, similar distributions of importances as for the prot_no-edge model and deprot_no-edge model can be observed.



(a) attributions for *prot* data **(b)** attributions for *deprot* data

Fig. 3.7: Importances of GCN_pair_edge models. Feature importances for all nodes and edges of a random sample of 100 molecules from the train set were calculated and filtered by the maximal value for each feature and molecule. Compared to the impacts of the GCN_pair_no-edge model, each individual feature has less impact as the information is spread across a large variety of features. The most important features for the protonated input are atomic number and bond type with the medians of most other features ranging between 0.5 and 1. The impacts of the deprotonated inputs have predominately values below 1 and spread in a range of a bit more than half the size of the range for the protonated features.

3.2.4 Error Analysis

To further analyze the generalizability of the GCN models, we wanted to see, if their prediction performance was significantly dependent on the similarity of the test molecule to the training molecules. As a metric for estimating the similarity of two compounds, we use the Tanimoto coefficient. The Tanimoto similarity for two molecules is calculated from their morgan fingerprints by dividing the number of common fingerprint bits by the number of total bits of both molecules. The Tanimoto-similarity, therefore, ranges between values of 1 and 0 meaning, whereby a value of 1 indicates identical morgan fingerprints and 0 a pair of molecules with no fingerprint bit and therefore no substructure in common. For each molecule in the two test sets, the maximal Tanimoto-similarity coefficient it has with any molecule of the train set was calculated. Figures 3.8 and 3.9, show the RMSE values calculated for filtered test sets, containing only molecules with a maximum Tanimoto coefficient below a certain threshold. The RMSE values are plotted on the y-axis against the threshold Tanimoto coefficient on the x-axis. The discussion of the plot is included in their description.

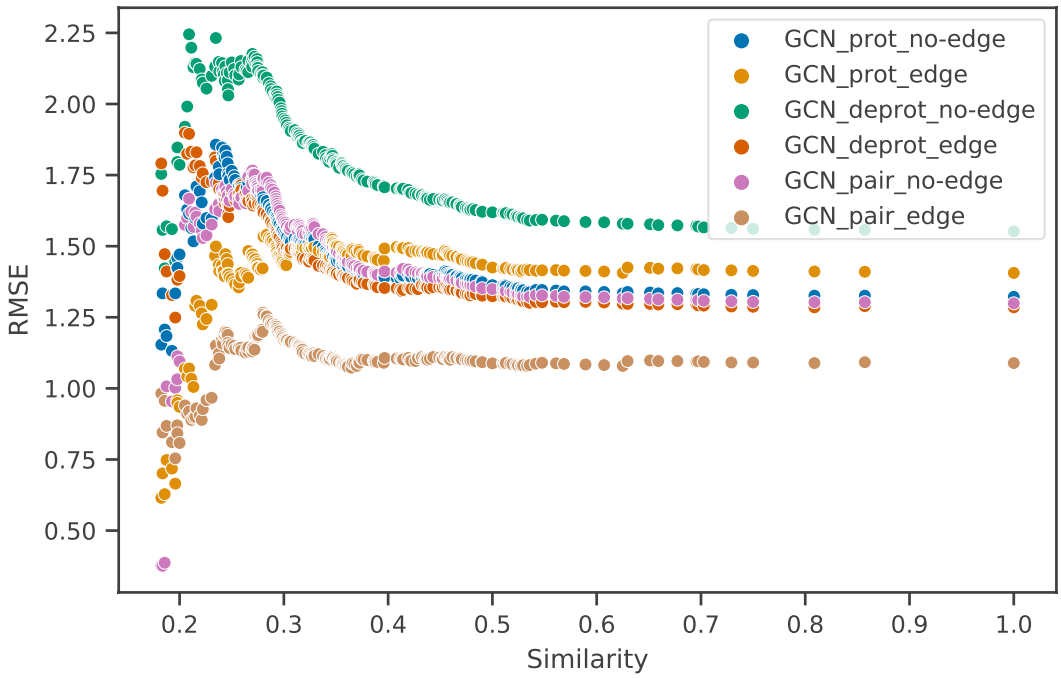


Fig. 3.8: Performance trend of GCN models depending on the maximum Tanimoto coefficient each Novartis test set molecule has with the training molecules . We can see that until a Tanimoto threshold of about 0.35, the RMSE values are only rising slightly, with each model type staying on its respective plateau, while at lower similarity thresholds, the RMSE values start to rise significantly. Close to the threshold for the lowest Tanimoto the RMSE for some models tends to lower again, possibly due to fewer samples constituting the threshold dataset.

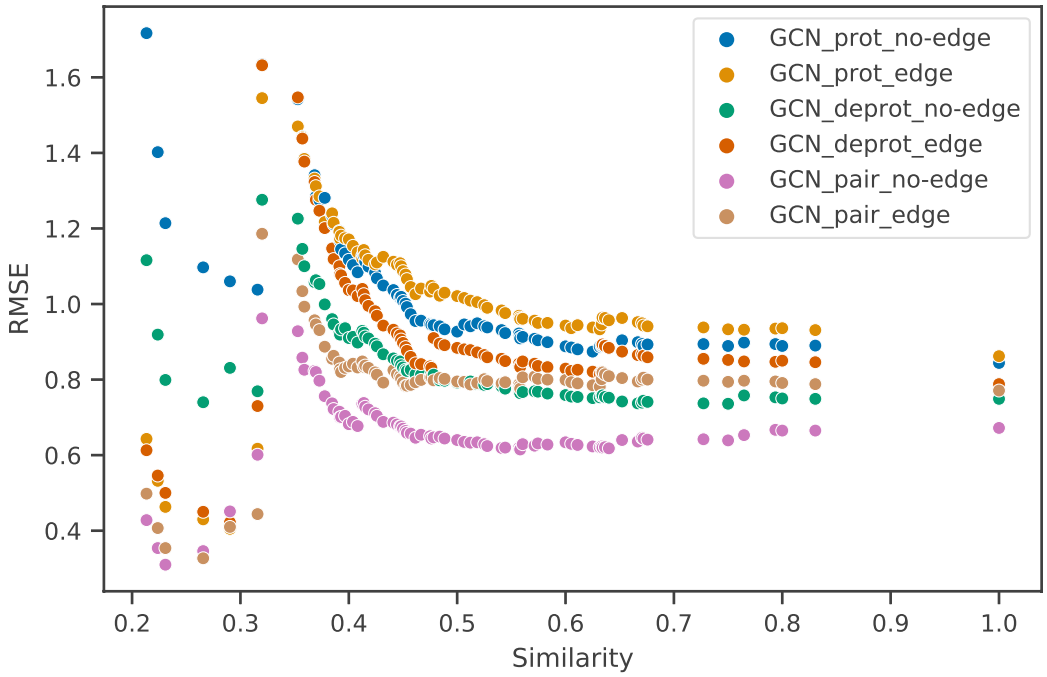
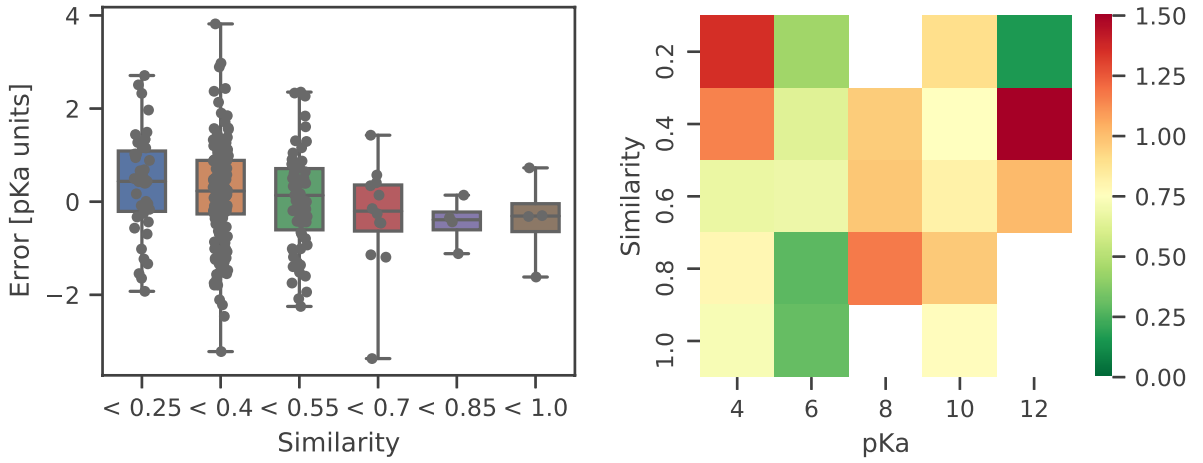


Fig. 3.9: Performance trend of GCN models depending on the maximum Tanimoto coefficient each molecule has with the training molecules (Literature). We can observe an exponential increase in the RMSE of all models, peaking around a similarity of 0.35. In the range of 0.35 to 0.2 the models yield quite divergent RMSE values which could coincide with their overall performance on the *Literature* test set.

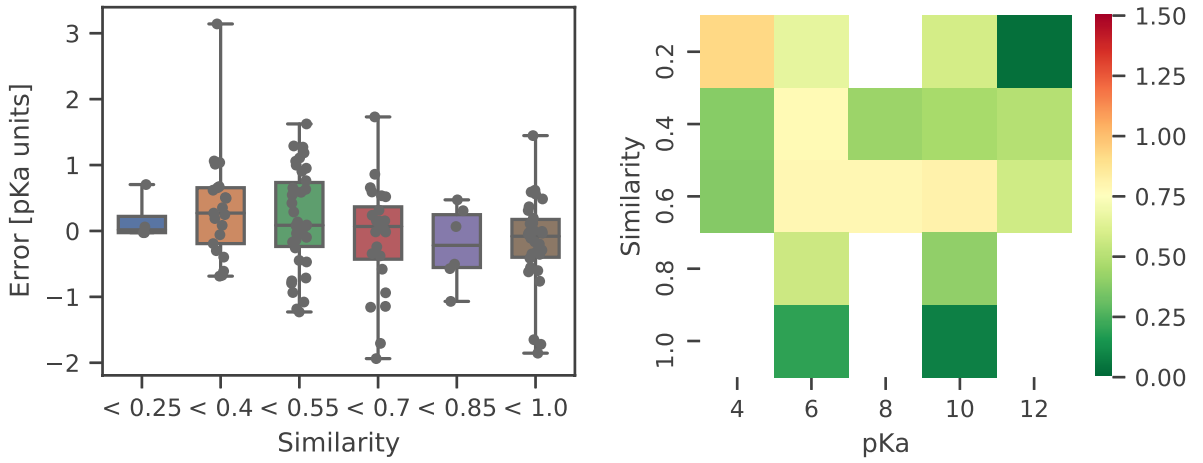
Figures 3.10 and 3.11 contain boxplots of prediction errors for different similarity ranges and heat maps showing the average prediction error of the GCN_pair_edge on a grid defined by pK_a and similarity for the predictions on the *Novartis* and *Literature* test set, respectively. The discussion of the plots is included in their descriptions.

Figures 3.12 and 3.13 shows and discusses the molecules corresponding to the 6 best and worst predictions of the GCN_pair_edge model on the *Novartis* and the *Literature* test set, respectively.



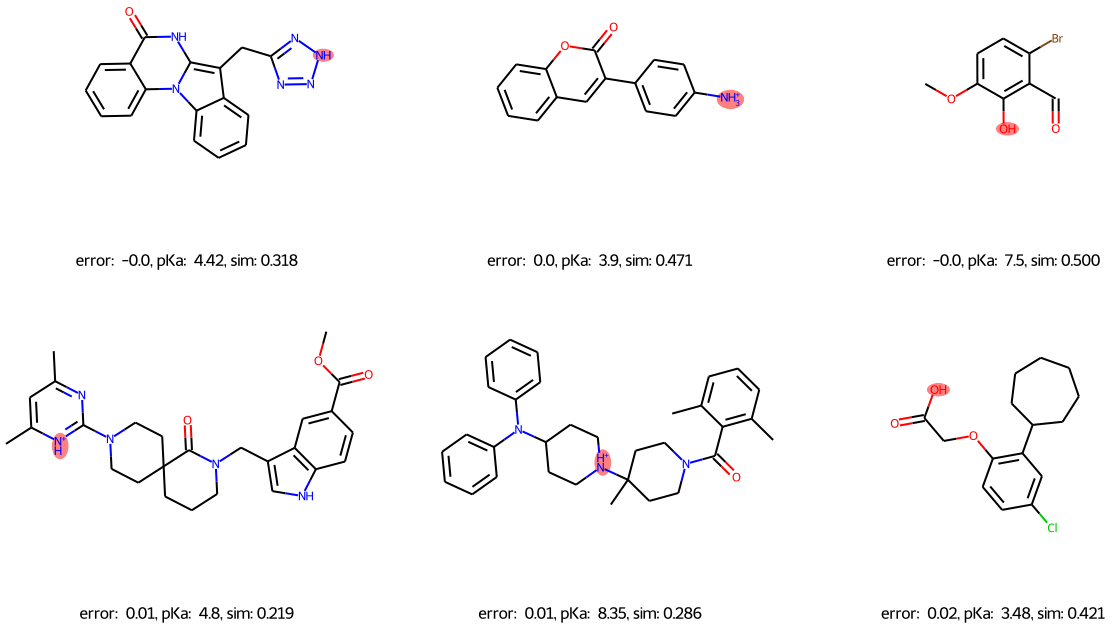
(a) error-similarity - plot **(b)** Novartis testset

Fig. 3.10: displays box plots of prediction errors on different similarity ranges **(a)** and a heat map showing the average prediction error of the GCN_pair_edge mode on a grid defined by pK_a and similarity **(b)** for the *Novartis* test set. The similarity ranges between 0.25 and 0.4 have the highest error spread, while the median of the ranges of 0.00 to 0.25 is the most skewed away from 0. The heat map indicates that molecules with low similarity and marginal pK_a values tend to yield poorer results on this dataset.

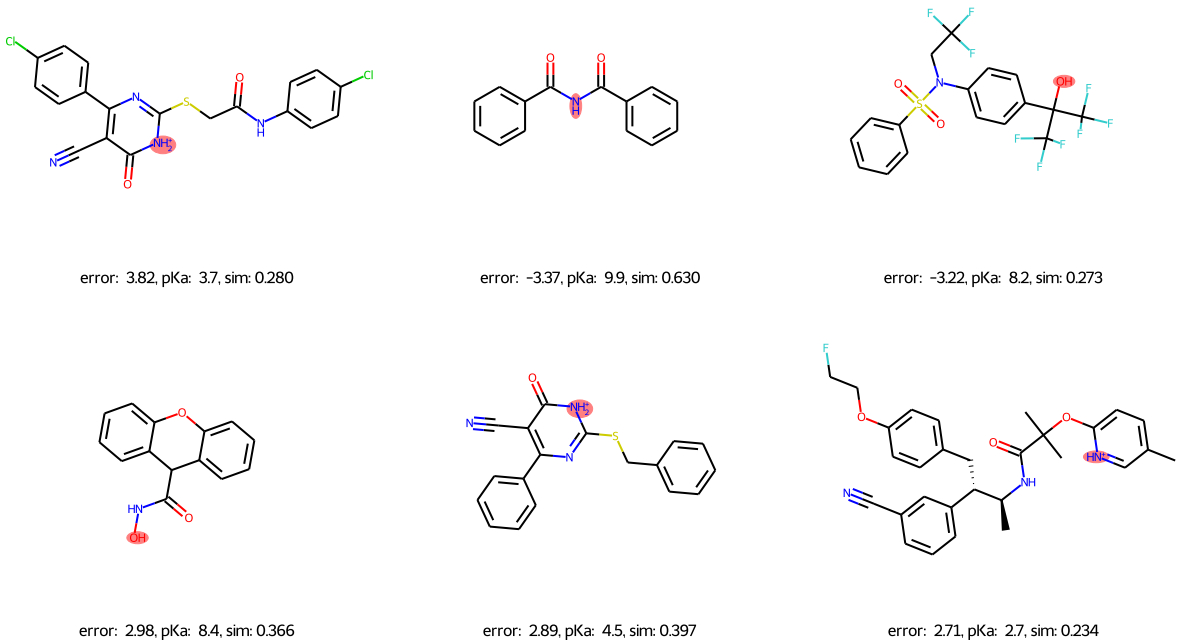


(a) error-similarity - plot **(b)** Novartis testset

Fig. 3.11: displays box plots of prediction errors on different similarity ranges **(a)** and a heat map showing the average prediction error of the GCN_pair_edge mode on a grid defined by pK_a and similarity **(b)** for the *Literature* test set. The similarity range between 0.4 and 0.25 has the highest error spread. the medians of all ranges stay within 0.5 units around 0. The heat map shows good overall performance in all quadrants withe a tendency for poorer predictions at a similarty < 0.2 and $pK_a < 4$.

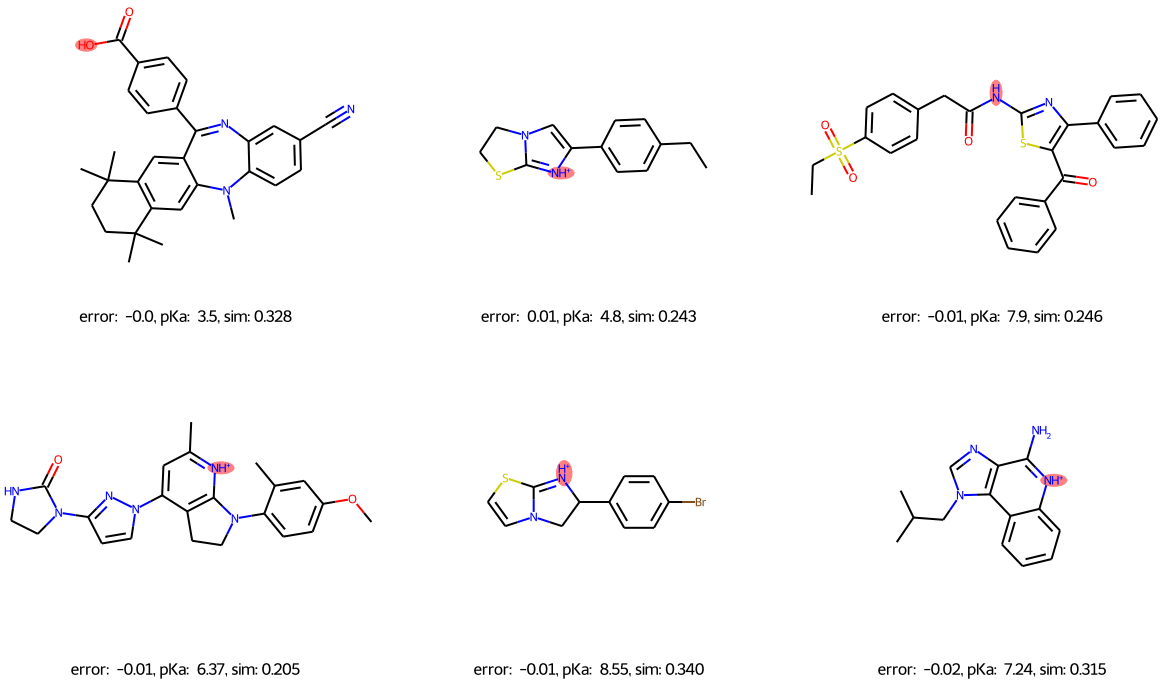


(a) best predictions

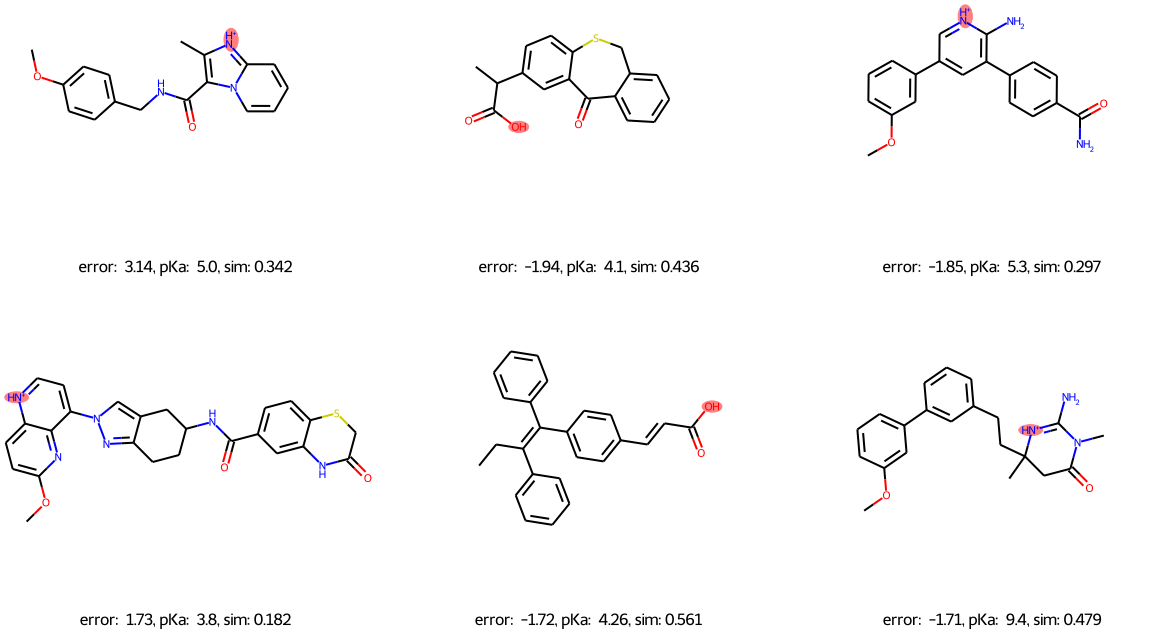


(b) Outliers

Fig. 3.12: shows the molecules corresponding to the 6 best **(a)** and worst **(b)** predictions of the GCN_pair_edge model on the *Novartis* test set. The reaction centre of each molecule is marked with a red-filled circle and the prediction error, the Tanimoto similarity and the empirical pK_a value are displayed. It is observable that the similarity to the train data is in a similar range for both the best and the worst predictions and is, therefore, no distinguishing feature. No obvious structural differences between the molecules of the best and the worst predictions can be noticed.



(a) best predictions



(b) Outliers

Fig. 3.13: shows the molecules corresponding to the 6 best **(a)** and worst **(b)** predictions of the GCN_pair_edge model on the *Literature* test set. The reaction centre of each molecule is marked with a red-filled circle and the prediction error, the Tanimoto similarity and the empirical pK_a value are displayed. As for the *Novartis* set, no correlation between similarity and outliers/best molecules and no obvious structural differences between the molecules of the best and the worst predictions can be observed.

3.2.5 GCN results summary

We were successfully able to show that using graph convolution networks on molecular graph data is a viable way to predict pK_a values. The GCN models achieved to considerably improve upon the best results of the random forest models of Baltruschat et al. [1], as well as on our own improved random forest baseline models. It can also be considered quite a success that our models were even able to outperform the results of the commercial program Marvin by Chemaxion. The interpretation of the models and the impact of its input features, as well as the interactions occurring for different combinations of features (node features only, node and edge features, etc.), is complex and remains to be further investigated. The approach of using both the protonated and deprotonated molecules of the acid-base reaction appeared fruitful in this work. In our case, the pair models also had a doubled amount of adjustable parameters, which could account in part for the enhanced performance compared to the single input model. Therefore, further experiments using equal amounts of parameters must be conducted to confirm the positive effect of the pair molecule inputs on the predictive performance of the GCN models.

Chapter 4

Outlook

Pathways for further improvement upon the results of this work can involve data, featurization and model architecture. In terms of data, it would be interesting to revise the training dataset provided by Baltruschat et al. [1]. The data mining on the *CHembl* database could be repeated to include the latest pK_a data and for the preprocessing, the filter criteria could be loosened by eliminating the filter steps involving the removal of all molecules that contain nitro groups, Boron, Selenium or Silicium or that break more than one of Lipinski's rules of five as this further increases the variety of molecules the models will be exposed to, thus hopefully further improving their generalizability.

Furthermore, the IUPAC pK_a database by as presented in its most recent version by Slater [14] contains 29,946 unique pK_a values for 12,189 unique molecules from primary sources categorized by the IUPAC precision criteria, aforementioned in section 1.3.2and would be a great addition to the train and test data used for further pK_a modelling.

In terms of featurization, there is probably still quite some room for experimentation and further improvements. On one hand, different combinations of the features used in this work could be tried out as there might be some redundancy or even features that oftentimes do not actually affect pK_a and can therefore lead to overfitting of the model to the training data, thereby downgrading the prediction performance on the test data. Also, additional features involving substructure matching on atoms and bonds (e.g. cyanide, imids, etc.), could be added.

There is a range of variations to be tried in terms of model architecture and Training, that remain to be tested for their impact on the quality of the present models. Using different numbers of convolutional and linear layers, different types of convolutional layers or varying training parameters like the learning rate, batch size or the loss criterion. Furthermore, the feature vector resulting from the pooling layer could additionally be fed with molecule-wide property metrics.

Also the fact that our models seem to be fulled trained after 500 epochs and cannot be significantly enhance by further training should be examined more closely.

Finally, further investigation into the efficiency of the *pair* input data approach, which uses two separate stacks of convolutional layers in parallel is definitely needed. As indicated in the discussion of the GCN results (3.2.5), it could easily be the case, that a model using single inputs of molecules, correctly protonated at pH=7.4, perform just as good or even better than the paired or the protonated inputs. However, for further generalisation and to predict pK_a values of multiprotic molecules, where the same input structure might have different ways to be protonated or deprotonated, corresponding to different pK_a values, the use of models that have both the protonated and the deprotonated molecule's graph as inputs, ensures that the model understands what reaction with which educt and product it is supposed to predict. These types of paired graph data models are also in no way restricted to the subject of pK_a prediction and could actually be used to model reaction constants or energies for all sorts of chemical reactions, even those involving more than one molecule on either side of the reaction as all educts and all products could be subsumed as subgraphs into a single graph respectively.

Abbreviations

ADMET absorption, distribution, metabolism, excretion, and toxicity

clogP logarithmic partition coefficient

CV cross-validation

ECFP extended connectivity fingerprint

GCN graph convolutional network

HSA human serum albumin

IUPAC International Union of Pure and Applied Chemistry

logD$_{pH=7.4}$ logarithmic distribution constant

PLS partial least squares

RFR random Forest regressor

Bibliography

[1] M. Baltruschat and P. Czodrowski. "Machine learning meets pK a". In: *F1000Research* 9 (2020).

[2] D. D. Perrin, B. Dempsey, and E. P. Serjeant. *pKa prediction for organic acids and bases*. Vol. 1. Springer, 1981.

[3] O. Exner. "The inductive effect: theory and quantitative assessment". In: *Journal of physical organic chemistry* 12.4 (1999), pp. 265–274.

[4] D. T. Manallack, R. J. Prankerd, E. Yuriev, T. I. Oprea, and D. K. Chalmers. "The significance of acid/base properties in drug discovery". In: *Chemical Society Reviews* 42.2 (2013), pp. 485–496.

[5] L. S. Schanker, D. J. Tocco, B. B. Brodie, and C. A. M. Hogben. "Absorption of drugs from the rat small intestine". In: *Journal of Pharmacology and Experimental Therapeutics* 123.1 (1958), pp. 81–88.

[6] K. Palm, K. Luthman, J. Ros, J. Gråsjö, and P. Artursson. "Effect of molecular charge on intestinal epithelial drug transport: pH-dependent transport of cationic drugs". In: *Journal of Pharmacology and Experimental Therapeutics* 291.2 (1999), pp. 435–443.

[7] M. Boisset, R. P. Botham, K. D. Haegele, B. Lenfant, and J. I. Pachot. "Absorption of angiotensin II antagonists in Ussing chambers, Caco-2, perfused jejunum loop and in vivo:: Importance of drug ionisation in the in vitro prediction of in vivo absorption". In: *European Journal of Pharmaceutical Sciences* 10.3 (2000), pp. 215–224.

[8] J. L. Castro, I. Collins, M. G. Russell, A. P. Watt, B. Sohal, D. Rathbone, M. S. Beer, and J. A. Stanton. "Enhancement of oral absorption in selective 5-HT1D receptor agonists: fluorinated 3-[3-(piperidin-1-yl) propyl] indoles". In: *Journal of medicinal chemistry* 41.15 (1998), pp. 2667–2670.

[9] M. P. Gleeson. "Generation of a set of simple, interpretable ADMET rules of thumb". In: *Journal of medicinal chemistry* 51.4 (2008), pp. 817–834.

[10] R. J. Prankerd. "Critical compilation of pKa values for pharmaceutical substances". In: *Profiles of drug substances, excipients and related methodology* 33 (2007), pp. 1–33.

[11] B. Pathare, V. Tambe, and V. Patil. "A review on various analytical methods used in determination of dissociation constant". In: *Int. J. Pharm. Pharm. Sci* 6.8 (2014), pp. 26–34.

[12] S. Babić, A. J. Horvat, D. M. Pavlović, and M. Kaštelan-Macan. "Determination of pKa values of active pharmaceutical ingredients". In: *TrAC Trends in Analytical Chemistry* 26.11 (2007), pp. 1043–1061.

[13] J. Reijenga, A. Van Hoof, A. Van Loon, and B. Teunissen. "Development of methods for the determination of pKa values". In: *Analytical chemistry insights* 8 (2013), ACI–S12304.

[14] A. M. Slater. "The IUPAC aqueous and non-aqueous experimental pKa data repositories of organic acids and bases". In: *Journal of computer-aided molecular design* 28.10 (2014), pp. 1031–1034.

[15] A. C. Lee and G. M. Crippen. "Predicting p K a". In: *Journal of chemical information and modeling* 49.9 (2009), pp. 2013–2033.

[16] G. C. Shields. *Computational approaches for the prediction of pKa values*. CRC Press, 2019.

[17] D. Rogers and M. Hahn. "Extended-connectivity fingerprints". In: *Journal of chemical information and modeling* 50.5 (2010), pp. 742–754.

[18] H. L. Morgan. "The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service." In: *Journal of Chemical Documentation* 5.2 (1965), pp. 107–113.

[19] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[20] C. C. Margossian. "A review of automatic differentiation and its efficient implementation". In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 9.4 (2019), e1305.

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

[22] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[23] W. McKinney et al. "pandas: a foundational Python library for data analysis and statistics". In: *Python for high performance and scientific computing* 14.9 (2011), pp. 1–9.

[24] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, et al. "Array programming with NumPy". In: *Nature* 585.7825 (2020), pp. 357–362.

[25] G. Landrum et al. "RDKit: Open-source cheminformatics". In: (2006).

[26] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019), pp. 8026–8037.

[28] M. Fey and J. E. Lenssen. "Fast graph representation learning with PyTorch Geometric". In: *arXiv preprint arXiv:1903.02428* (2019).

[29] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in science & engineering* 9.03 (2007), pp. 90–95.

[30] M. L. Waskom. "Seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021.

[31] T. Sander, J. Freyss, M. von Korff, and C. Rufener. "DataWarrior: an open-source program for chemistry aware data visualization and analysis". In: *Journal of chemical information and modeling* 55.2 (2015), pp. 460–473.

[32] A. Gaulton, L. J. Bellis, A. P. Bento, J. Chambers, M. Davies, A. Hersey, Y. Light, S. McGlinchey, D. Michalovich, B. Al-Lazikani, et al. "ChEMBL: a large-scale bioactivity database for drug discovery". In: *Nucleic acids research* 40.D1 (2012), pp. D1100–D1107.

[33] A. Avdeef. *Absorption and drug development: solubility, permeability, and charge state*. John Wiley & Sons, 2012.

[34] M. Morgenthaler, E. Schweizer, A. Hoffmann-Röder, F. Benini, R. E. Martin, G. Jaeschke, B. Wagner, H. Fischer, S. Bendels, D. Zimmerli, et al. "Predicting and tuning physicochemical properties in lead optimization: amine basicities". In: *ChemMedChem: Chemistry Enabling Drug Discovery* 2.8 (2007), pp. 1100–1115.

[35] F. Luan, W. Ma, H. Zhang, X. Zhang, M. Liu, Z. Hu, and B. Fan. "Prediction of p K a for neutral and basic drugs based on radial basis function Neural networks and the heuristic method". In: *Pharmaceutical research* 22.9 (2005), pp. 1454–1460.

[36] C. Dardonville. "Automated techniques in pKa determination: low, medium and high-throughput screening methods". In: *Drug Discovery Today: Technologies* 27 (2018), pp. 49–58.

[37] C. Liao and M. C. Nicklaus. "Comparison of nine programs predicting p K a values of pharmaceutical substances". In: *Journal of chemical information and modeling* 49.12 (2009), pp. 2801–2812.

[38] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, et al. "Captum: A unified and generic model interpretability library for pytorch". In: *arXiv preprint arXiv:2009.07896* (2020).