UNIVERSITY OF WEST ATTICA
FACULTY OF ENGINEERING
DEPARTMENT OF BIOMEDICAL ENGINEERING

Prediction of the retention time of natural product metabolites using transfer learning strategies.

Vasiliki Katsara
Student ID: 19088040

Supervisor
Matsoukas Minos, Assistant Professor

Athens, 10/09/2024

The Defense Committee

Supervisor	Member of the examination committee	Member of the examination committee
Dr. Minos Matsoukas	Dr. Spyridon Kostopoulos	Dr. Emmanouil Athanasiadis
MINOS TIMOTHEOS MATSOUKAS Digitally signed by MINOS TIMOTHEOS MATSOUKAS Date: 2024.10.13 19:01:27 +03'00'	Digitally signed by Spyridon Kostopoulos Date: 2024.10.14 11:14:59 +03'00'	Digitally signed by EMMANOUIL ATHANASIADIS Date: 2024.10.14 12:44:20 +03'00'

DIPLOMATIC THESIS AUTHOR'S STATEMENT

The undersigned **Vasiliki Katsara** of **Chariton**, with student ID number **19088040**, student of the Department of **Biomedical Engineering**, Faculty of **Engineering** of the University of West Attica, hereby declare responsibly that:

«I am the author of this thesis, and any assistance I received in its preparation is fully acknowledged and referenced in the thesis. Furthermore, any sources from which I have used data, ideas, or words, whether directly or paraphrased, are cited in their entirety, with complete references to the authors, the publishing house, or the journal, including any sources that may have been used from the Internet. Additionally, I affirm that this work has been written exclusively by me and constitutes intellectual property belonging to both me and the institution.

Violation of the above academic responsibility constitutes substantial grounds for the revocation of my diploma».

Date

10/09/2024

The declarant

Vasiliki Katsara

Abstract

Retention time (RT) prediction in chromatography can play an important role for numerous analytical applications, including drug discovery and environmental monitoring. This study aims to enhance RT prediction accuracy by employing deep learning techniques, particularly focusing on transfer learning to adapt models trained on synthetic compounds acquired by High Pressure Liquid chromatography–Mass Spectrometry (HPLC-MS) to predict RTs for natural products in different chromatographic methods. We utilized the extensive METLIN Small Molecule Retention Time (SMRT) dataset, comprising over 80,000 synthetic compounds, to train a deep neural network (DNN). This model was then fine-tuned on smaller datasets of natural products, from the RepoRT database using a two-stage transfer learning approach. Initially, the DNN's upper layers were frozen to retain knowledge about high level features while training on the new data. Subsequently, all layers were unfrozen for further training with a reduced learning rate, ensuring both general and unique patterns were captured. Hyperparameter optimization was conducted using Optuna, leveraging a 5-fold nested cross-validation to ensure robust performance. The evaluation metrics that we computed were the Mean Absolute Error (MAE), the Median Absolute Error (MedAE) and Mean Absolute Percentage error (MAPE). Transfer learning was then compared with new trained DNNs directly trained on the RepoRT database and showed that the strategy was successful according to the MAE and MadAE metric, although not according to the MAPE. We decided to remove the outliers and noticed that with the cleared data transfer learning performed better considering all the metrics. In the future, it will be necessary to refine this strategy to improve its performance, either by testing it on the same datasets or by incorporating additional data.

Acknowledgements

I would like to express my deepest gratitude to Professor Mino Matsouka from the University of West Attica and Guillermo Ramajo Fernández and Abraham Otero Quintana from Universidad San Pablo CEU in Madrid. Their invaluable guidance, support, and encouragement throughout this research journey have been crucial to the completion of this study. Their patience, and belief in my potential to perform this research have advanced my growth and development in the field. Their expertise, insightful feedback, and dedication were instrumental in shaping the direction and quality of my work. I am deeply thankful for the privilege of working alongside them.

I am also immensely grateful to my family and friends for the endless patience, encouragement, and understanding. Their unwavering support has been a constant source of motivation.

Contents

Tables of figures

1. Introduction

1.1 Metabolomics

Metabolomics is one of the fastest evolving fields of life science that looks for relationship between metabolites and physiological and pathological changes. It is defined as the comprehensive characterization and the quantitative and qualitative assessment of the metabolome [1] [2].

The metabolome is a set of metabolites and small molecule chemicals (with a molecular weight<1500 Da) that can be found in a cell, a biofluid, an organ, or an organism [2] [3]. There are two types of metabolites, the primary and secondary metabolites. The primary are the endogenous compounds synthesised by the host genome and are directly related to the growth and reproduction of organisms. The secondary are the xenobiotic compounds from the diet and the environment that are not directly related to growth and reproduction but have a crucial role in the organism's interactions with its environment. Metabolites are the reactants and products of metabolism and therefore the changes in the concentrations of specific groups of metabolites give us information about the reaction of the systems to various interferences (environmental, genetic or other) [1] [4] [5]. In the end, we understand that this scientific field allows scientists to monitor and track changes within the organism and reflects the phenotype response and the physiological state (physiological status in biological system) of an organism [3]. Metabolomics is the latest branch of the ‘omics’ science list (cascade) after genomics, proteomics, and transcriptomics. It is used in biomaterial production, in disease screening, in biomedical research, in drug discovery and development, in veterinary studies, and in food and nutritional analysis [3] [6].

1.1.1 The Metabolomics Workflow

There are two types of metabolomic studies, the targeted and untargeted studies. The targeted has a prior hypothesis and specific metabolites of interest while the untargeted one does not have a prior hypothesis and has to measure more metabolites. The latest one is very useful if the hypothesis of the first one fails, which could happen because of the existence of the two types of metabolites that we discussed previously [7]. For a successful study there is a specific research process and includes an experimental design, sample preparation (source management), collection, and metabolite extraction; data extraction/acquisition(to visualise the map), preprocessing (to filter the noise, overlapping the peak resolution, peak alignment normalization etc.) and pattern recognition with supervised or unsupervised models; statistical analysis; and metabolite identification to provide biological meaning to the dataset, see in Figure 1.1 a schematic of the typical metabolomic workflow [2] [7] [6].

During the data acquisition an analytical platform is required. Most of the times this can be a Mass Spectrometer (MS) or a Nuclear Magnetic Resonance (NMR), and sometimes a Fourier Transform Infrared Spectroscopy (FTIR). When using a MS it is very common approach to incorporate chromatographic techniques, such as Liquid Chromatography (LC–MS), most of the times High-Performance Liquid Chromatography (HPLC MS) in particular, Gas Chromatography (GC–MS) or even coupled Capillary Electrophoresis MS (CE–MS) [6]. In

our study the dataset, as we will see on the next chapters, was acquired by the implementation of Coupled chromatography LC-MS system.

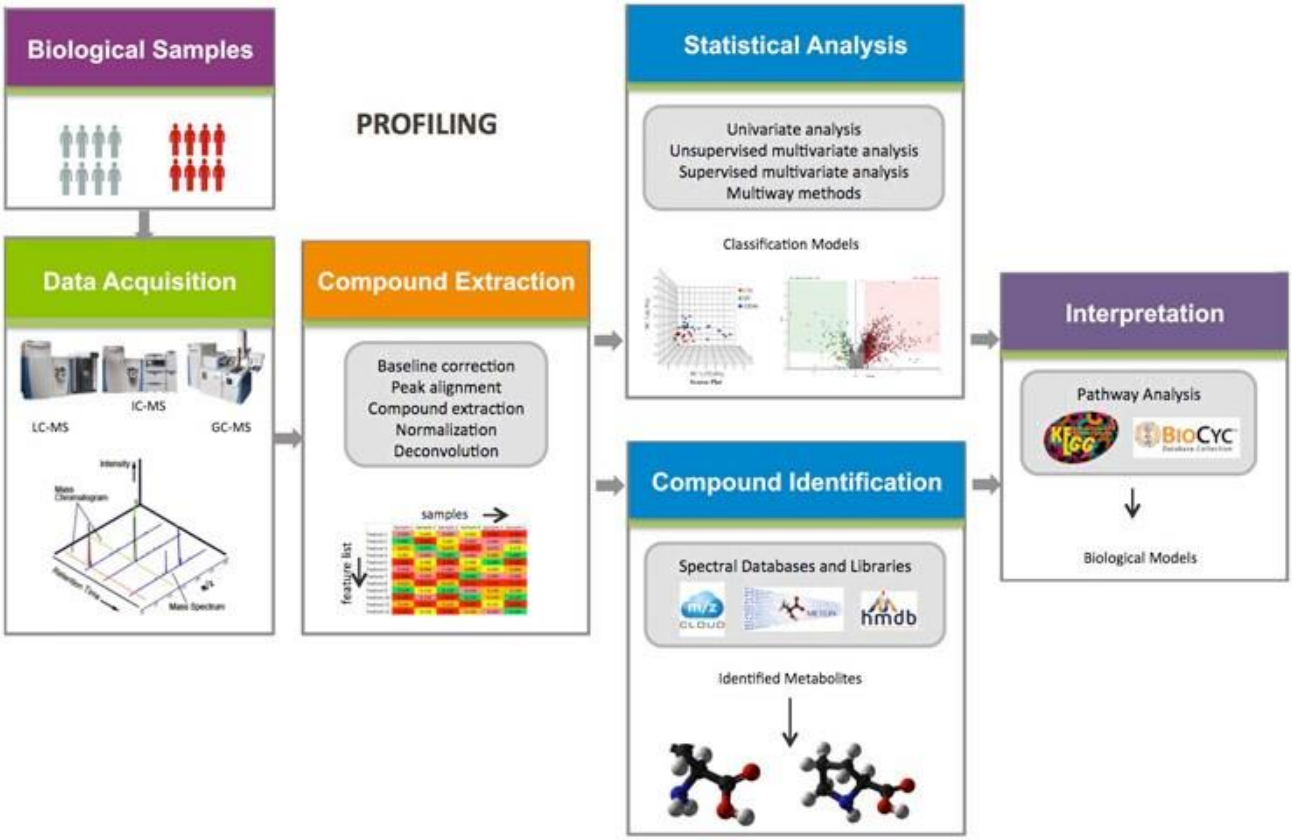

Figure 1.1 Metabolomic workflow [8].

1.2 Chromatography

Chromatography is an analytical chemistry technique for the physical separation and identification of mixture compounds and organic substances, and it is used in many scientific fields. The separation happens because of the different interactions of compounds due to the physicochemical affinity in two immiscible phases, one stationary and one mobile. In other words, the components of mixtures have different properties (boiling point, polarity, hydrophobicity, electric charges (for ionic compounds), size, shape and weight of molecules/atoms, solubility), the different properties lead to different interactions with the two phases, with the result that they do not leave the column at the same time [9].

The mobile phase consists of the solvent or the solvent system and the sample that will get separated. This phase is liquid, gas or supercritical fluid. It moves through the stationary phase and the components get separated because they travel at different rates based on their interactions. The stationary phase consists of porous solid, or a viscous liquid fixed on a solid

surface [9] [10]. Chromatographic techniques can be grouped, by various criteria [9]. You can see more examples in Figure 1.2.

1) **Depending on the nature of the mobile phase**
   - Liquid chromatography (LC): The mobile phase is liquid, and the stationary phase is solid or liquid on solid. For example, Liquid-solid-chromatography (LSC), Liquid-liquid-chromatography (LLC)
   - Gas chromatography (GC): The mobile phase is gas, and the stationary phase is solid or liquid on solid. For example, Gas-solid-chromatography (GSC) and Gas-liquid-chromatography (GLC)
   - Supercritical fluid chromatography (SFC): The mobile phase is a supercritical fluid.
2) **Depending on the nature and form of the stationary phase:**
   - Column chromatography: Packed columns, open capillary or tubular and high-performance LC (HPLC)
   - Planar chromatography: Thin layer chromatography (TLC), Paper chromatography (PC)
3) **Depending on the separation mechanism**
   - Adsorption (hydrophobic compounds)
   - Ion exchange (polar compounds)
   - Partition (slightly polar compounds)
   - Molecular exclusion
   - Gel permeation or gel filtration (macromolecular compounds)
   - Affinity
4) **Depending on the method of introduction and movement of the sample**
   - Frontal
   - Displacement
   - Elution

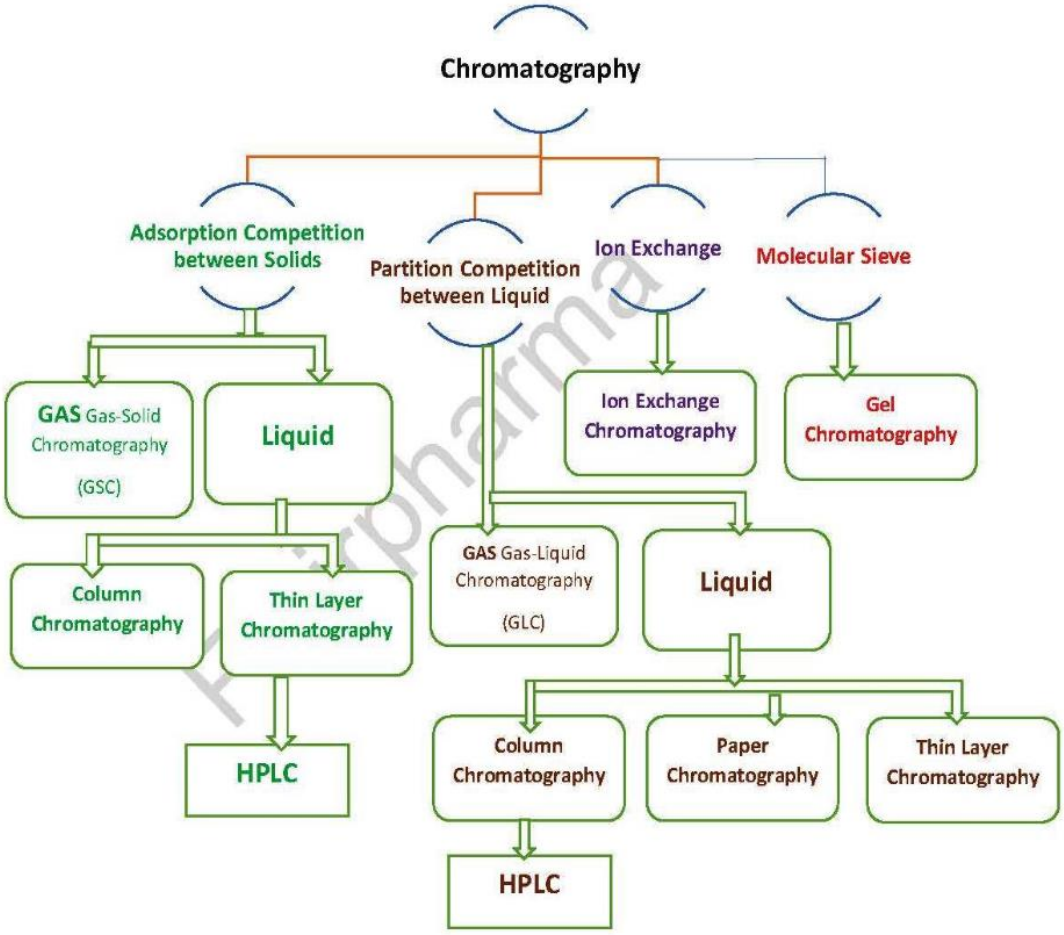

Figure 1.2 Classification of basic chromatographic techniques [11]

Detectors and Chromatogram

A very important part of the chromatographic system is the detector. The detector identifies, qualitatively and quantitatively, the successive fractions leaving the column. They produce a different signal when one substance reaches the detector. Detectors must be sensitive, meaning that they must be able to separate any noisy signals from other components of the sample and be able to efficiently separate the compounds (this is affected by the elution times between the peaks and how broad the peaks are), and have short response time. The most common detectors are the following [9]:

- UV-VIS spectrophotometric detectors.
- Fluorescence detectors.
- Detectors based on the different refractive index of successively exiting fractions.
- Detectors based on the different electrical conductivity of successively exiting fractions.
- Detectors based on the different conductivity of successively exiting fractions.
- Mass spectrometers. They provide important qualitative and quantitative information about the substances in a sample. Thus, perfect chromatographic separation is not required while signal-to-noise ratio is increased, and sample preparation requirements are reduced.

A chromatogram is a graph that displays the response of the detector in relation to the elution time. Each compound corresponds to a peak in the chromatogram. The peaks are driven to chart recorders, magnetic media or monitors, after being amplified. The area of the pulse is a measure of its relative concentration, while the relative position of the peaks (retention time) is a measure of the nature of the detected component [10], see Figure 1.3.

The retention time (RT or t_r) is the time required for a certain substance to "travel" through the column from its entrance to the detector. The measurement starts when the sample is injected into the column. It is a characteristic constant of a substance under strictly defined experimental conditions, like the pressure exerted on the column, the type of stationary phase, the composition of the mobile phase, and the column temperature. [12]. The RT is an important parameter for the compound identification because an unknown peak generated by the system can be matched to a specific compound if we know the RT [9] [10].

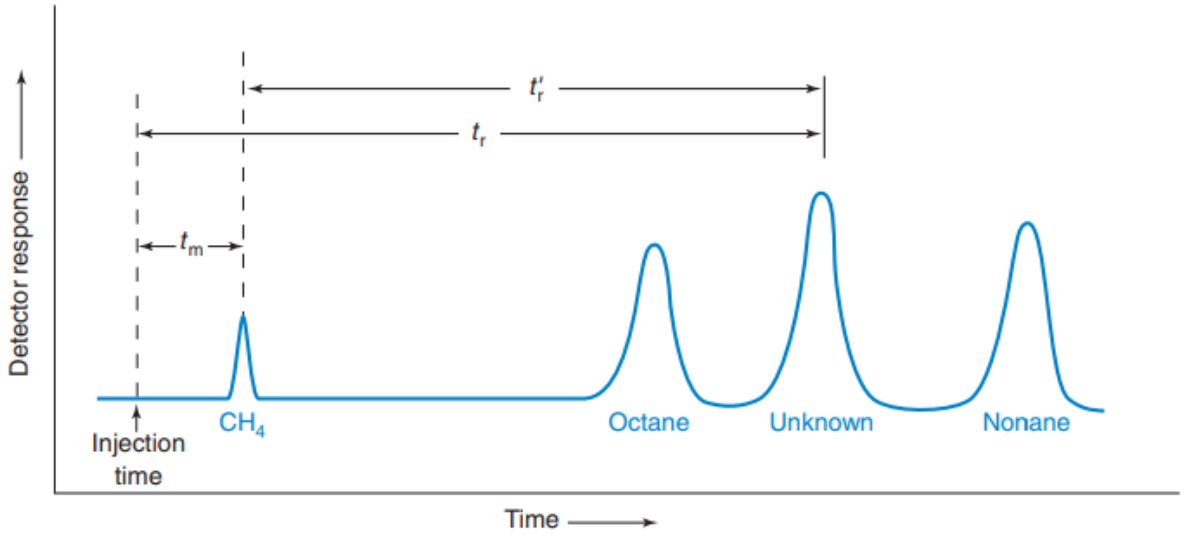

Figure 1.3 Schematic chromatogram of retention time [10].

1.2.1 HPLC

LC, and particularly HPLC, offers more simple sample preparation and instrumentation, and lower temperatures compared to other chromatographic methods e.g. Gas Chromatography (GC). For the previous reasons, it is widely used clinical practice [9]. HPLC is an advanced form of liquid column chromatography with better sensitivity, sample analysis speed, etc. The solvent is passed through the column under high pressure as the name indicates. The mobile phase is liquid which does not flow into a column under the influence of gravity but with the help of a pump. The stationary phase is solid, and the sample has to be in a liquid form. There are two types of HPLC the Normal Phase (NP-HPLC) and the Reverse Phase (RP-HPLC). In normal phase the stationary phase consists of Silicon dioxide (SiO_2) or Aluminium oxide (Al_2O_3) and the mobile phase is non-polar which results in better adsorption (higher eluent

strength) of the more polar compounds in the stationary phase making them the last to leave the column. In the reverse phase, which is the most commonly used technique, the stationary phase is non-polar and the mobile phase is polar, the non-polar molecules of the sample bond with more strength to the hydrocarbon chains of the stationary phase, while the polar molecules move faster and leave the column first (less polar solvent has higher eluent strength) [5] [12].

Another important factor is the way the mobile phase is supplied. There are two possibilities. First, the isocratic elution, the mobile phase and solvent that have constant composition and pH, get administered with a constant supply to the column throughout the chromatography. Additionally, the gradient elution, a component of the mobile phase and solvent changes continuously with respect to the time during the chromatography [5] [10] [12].

As in any other type of chromatography, the identification of compounds from the sample in HPLC can be feasible by comparing the retention characteristics, such as the retention time of an unknown compound with those of a reference, under the same experimental conditions [13].

The basic components of HPLC are the following, you can see a simple schematic in Figure 1.4 [5] [9]:

- The administration system with the corresponding high-pressure pump.
- The sample injection system.
- The column and possibly the thermostatic oven.
- The detector and the recorder.

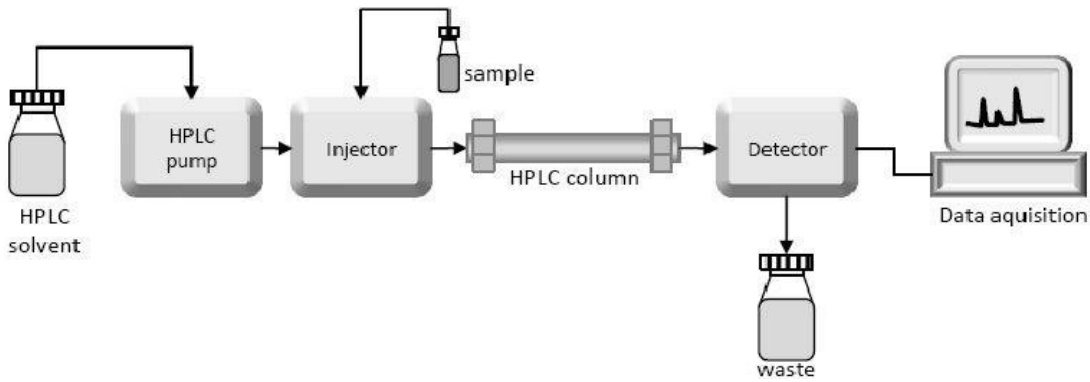

Figure 1.4 HPLC instrumentation [14].

1.3 Mass spectrometry (MS)

Mass spectrometry is one of the widely used methods of analytical chemistry used for qualitative and quantitative analysis of small samples. Main field of application is life sciences, food, environmental sciences and omics technologies. Mainly used for determination of atomic/molecular weight, separation of elements in a sample and determination of number of

specific elements, of molecular composition, for information on structural characteristics of compounds. Common application today, as we already discussed previously, is as a detector in liquid and gas chromatography for more accurate separation (qualitative and quantitative) [5].

The process of a correct mass spectrometry follows:

- Sample introduction and venting: The molecules that will be separated by MS must be in a gaseous state. When the MS is connected to a chromatography system, the sample introduction is done with a special coupling device depending on the type of chromatography [9].
- Ionization: The sample molecules that are in gas phase are converted into ions with positive, usually, charge by one of the techniques listed in Table 1.1 [9].

Table 1.1 Types of ionization **[15]**

Type	Phase	Fragmentation
Inductively Coupled Plasma (**ICP**)	Liquid feed	Gives elements
Electron Impact (**EI**)	gas	lots
Chemical Ionization (**CI**)	gas	some
Electrospray (**ESI**)	liquid	very little
Atmospheric Pressure Chemical Ionization (**APCI**)	liquid	some
Matrix Assisted Laser Desorption Ionization (**MALDI**)	solid	some
Desorption Electrospray Ionization (**DESI**)	Portable	Very little

- Ion separation based on mass-to-charge ratio (m/z): The ion beam is passed through a "mass separator or filter" and each ion, depending on its m/z, deviates differently from the original beam direction. The path radius depends on the m/z ratio. Heavier isotopes deviate less. The types of separation are mentioned in Table 1.2 [5] [9].

Table 1.2 Separation types **[15]**

Type	Speed	Basis	Cost
Magnetic Sector	slow	Acceleration in magnetic field	moderate
Double Focusing	slow	Magnetic plus electric field	high
Quadrupole	fast	Passage through ac electric field	moderate
Ion trap	fast	Orbit in quadrupole	moderate
Time-of-Flight	very fast	Time to travel through tube	moderate
Newer High Resolution	varies	Various, usually involving orbits	high

- Ion detection: With a suitable detector, the electric current provided by the ions with different m/z is measured and the spectra are analyzed, which are diagrams of signal intensity or relative abundance, see Figure 1.5. The different detector types are introduced in the Table 1.3 [5] [9].

Table 1.3 Detectors **[15]**

Type	Internal Amplifications?	Uses
Faraday Cup	No	Isotope Ratio MS
Electron Multiplier	Yes	Fairly Common
Microchannel plate	Yes	Higher end instruments
Induction	No	Used in FT-ICR

- Vacuum system. The spectrograph is under vacuum which is created by internal and external vacuum pumps. High vacuum is necessary in mass spectrometry to avoid molecular collisions during the ion separation process [5] [9].

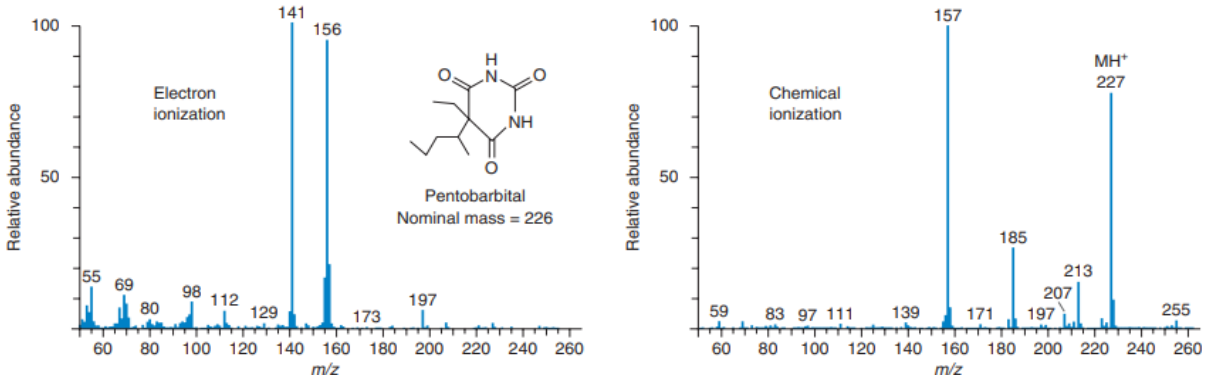

Figure 1.5 Mass spectra of the sedative pentobarbital obtained through electron ionization (left) and chemical ionization (right) [10].

1.4 Chromatography and mass spectrometry

Mass spectrometers as we previously discussed are used as detectors in chromatography. They provide important qualitative and quantitative information about the substances in a sample. The MS has high selectivity and thus, perfect chromatographic separation is not required, the signal-to-noise ratio is increased, and sample preparation requirements are reduced [10].

Liquid chromatography is used to separate mixtures with multiple components, while mass spectrometry offers spectral data that can aid in identifying each separated component [5]. Consequently, LC-MS is applicable across various fields, including biotechnology, pharmacokinetics [16], proteomics [17] [18]and metabolomics [19], drug development [20] etc. You can see a simple schematic of the HPLC-MS instrumentation in Figure 1.6 and the LC-MS spectrum in Figure 1.7.

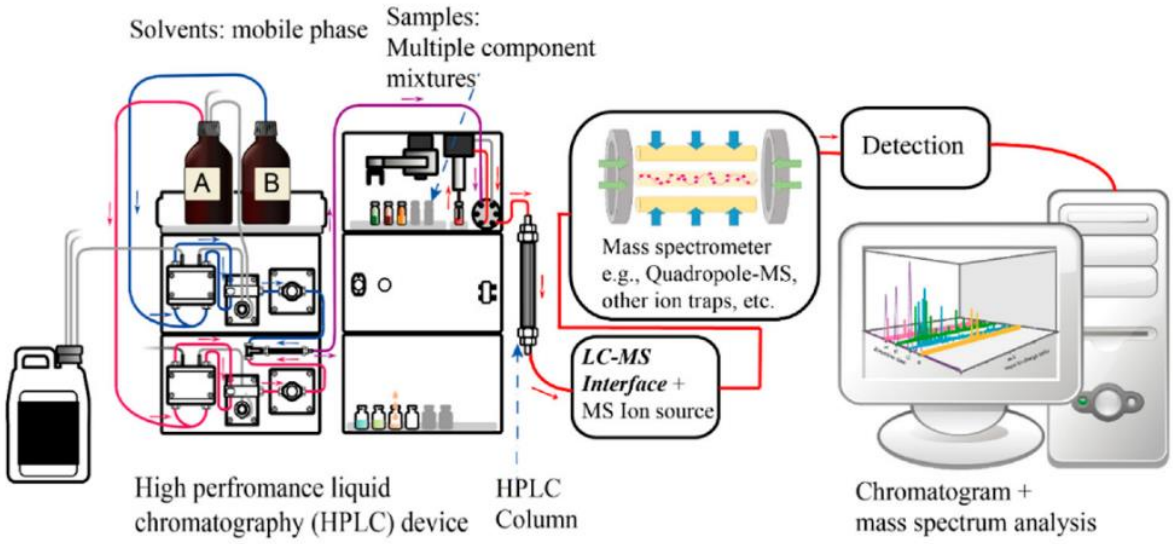

Figure 1.6 A diagram of an HPLC-MS spectrometry system [21].

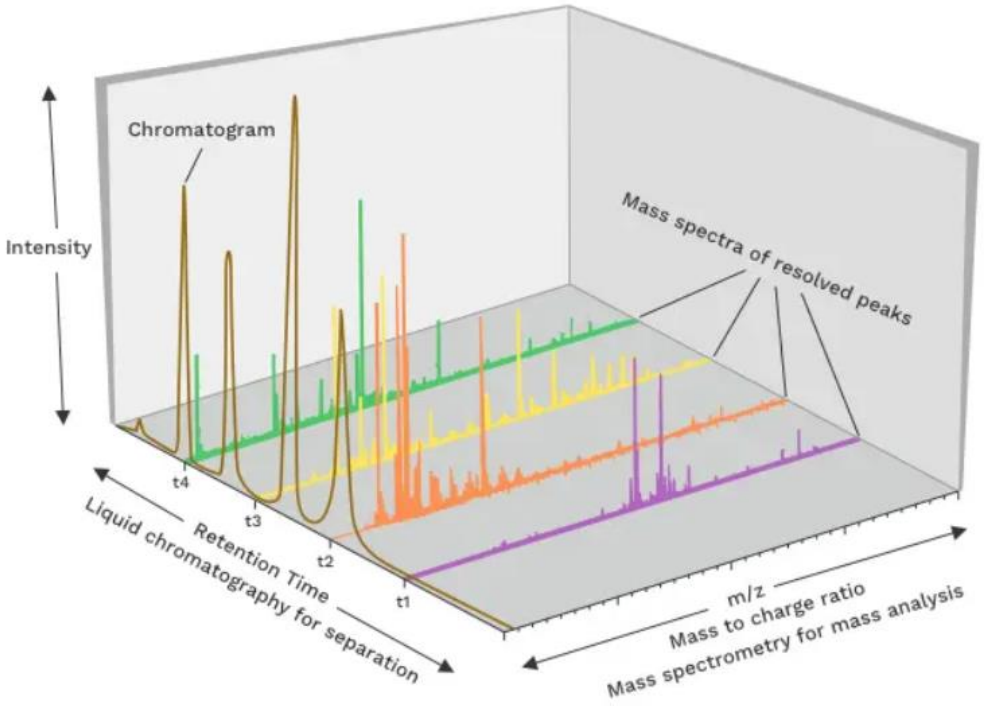

Figure 1.7 LC-MS spectrum [22].

1.5 Metabolite identification

As we previously discussed, the metabolite identification is very important because it provides biological meaning to the dataset. It is the last step of the metabolomic workflow, and Its purpose is to determine an identifier, for example the structure, the name or the database id of the metabolite, for each of the features, since the features by themselves are just analytical data and do not provide any biological meaning. It is a critical step because misidentification could lead to wrong biological interpretations [7] .

There are a lot of different ways to refer to the confidence levels of identification. The most common is the Metabolomic Standards Initiative (MSI) which includes five levels, see Table 1.4. To reach level 3 confidence usually, the mass to charge (m/z) values that are obtained by the MS are taken into consideration in order to attempt to assign one or more potential metabolites that could correspond to that specific m/z value. For confidence level 2 we need to incorporate new information from other sources, for example RT or Collision Cross Section (CCS) ion mobility from adapting a chromatographic method or Migration Time (MT) from adapting Capillary Electrophoresis (CE). For reaching level 1 and 0 an analysis of an authentic standard is required in order to compare their properties under identical analytical conditions. There are also other proposed confidence levels, for example the one from Schrimpe-Rutledge or Sumner but we will not analyse them in this book [7].

Table 1.4 Confidence levels proposed by the MSI **[7]**

Confidence level	*Description*	*Matching requirement*
Level 0	Unequivocal 3D structure, including full stereochemistry.	Determination of 3D structure following natural product guidelines.
Level 1	Confident 2D structure, using reference standard or full 2D structure elucidation.	At least two orthogonal characteristics, such as MS/MS fragmentation pattern, RT or CCS.
Level 2	Probable structure using literature data and/or fragmentation spectra and/or knowledge over the RT.	At least two orthogonal characteristics matching and evidences for excluding the rest of candidates.
Level 3	Possible structure, isomers or class.	More than one candidate, only one characteristic matched is required for supporting the proposed candidate.
Level 4	Unknown.	Detectable feature in a sample.

1.6 Biological and Chemical Databases

An important reason of the success in bioinformatics is the creation of databases. They are the main source of data and knowledge for the research community. In the past, the large volume of data that started to rise led to the development of comprehensive databases and nowadays it is necessary to update them daily. The maintenance of a database requires many specialized scientists who will deal exclusively with the marking of possible errors as well as with the annotation of the new data. They are crucial for storing, organizing, and analyzing big data, enabling researchers to make significant scientific advancements [5].

1.6.1 Categories of Databases

1) Primary Databases

Primary databases contain the primary, raw experimental data that are submitted by researchers. In metabolomics primary databases are used for querying the data and retrieving detailed experimental records, adjusted according to the needs of each user, so that the information it receives is more useful and functional. Each entry typically has a unique

identifier, known as a primary key, ensuring distinct and precise retrieval of records. The records include fields such as molecular structures, experimental conditions, and spectra. Primary databases can include nucleotide sequence, protein amino acid sequence, 3D biological structures, gene expression, genetic variation and literature data. Some examples of the above are [5]:

- GenBank: Stores DNA sequences [23].
- Protein Data Bank (PDB): Contains 3D structural data of proteins [24].
- METLIN: Provides mass spectrometry data for small molecules [25].

2) Secondary Databases

Secondary databases process data from primary databases, adding annotations, interpretations, and integrating information from multiple sources. There are two categories the relational and specialized databases. So, this category of databases is crucial for analytical purposes, generating reports, and synthesizing information [5]. Examples include:

- UniProt: Annotates protein sequences with functional information [26].
- KEGG: Offers pathway maps for understanding biological functions [27].
- Human Metabolome Database (HMDB): Provides detailed information on metabolites, their properties, and roles [28].

1.6.2 Databases in Metabolomics

As we mentioned previously, the main objective of metabolomics is the characterization of metabolites, and this relies heavily on databases that provide spectral information and other detailed data on metabolites. Key databases in this field include:

- ChemSpider: A free chemical structure database providing access to structures. It includes data on chemical and physical properties, spectra, literature references, and links to various resources [29].
- Human Metabolome Database (HMDB): A free database that offers detailed information on human metabolites, including their chemical structure, pathways etc. It hyperlinks data fields to other databases [28].
- MassBank: An open-source database for mass spectrometry data for identification of of small molecules through spectral matching [30].
- LipidMaps: Is a relational database specializes in lipidomics. It provides data on lipid structures and annotations [31].
- METLIN: Contains high-resolution mass spectrometry data for small molecules, aiding in the identification and quantification of metabolites. It includes extensive data on molecular structures, MS/MS spectra, detailed representations of molecular structures, information on metabolic pathways, interactions, and biological activities [25]. Furthermore, it provides a high data coverage with more than 500.000 molecular standards. All the above are essential for metabolomic analysis and research [25]. Because of that, METLIN could possibly dominate in many aspects of metabolomic research applications, such as in metabolite identification, facilitating the identification

of metabolites in complex biological mixtures, drug discovery, by linking small molecules to biological activities, aiding in the development of new therapeutics, biomarker discovery, by assisting in the identification of potential disease biomarkers through spectral and chemical analysis etc.

In general, after obtaining the experimental information on the compounds (mainly for the case that concerns us m/z and RT) searches are carried out in these databases for compounds with properties compatible with those that we have measured experimentally. These compounds are putative annotations ("plausible identifications"). But many times, there is no experimental information in the database for RT measured in a particular chromatographic method. The m/z is always available because it is derived directly from the formula of the chemical compound (so if you know what the chemical compound is you know its m/z) but the RT is not. Hence the interest in predicting RT computationally.

RT prediction in chromatography plays a crucial role in analytical chemistry, affecting a wide range of fields such as drug discovery, environmental monitoring, and food safety. Accurate prediction of RT can enhance the identification and quantification of compounds in complex unknown mixtures, which is essential for many applications [32] [33] [34]. One of the biggest RT databases available is the METLIN Small Molecule Retention Time (SMRT) database, which consists of more than 80,000 compounds measured using a specific chromatographic method [35]. Due to its size, it can be a solid foundation for developing predictive models. The methodology of how we can represent a three-dimensional molecule using a feature vector for creating predictive models will be discussed in Section 1.7.

1.7 Computational representation of metabolites

1.7.1 Descriptors

Molecular descriptors are numerical features that arise from the application of mathematical functions on chemical graphs. They are extracted from the chemical structure of a molecule and provide information about that molecule. They can be zero-dimensional (0D), one-dimensional (1D), two-dimensional (2D), three-dimensional (3D) or four-dimensional (4D). 0D it gives information about the atom types and their occurrences within the molecule. It does not give information about atom connectivity. 0D descriptors are the molecular weight and atom type counts, bond types etc. 1D descriptors are 1D quantities that provide aggregated information about the molecule according to its chemical formula. It gives information about atomic composition, structural fragments in the molecule. They do not represent the whole topology of the chemical structure, only a part of it. Although they are easy to calculate, they suffer from some degeneracy problems, in which different molecules take the same values for the same descriptor. For this reason, 1D descriptors are usually used in conjunction with higher-dimensional descriptors. Some 1D descriptors are the molecular fingerprints and structure keys (we will discuss about these in the next section). 2D molecular descriptors, else called topological indices, are based on the topology. They are the most popular ones, and they include the information about atomic composition and connectivity of atoms in the molecule. 3D descriptors extract information from the representation of the 3D coordinates of the molecule,

therefore they are based on its geometry. So, 3D descriptors add spatial information about the positions of the atoms. There are two main classes of 3D descriptors, the (Weighted Holistic Invariant Molecular (WHIM) and Geometry, Topology, and Atom Weights Assembly (GETAWAY). WHIM descriptors include information about the size, symmetry, shape and atomic property distribution. GETAWAY descriptors capture knowledge about how each atom affects the overall shape of the molecule and assess how atoms interact in relation to their geometrical positions in 3D. Finally, 4D molecular descriptors that take consider molecular properties for example from electron distribution or from the interaction of the molecule with probes [36] [37].

1.7.2 Molecular fingerprints

Molecular fingerprints of a chemical structure attempt to identify a molecule according to some specific characteristic it has. These characteristic properties of molecules can be described by structure or in other words by the structural keys, which are defined as a set of structural features, atom pairs, functional groups, ring systems etc. and indicate whether a particular subgroup or molecular fragment is present within the molecule. The subgroups in the chemical structure of molecules can be coded into binary structural keys. Thus, the structural subgroups are represented as sequences of '0' and '1' (bit strings), where '0' symbolizes the absence of the specific subgroup from the structure of the molecule and correspondingly '1' symbolizes its presence. This characteristic sequence of '0s' and '1s' of a chemical structure is called a molecular fingerprint and can typically be 150−2500 bits long [38] [37]. The most famous and widely used structural key sets are the Molecular ACCess System (MACCS) keys sets, which include the MACCS960 and MACCS166, the number indicates the number of structural features and the second is the most popular because of the small number of bits (166) and finally the PubChem structural keys which include the search of presence of 881 structural features [36]. See an example of the substructure key-based fingerprints in Figure 1.8

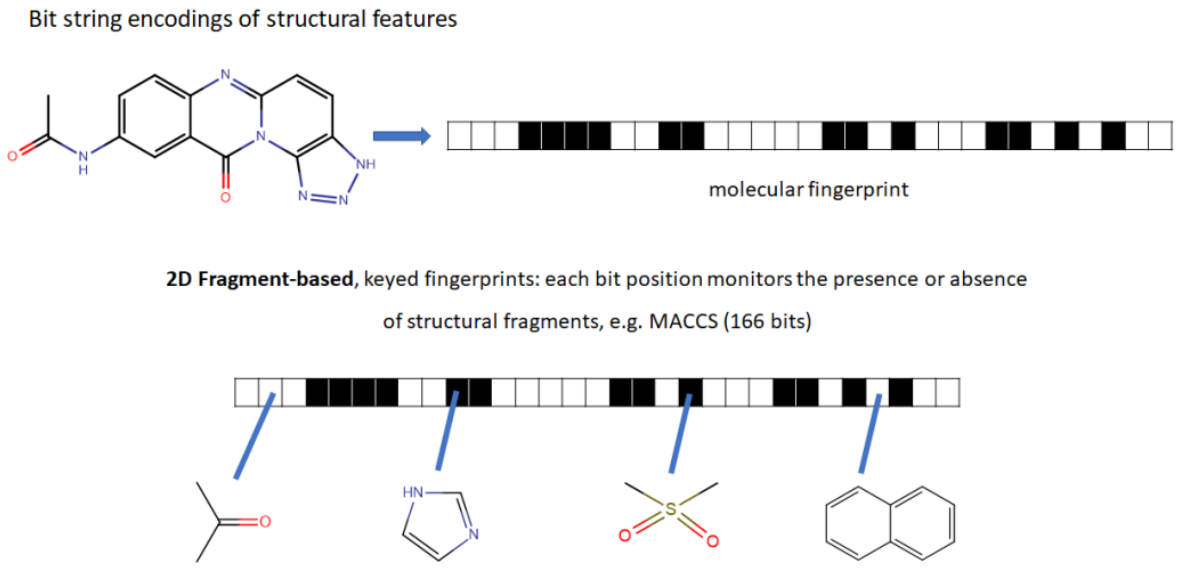

Figure 1.8 Representation of a hypothetical 31-bit substructure key-based fingerprints [39]

Apart from the fingerprints that are developed based on structural keys, there is also another category, called hashed chemical fingerprints and, in this case, the structural features are not predefined, thus they need to be explored storing all the possible identified substructures. This is done following a set of rules and depending on those rules different types of fingerprints derive. The typical types of patterns that can be analyzed are the paths and atom centered (circular patterns), so from them we get two classes of fingerprints the path-based and the circular fingerprint [36].

- Path-based fingerprints create molecular features by examining the pathways within a molecule's graph, focusing on pairs of atoms and encoding these paths into a fixed-size vector through hashing, see Figure 1.9. For instance, Depth First Search (DFS) characterizes a molecule by identifying and storing all unique paths in its graph, starting from each atom and extending outward up to a specified number of bonds. Another method in this category is the Atom Pair (AP) fingerprint, which describes a molecule by gathering all possible triplets formed by two atoms and the shortest path that connects them [40].

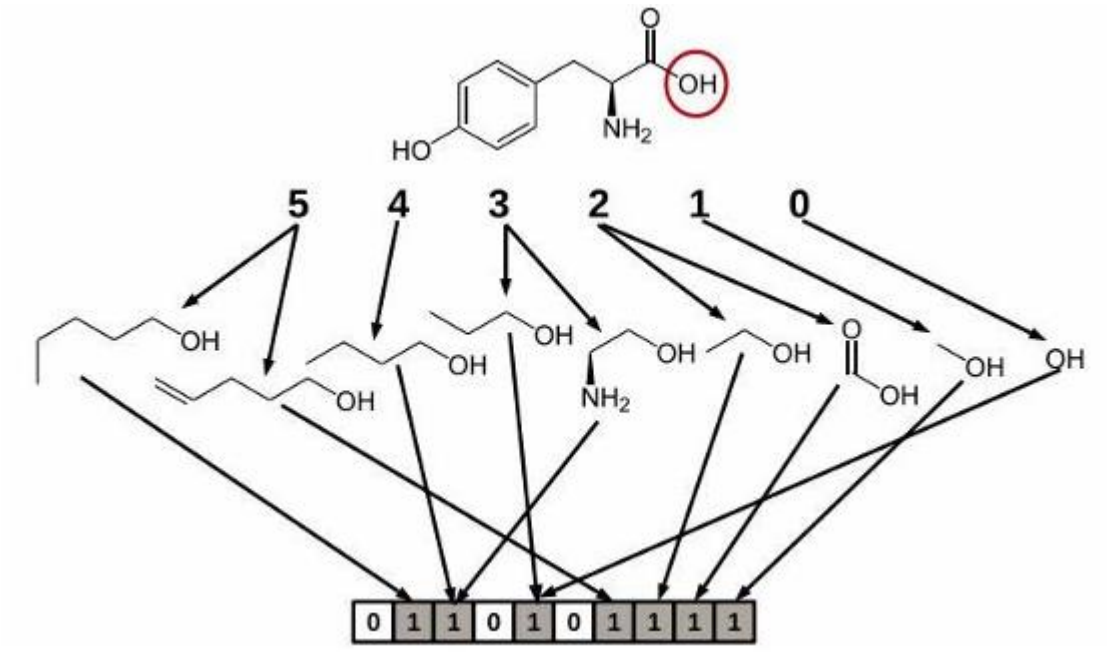

Figure 1.9 Representation of a hypothetical 10-bit path-based fingerprint [41]

- Circular fingerprints, similar to substructure-based fingerprints, decompose a target compound into various fragments. However, unlike substructure-based methods that rely on predefined structural patterns, circular fingerprints dynamically generate these fragments from the molecular graph for each specific compound. The process begins by assigning each atom a representation based on certain properties. These properties can be for example the atomic mass or valence. Neighboring atoms' identifiers are then combined to create a fragment identifier, and this step is repeated with an expanding radius to include

more surrounding atoms, see Figure 1.10. Ultimately, all unique fragments identified are hashed into a fixed-size vector. The primary distinction among these types of fingerprints is in the atom properties they use. For instance, Extended Connectivity Fingerprints (ECFP) focus on features like atomic number and atomic charge, while Functional Class Fingerprints (FCFP) consider whether an atom is a base, an acid, or a hydrogen bond donor/acceptor, among other characteristics [40].

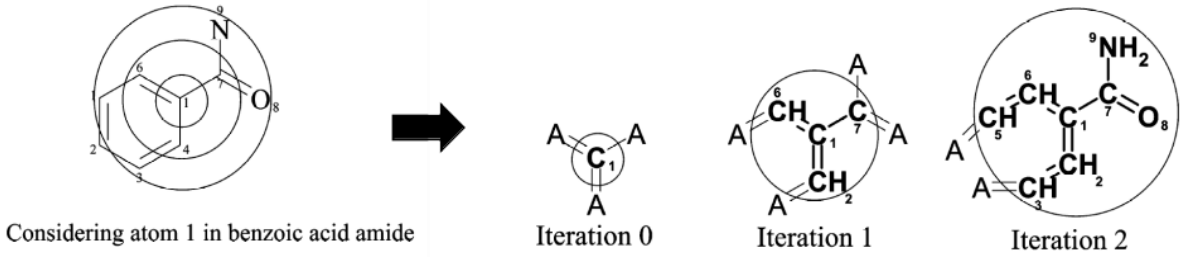

Figure 1.10 Illustration of how iterative updating influences the information encoded by an atom identifier. This example focuses on atom 1 in benzoic acid amide [42].

In general, there are different types of molecular fingerprints that arise according to the form of molecular representation that will be expressed in binary sequence. The most commonly used type is the one that results from the 2D molecular graphs and therefore they are also called 2D fingerprints. However, in some cases, for example in the pharmacophore fingerprints, there is also the ability to store 3D information [43]. 2D molecular fingerprints can be divided into four major categories: substructure key-based fingerprints, topological or path-based fingerprint, circular (such as ECFP) and finally pharmacophore [44].

1.7.3 AlvaDesc

Chemical structures are converted into numerical values, molecular descriptors and fingerprints by many tools, such as Mordred, Padel-Descriptor, CDK, RDKit, Dragon, and AlvaDesc. The later, is a recent tool that offers 5471 descriptors and three types of fingerprints: MACCS166, circular, and path fingerprints. It can handle various chemical structures full-connected and non-full-connected, such as salts, mixtures, and metal complexes. It supports macOS, Linux, and Windows [37].

1.8 Machine learning (ML)

Machine learning, a branch of Artificial intelligence (AI), is a computer science field that can solve hard problems that a human being would not be able to solve, learn from experience and cope with new tasks. With ML algorithms we can get insights into structures, patterns within a dataset and we can predict an outcome or behaviour. To do that we need to train a ML model through acquired experience by examples or based on analogies and similarities [45] [46]. The algorithm needs to be feed with data. Every dataset consists of features which are the important

information that we need. The quality of the features that will feed the computing system plays a crucial role in the accuracy of our model and therefore, before using the dataset it is very important to preprocess it because the data may contain errors, inconsistencies, missing or irrelevant data [47]. In general, all ML algorithms follow these specific steps, train, test and validate [48]. The learning models can be divided in the four following [45] [48], you can see some of the models in Figure 1.11.

- Supervised Learning: In supervised learning the given dataset's inputs of our training set (feature vectors) include specific known labels corresponding to their outputs-outcomes. In this case the algorithm knows the patterns before processing the data. The final goal is to generalize the patterns and given new inputs to give the correct label of their output. Supervised learning is mainly used for classification and regression problems. Some of the supervised techniques include: artificial neural networks (ANN), linear regressors, bayes, k-nearest neighbours (KNN), decision trees, Support Vector Machine (SVM) etc. [45] [48] [49] [50].
- Unsupervised Learning: In unsupervised learning the given training set does not include labels. So, the ML algorithm will have to identify similarities between the inputs, group them and create clusters. Finally, when new set of data feeds our model, the algorithm will have to search between those existing clusters and choose in which one the new data points will belong to. The algorithm uses clustering techniques such as are k-means, fuzzy logic algorithms [45] [48] [49].
- Semi-supervised Learning: As the name indicates it is a learning model that combines the idea of the two previous types of learnings. In this case, only some of the instances are labelled. Uses clustering techniques to create the clusters and use the labelled data to provide labels to the non-labelled instances from the same group [45] [48].
- Reinforcement Learning: It works dynamically by using feedback mechanisms to automatically evaluate the performance of the model in a specific environment, with the ultimate aim of maximizing a specific predefined goal and improving its effectiveness. This way of learning utilizes the reward and penalty method to maintain the knowledge that the system receives from its environment, increasing the reward each time [45] [51].

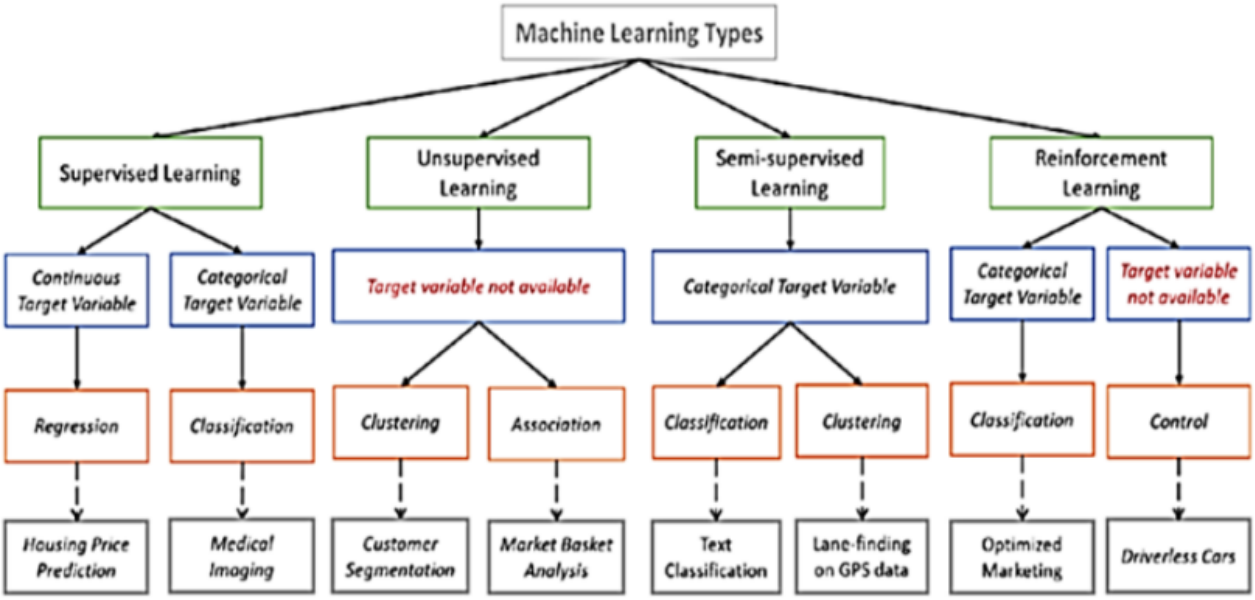

Figure 1.11 Types of ML algorithms [52].

1.9 Neural Networks and Deep Learning

Neural networks and deep learning represent a significant leap forward in the field of artificial intelligence (AI). An ANN is a huge parallel processor with distributed architecture, consisting of interconnected layers of simple processing units, nodes or neurons and has by its nature the ability to store empirical knowledge and make it available for use. The ANN resembles the human brain because knowledge is acquired from the network by the learning process (trial and error) as well as because the connections between the neurons which are known as synaptic weights are used to store knowledge and they express the importance of each connection of the ANN and how they contribute to the final result. The analogy between biological and technical neural networks is as follows: cell body-neuron, dendrites-input, axon-output, synaptic terminals-weights respectively, see Figure 1.12 [53] [54].

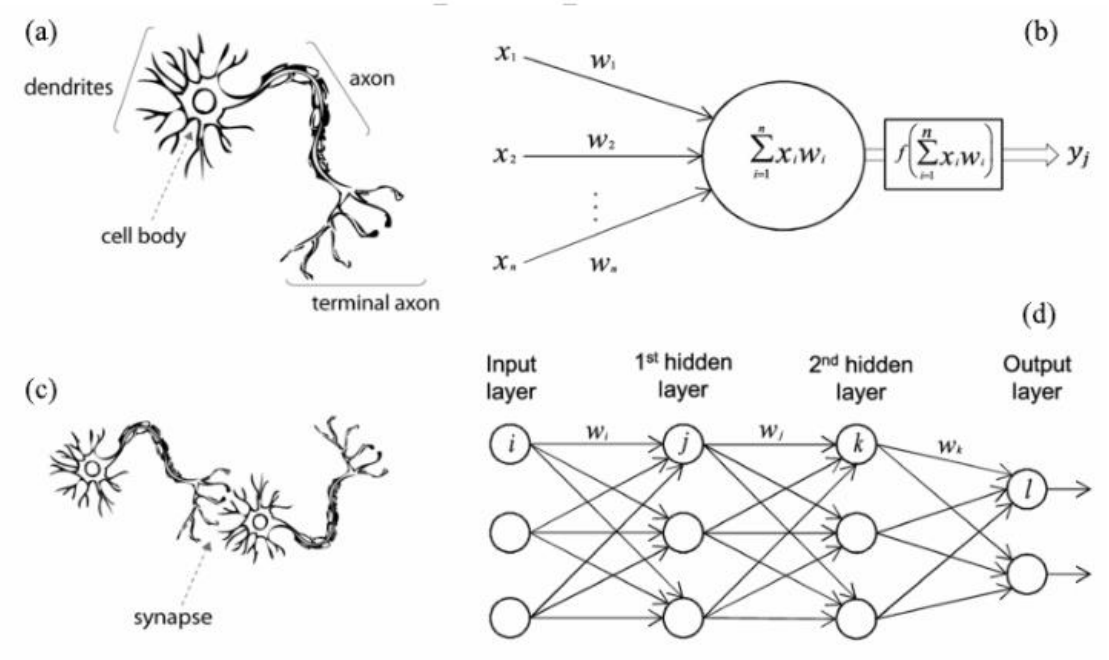

Figure 1.12 A biological neuron in comparison to an ANN. (a) a human neuron, (b)an artificial neuron, (c) a biological synapse, (d) an ANN synapses [55].

1.9.1 Basic Concepts and Terminology

1) Neurons and Layers

The Perceptron

is among the most basic and simple feedforward artificial neural network architectures. It processes one training example at a time, making predictions for each instance. Whenever an output neuron makes an incorrect prediction, the Perceptron adjusts the connection weights, strengthening those inputs that would have led to the correct prediction [53] [54]. There are two types of perceptron.

- Single-layer Perceptron (SLP): As the name indicates it only consists of a single-layer of input nodes and an output layer without including any hidden layers. Because of the architecture it is only able to learn linearly separable patterns. The forward propagation in a SLP for a single output node is [54]:

$$y = \boldsymbol{f}(u) = f(\boldsymbol{w} \times \boldsymbol{x} + \boldsymbol{b}) \qquad \textbf{1.1}$$

where:

- **x** is the input vector
- **w** is the weight vector
- b is the bias of the output neuron of the output layer
- y is the output of the SLP

- Multilayer Perceptron (MLP): When the ANN has hidden layers between the input and the output layers it is called a multi-layer ANN or multi-layer perceptron (MLP). This architecture can solve more than linearly separable problems. The forward propagation in an MLP for a single output node is [54]:

$$y = \boldsymbol{f}(u) = f(\sum_{i=1}^{n} w_i \times y_i + b) \quad \textbf{1.2}$$

Where:

- w_i is the weight of the output neuron which connects it to the i neuron from the previous hidden layer
- y_i is the output of the previous hidden layer
- y is the output of the MPL
- i is the index of the node in the hidden layer, i=1, …r
- b is the bias of the output node

2) Activation Functions

Activation functions determine if the neuron will be activated based on the inputs. Common activation functions include [54] [56]:

- Step**:** Maps input values to a range between 0 and 1

$$y_{step} = f(u) = step(u) = \begin{cases} +1 \; if \; u \geq 0 \\ 0 \; if \; u < 0 \end{cases} \quad \textbf{1.3}$$

- Sigmoid: Maps input values to a range between 0 and 1.

$$y_{sigmoid} = f(u) = sigmoid(u) = \frac{\mathbf{1}}{\mathbf{1}+\boldsymbol{e}^{-u}} \quad \textbf{1.4}$$

- Tanh: Maps input values to a range between -1 and 1.

$$y_{tanh} = f(u) = tanh(u) = \frac{\mathbf{1}-\boldsymbol{e}^{-u}}{\mathbf{1}+\boldsymbol{e}^{-u}} \quad \textbf{1.5}$$

- ReLU (Rectified Linear Unit): Outputs the input directly if positive, otherwise zero.

$$y_{relu} = f(u) = relu(u) = \begin{cases} 0 \; if \; u \leq 0 \\ u \; if \; u > 0 \end{cases} \quad \textbf{1.6}$$

3) Learning Process

Different types of learning

- Supervised Learning
- Reinforcement Learning
- Unsupervised Learning (Self-Organizing)

Typical ANN learning methods, see Figure 1.13 for more examples [54].

- Error Correction Learning
- Hebbian Learning
- Competitive Learning
- Boltzmann Learning

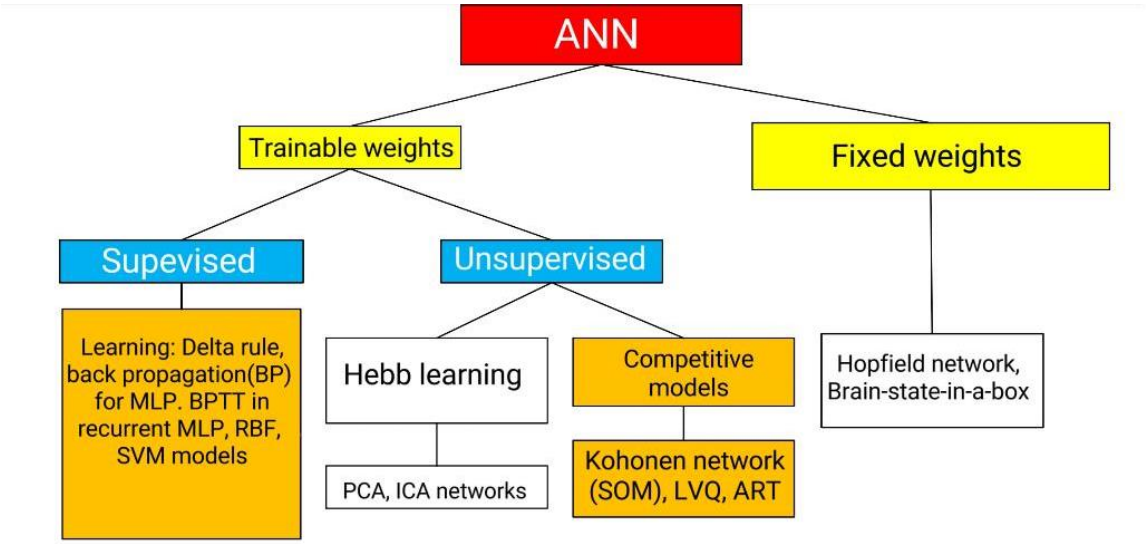

Figure 1.13 Typical ANN learning methods [54]

4) Weights and Biases

Neurons are parameterized as we previously explained by the weights and biases. The weights are applied to the interconnections, to represent synapse strengths, in other words the strength of the connections between neurons. Unlike the weights, biases are not applied to interconnections but are added to the neuron's output. They determine the activation threshold of each neuron. The weights and sometimes the biases are adjusted during training to minimize the error in the network's predictions [54].

5) Forward Propagation

Input data is passed through the network layer by layer, producing an output. First, we have the net input (sum of the inner product of the input and weight vectors, plus the biases). Then, we have the activation function which as we discussed is applied to the net input which gives the output of the node [45], see Figure 1.12 (b) which represents a schematic of the information flow in the artificial neuron.

6) Loss Function

The loss function compares the true and the predicted output of the network. It provides some error measurement. In the case of regression problems some typical loss functions are the following [56]:

- Mean squared error (MSE): measures the average squared difference between the predicted values and the true values. The squared term makes this metric sensitive to outliers [57]:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{i\_true} - y_{i\_pred})^2 \quad \textbf{1.7}$$

- Mean absolute error (MAE): measures the average absolute difference between the predicted values and the true values. It does not square the errors, which makes this metric less sensitive to outliers [57]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_{i\_true} - y_{i\_pred}| \quad \textbf{1.8}$$

- Median absolute error (MedAE): measures the median absolute difference between the predicted and the true values. Measuring the median instead of the average makes it less affected by extreme values, meaning that it is robust to outliers [58] :

$$\boldsymbol{MedAE = median(|y_{i\_true} - y_{i\_pred}|)} \quad \textbf{1.9}$$

- Mean absolute percentage error (MAPE): measures the average percentage difference between the predicted and true values. It is a useful metric when you want to understand the results in terms of percentage. A big disadvantage as we will see in the results is that MAPE can be problematic when true values are close to zero [57]:

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_{i\_true} - y_{i\_pred}}{y_{i\_true}}\right| \quad \textbf{1.10}$$

Where:

- i is the i-th data point
- yi_true is the true value of RT
- yi_pred is predicted by the model value of RT

7) Backpropagation

After the forward pass and the loss calculation we have the backward pass. In other words, this algorithm moves backward through each layer to evaluate the error contributed by each connection and then modifies the weights and biases based on this evaluation by calculating

the gradient of the loss function (Gradient Descent step) with respect to each parameter and adjusting them to reduce the loss [54].

- Gradient descent: The objective is to minimize the cost function j. This happens in a specific value for each weight. In gradient descent an important parameter is the learning rate. If the learning rate is too small the algorithm will have to go through many iterations which will take longer to reach the desired point and if the learning rate is too high the algorithm might overshoot the solution and keep going further from the minima [54] [56].

$$\frac{\partial w}{\partial t} = -\frac{\partial J}{\partial w} \quad \textbf{1.11}$$

Where:

- w is the weight
- t is the time
- J is the loss function

Because the computer is not possible to simulate the continuous time we use a distinct time. Therefore, the above relationship changes into:

$$dw = -dt \times \frac{\partial J}{\partial w} \quad \textbf{1.12}$$

$$dt \approx \eta \quad \textbf{1.13}$$

Where:

- η is learning rate

$$dw \approx \Delta w \quad \textbf{1.14}$$

$$\Delta w = wnew - wold \quad \textbf{1.15}$$

$$g = \frac{\partial J}{\partial w} \quad \textbf{1.16}$$

Where:

- g is the slope

From the above, the equation 1.12 changes into

$$wnew = wold - \eta \times \frac{\partial J}{\partial w} = wold - \eta \times g \quad \textbf{1.17}$$

Let's consider a simple MLP with one hidden layer. For l=1 we have the input layer, l=2 the hidden layer and l=L the output layer. The neuron i belongs to the hidden layer and the neuron k to the output layer. See Figure 1.14 for this representation.

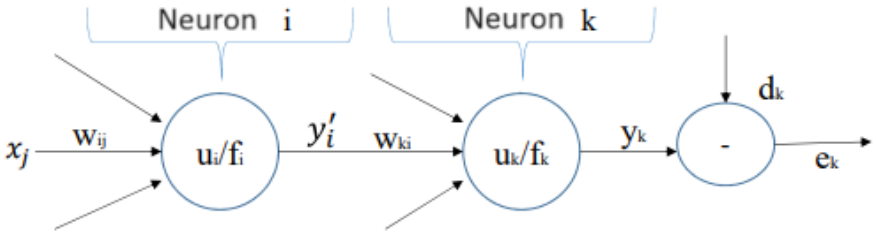

Figure 1.14 Diagram of the signal in an MLP [54].

The weighted error signal of a hidden neuron is determined by backpropagation based on the errors of all neurons from the layer l + 1 to which the hidden neuron that is directly connected to, see on Figure 1.15 the error backward pass. Local gradient, δ(delta) of a i node is calculated as follows:

$$\delta_i(l) = -\frac{\partial J}{\partial u_i} = -\frac{\partial J}{\partial y_i} \times \frac{\partial y_i}{\partial u_i} = -\frac{\partial J}{\partial y_i} f_i'(u_i) \qquad \textbf{1.18}$$

- Cost function J: We want to minimize the cost function, therefore, given the observed data, the cost value is very low when the hypothesis agrees with the observation and high when the hypothesis does not agree with the observation [45].

$$J = \frac{1}{2}\sum_k e_k^2 \rightarrow \frac{\partial J}{\partial y_i} = \sum_k e_k \times \frac{\partial e_k}{\partial y_i} = \sum_k e_k \times \frac{\partial e_k}{\partial u_k} \times \frac{\partial u_k}{\partial y_i} \qquad \textbf{1.19}$$

We calculate the partial derivatives that appear in the sum after applying the chain rule

$$e_k = d_k - y_k = d_k - f(u_k) \rightarrow \frac{\partial e_k}{\partial u_k} = -f_k'(u_k) \qquad \textbf{1.20}$$

$$\mathrm{u_k} = \sum_{\mathrm{i=0}}^{\mathrm{m}} \mathrm{w_{ki} y_i} \rightarrow \frac{\partial \mathrm{u_k}}{\partial \mathrm{y_i}} = \mathrm{w_{ki}} \qquad \textbf{1.21}$$

We replace the partial derivatives of equation 1.19 with the calculations made in equations 1.20 and 1.21

$$\frac{\partial J}{\partial y_i} = -\sum_k e_k \times f_k'(u_k) \times w_{ki} \qquad \textbf{1.22}$$

For the delta rule, of node i, we replace in equation 1.18 its equal from equation 1.22:

$$\delta_i(l) = f_i'(u_i) \times \sum_k e_k \times f_k'(u_k) \times w_{ki} \qquad \textbf{1.23}$$

Where: $e_k \times f_k'(u_k) = \delta_k(L)$

$$\delta_i(l) = f_i'(u_i) \times \sum_k \delta_k(L) \times w_{ki} \quad \textbf{1.24}$$

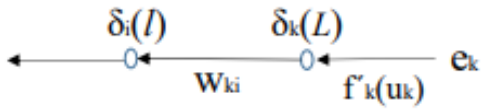

Figure 1.15 Flow diagram of error backward pass [54].

So, in general we have the following [54] :

- Calculation formulas of δ for MLP:

1.If neuron k is an output node:

$$\delta_k(L) = (d_k - y_k) \times f'(u_k(L)) = e_k \times f'(u_k(L)) \quad \textbf{1.25}$$

2. If neuron j is a node of the hidden layer l:

$$\delta_j(l) = f'(u_j(l)) \times \sum_{k=1}^{r} \delta_k(l+1) \times w_{kj}(l+1) \quad \textbf{1.26}$$

- Formulas for the update of weights for MLP:

1. For weights between the output and the last hidden layer:

$$\Delta w_{kj}(L) = \eta \times \delta_k(L) \times f(u_j(L-1)) \quad \textbf{1.27}$$

Where:

$$f(u_j(L-1)) = y_j(L-1)) \quad \textbf{1.28}$$

2. For weights between the input and the hidden layers:

$$\Delta w_{ji}(l) = \eta \times \delta_j(l) \times x_i(l-1) \quad \textbf{1.29}$$

Where:

- l is the number of the layer
- L is the number of output node
- j is the number of the node from the hidden layer
- i is the number of the input node

8) Adaptive Moment Estimation (Adam) optimizer:

Adam is an optimization algorithm that is used to minimize the cost function. It is an extension of stochastic gradient descent. The main difference is that Adam does not maintain a specific learning rate, but it uses momentum that adapts per-parameter learning rates. The parameter adjustments consider the earlier gradients, particularly the squared gradients to scale the learning rate like RMSprop and the average of the gradients like SGD with momentum [56].

1.9.2 Types of Neural Networks

1) Feedforward Neural Networks (FNN)

These are the simplest type of neural networks, where connections between nodes do not form cycles. They are typically used for tasks like classification and regression [56]. A typical FNN is shown in Figure 1.12 (d).

2) Convolutional Neural Networks (CNN)

CNNs are mostly used for image and video recognition. They consist of convolutional layers that apply filters to input data, pooling layers that down sample data, and fully connected layers that output predictions [56].

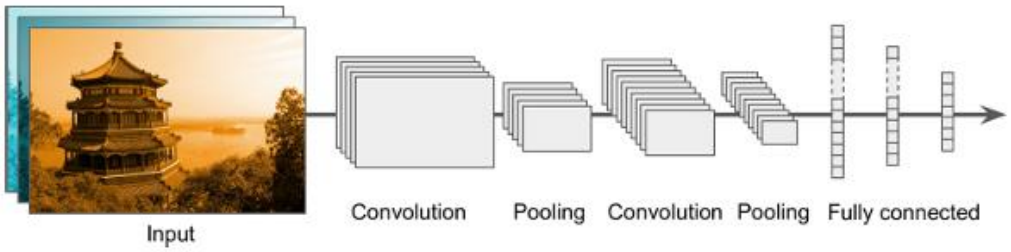

Figure 1.16 Typical CNN architectures [56].

3) Recurrent Neural Networks (RNN)

RNNs are designed for sequential data, for small and very long sequences such as time series and natural language processing. They incorporate connections that create directed cycles, which enable them to retain a memory of past inputs [59].

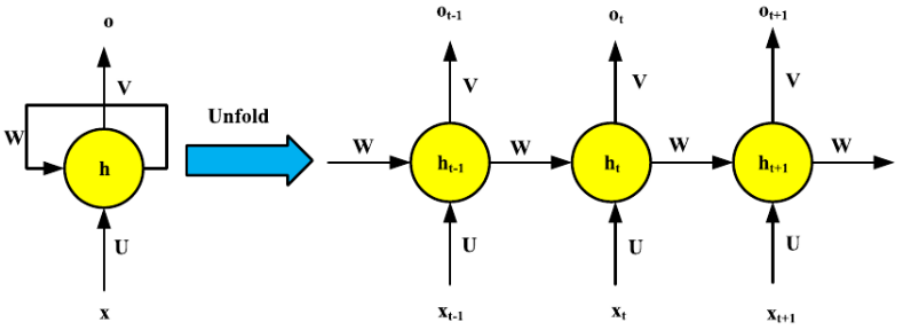

Figure 1.17 Typical architecture of RNN [60].

4) Other Architectures

- LSTM (Long Short-Term Memory): A type of RNN that can detect long-term dependencies in the data. It is the most popular of the long-term memory cells [56] [59].
- GANs (Generative Adversarial Networks): Networks used for generating new data samples. A GAN is composed of two neural networks, the generator and the discriminator. The discriminator's role is to distinguish between real and fake data, most of times the data are images, while the generator's goal is to create data that appear authentic in order to deceive the discriminator [56].
- Transformers: Advanced models for natural language processing that handle dependencies without requiring sequential processing [56] [61].

1.9.3 Training Deep Neural Networks

An ANN with many hidden layers is called a deep neural network (DNN) that is why it is often said that ANNs are at the very core of Deep Learning [62]. They are very powerful and can solve very complex problems that simple, shallow ANN would not be able to solve. They require a very large dataset in order to give promising results and because of the many hidden layers, they require a very high computational power. They are used in a lot of aspects of life, in entertainment through recommendation of songs, videos, movies, in autonomous vehicles, in healthcare for prediction, medical image analysis, drug discovery etc. [45] [56].

1) Data Preparation

As we previously said, the quality of the features that will feed the computing system plays a crucial role in the performance of our model and therefore, before using the dataset we must make the important preparations. Preprocessing steps include attribute selection, normalization, augmentation, and splitting data into training, test and validation sets [48] [56].

- Attribute selection/Feature reduction: The main reasons for the feature reduction are to improve the quality of our data by keeping only the relevant attributes for our problem, to reduce the dimensionality and reduce the computational complexity/cost and lastly to prevent overfitting (overfitting means that the model has a good performance on the training data but not on new unseen data). There are two types of feature selection methods [47]:

a. The filter method: They use statistical techniques for the identification of the relevant attributes. Some examples are, analysis of correlations, counting frequency of attributes, nearest neighbor search etc.

b. Wrapper method: They employ a data mining technique to assess the quality of the attributes, there are two search strategies the forward and backward.

- Normalization: This transformation is very important for two main reasons. First, it allows finding patterns and drawing conclusions without being affected by the units of the measurements. Second, some analytical methods depend on ranges of the variables, thus the variables should be transformed in a way that they all have the same range [47].

a. The simplest form of normalization (this form is affected by outliers):

$$x_{new} = newmin + \frac{newmax - newmin}{oldmax - oldmin}(x - oldmin) \quad \textbf{1.30}$$

Where:

- x_{new} is the new normalized variable
- x is the original variable
- newmin is the minimum value of the range we want to employ (most of the times it is set to 0)
- newmax is the maximum value of the range we want to employ (most of the times it is set to 1)
- oldmin is the minimum value of a variable in the dataset
- oldmax is the maximum value of a variable in the dataset

b. Adjust average and standard deviation(std) (this form is less affected by outliers):

$$x_{new} = \frac{x - x_{avg}}{s} \quad \textbf{1.31}$$

Where:

- x_{new} is the new normalized variable
- x is the original variable
- x_{avg} is the sample average
- s is the sample std

c. Normalization robust to noise:

$$x_{new} = \frac{x - median(x)}{MAD(x)} \quad \textbf{1.32}$$

$$MAD(x) = median(|x - median(x)|) \quad \textbf{1.33}$$

Where:

- x_{new} is the new normalized variable

- x is the original variable
- median(x) is the sample median
- MAD(x) is the sample median absolute deviation

- Data augmentation: is a technique that increases the size of the data. Sometimes it might be needed because either the whole dataset is small resulting in lower performance of the model and overfitting or one of the labels includes more instances making the dataset not representative of all the categories. Because of that we can create new fake data by making small changes to the existing ones, most commonly by adding noise, either to the whole dataset or in some of the subgroups of instances [63].

2) Training Techniques

- Batch Training: Divides the dataset into small batches for gradient descent. Hardware accelerators process large batch sizes more efficiently but in practice it can lead to overfitting. Smaller batch size results to better model generalization [56].
- Regularization: Techniques for preventing overfitting. Some of the popular techniques are the dropout, L1 and L2, max-norm regularization, early stopping. Also, batch normalization behaves like a regularizer [56].

  Dropout is one of the most popular regularization techniques for deep neural networks. During training, in every step, each neuron (except the output neurons) has a probability p, called dropout rate, of being temporarily ignored for this step of the training. The dropout rate most of the times it is set somewhere between 10%-50% [56].

- Optimization Algorithms: Algorithms like SGD, Nesterov Accelerated Gradient AdaGrad, RMSprop Adam, and Nadam optimize the learning process [56].

1.9.4 Challenges and Transfer Learning

One of the main challenges of DNN is that in order for them to give promising results they need to be fed with a lot of data. In practice, in a lot of domains of life we do not have huge databases that we could utilize for deep learning, or if we do have big data, it might not include labels which are needed for supervised learning. This creates the problem of data scarcity [56]. For example, some of the applications that suffer from data scarcity are electromagnetic imaging, monitoring the health of civil structures, medical imaging, weather forecasting, wireless communication, and cybersecurity etc. [64]. Some of the ways that we could overcome this challenge could be by increasing the available data through data augmentation, learning a data generator, merging datasets form different tasks but all of these possible solutions do not always work efficiently and effectively [65].

One of the best techniques that has been suggested to address the challenges of data scarcity is transfer learning. It is a type of machine learning that uses generalizable knowledge from a ML model (neural network) and uses it to make predictions about a different but very similar tasks. A previous pretrained model is used and this time you input your new data, which are fewer. This technique has many advantages, it improves generalization, reduces overfitting and the training is faster since you do not have to learn new tasks from scratch [56] [65] [66] [67]. An

approach to implement transfer learning is by freezing most of the upper layers, in order for the weights to not change, and keep unfrozen a small number of lower layers. This is because the high-level features that are most useful and different for the new task are learnt at the lower layers. In general, the more similar the tasks the more layers you should freeze. After training with the frozen layers in some case it is a good idea to unfreeze them and train one final time, with a reduced learning rate in order for the weights to not change a lot [56].

Generally, transfer learning has been successfully applied in various fields, including image recognition [68], natural language processing [69], and biomedical research [70]. In the context of RT prediction, transfer learning offers a promising solution to the limitations of existing databases by enhancing the representativeness and predictive power of ML models for natural products.

1.10 Project's goal

SMRT contains a large number of compounds for which their RT has been measured. But despite the size of the SMRT database, it is mainly composed of synthetic compounds, resulting in a low natural product-likeness score [71]. This is not desirable because synthetic compounds often have different chemical properties than natural products [72]. This means that a model that is trained with synthetic compounds will probably not generalize well when applied to natural products. In practice, research has confirmed that these models, most of the times, have a lower performance when predicting the RTs of natural compounds compared to the performance on the SMRT dataset and can fail to make accurate RT predictions in real-world applications [56].

On the other hand, RT databases corresponding to natural products are very small, typically containing only a few dozen to a few hundred compounds [73]. The small size of the datasets presents a significant challenge for applying deep learning (DL) techniques, because it generally requires large datasets to have a high performance and be able to generalize. However, DL models are very suitable for RT predictions because they can find complex patterns and interactions in a dataset but as we said before their performance reduces with the scarcity of large, representative datasets of natural products [45].

These two previous reasons create a problem in the prediction of RT of natural compounds. This study proposes using transfer learning as a different approach to address the issue. Transfer learning involves initially training a deep learning model on a large dataset, in our case the dataset is the SMRT, to capture more general knowledge on the basic features that influence the RT. This pretrained model is then fine-tuned using the smaller datasets of natural products, from the RepoRT database. By leveraging the knowledge gained from the extensive SMRT database, the model can adapt to predict RTs in natural products more accurately, despite the limited data available [56].

In summary, this work aims to address the limitations of SMRT, not being representative of real-world biological samples, and other available RT databases, being too small for deep learning techniques, by using transfer learning. The idea is to train a deep learning model on the extensive SMRT to learn to extract features of molecules that influence their retention time,

and then adapt and fine-tune that model to learn to predict RTs from natural compounds and in a different chromatographic method using a few dozen or hundreds of RTs of natural products.

2. Materials and methods

2.1 Packages and software

To implement this study a computer intel core i7 with memory 32 GiB was used. Later a swap space 22GB was also added to the computer because the ML algorithm we built was crashing. This happens when the running memory capacity is exceeded. In our case while training DNNs we needed bigger RAM capacity than 32GiB in order to store the model parameters, intermediate computations and data. The programming part was utilized in PyCharm using Python programming language. Pycharm is available in a free version and in a professional edition, which is the one we used. Its installation and use are particularly easy for the user. The dependencies that we used for this work are Keras, Numpy, Optuna, Pandas, Scikit_Learn and Tensorflow. The final code of this study can be found in the GitHub repository [74].

- Scikit-Learn is a very useful and easy-to-use package that can implement many Machine Learning algorithms efficiently. It is used for predictive data analysis, for classification, regression, clustering, dimensionality reduction, comparing, validating and choosing parameters and models and finally for preprocessing. So, it makes for a great entry point to learning Machine Learning [75].
- TensorFlow is a more complex library for distributed numerical computation. It is mainly used to build and train deep neural networks efficiently. It can create complex topologies by using features like Keras Functional API, it is very flexible and distributes the computations across potentially hundreds of multi-GPU (graphics processing unit) servers [76].
- Keras is an API used in deep learning for building, training, running and evaluating deep neural networks. It can run on top of either TensorFlow, Theano, or Microsoft Cognitive Toolkit. SomeTensorFlow comes with its own implementation of this API, called tf.keras. Keras also uses functions for fetching and loading datasets and the famous Sequential, functional and subclassing APIs for different model creations [77].
- Optuna is an open-source Python library its aim is to efficiently optimize the hyperparameters used in machine learning. It simplifies the process of finding the optimal hyperparameters and fine-tuning the ML models by dynamically constructing search spaces and automating the exploration of different configurations. There are two important concepts in Optuna, the study which is the optimization based on an objective function and the trial. The objective function is defined by the user and encompass training the model, evaluating its performance and returning the specified performance metric that Optuna will minimize or maximize, depending on the problem and the metric. A trial is a single execution of the objective function where a specific set of hyperparameters is evaluated. During a trial, Optuna suggests a set of hyperparameters which are used to train and validate the model. The model's performance metrics are computed and reported back to the study. Optuna uses this information to guide the next trials, in a way of improving the next hyperparameter searches. It also uses a feature called pruning which terminates unpromising trials. By systematically exploring different hyperparameter configurations through multiple trials, Optuna efficiently searches for the optimal set of hyperparameters [78].

2.2 Datasets and preprocessing

1) Datasets

First, we trained a machine learning regressor using the METLIN SMRT dataset which consists of the chemical structures and the RTs of 80.038 experimental small molecules, metabolites, drug like small molecules, some of them being natural and most of them being synthetic products [35]. All the data were acquired only specifically using HPLC-MS. The dataset includes both retained and non-retained in the column molecules. The chemical structures were used to obtain features that describe the properties of the molecules. These features were 2.214 fingerprints (MACCS166, ECFP and path-based fingerprints) and 5.666 molecular descriptors and were generated using alvaDesc [37] [79]. The complete list of the features can be seen in [80].

The second dataset we used is the RepoRT from a GitHub repository [81]. It contains data from real metabolomic experiments and therefore, the compounds are natural products. This repository includes training data for retention time prediction for the identification of metabolites from non-targeted LC-MS based metabolomics. It contains 373 datasets, 8.809 unique compounds, and 88.325 retention time entries measured on 49 different chromatographic columns under unique experimental conditions. This dataset, in contrast with the SMRT, does not include non-retained compounds [82]. Again, using the alvaDesc software we generated 2.214 fingerprints. We did not generate descriptors because the best results were acquired with just using the fingerprints. We used this dataset for validating the Transfer learning model from predicted to experimental RTs and for validating and comparing a new state-of-art DNN regressor for each experiment.

2) Preprocessing

For the preprocessing part of our data, we used a simplified code from a previous study that was made in 2022 [83] [84]. On the first dataset (SMRT), for descriptors, features with 0 variance and highly correlated features (correlation>0.9) were removed. In both datasets, for fingerprints the only feature elimination technique that was used was removing the features with low variance. Treating each feature X as a binary Bernouilli random variable, the variance threshold was selected using $Var[X]=p(1-p)$, where $p=0.9$. Finally, in RTs quantile transformation was applied to make them follow standardized normal distribution the average value is 0 and the standard deviation is 1(which has to be inversed before evaluation).

Since the second dataset does not include non-retained molecules, we removed the instances from SMRT that had a RT lower than 300s. That was perceived by plotting the histogram of the RTs in the SMRT dataset, see Figure 2.1.

Another very import part of the preprocessing of the RepoRT dataset was the transformation of RTs from minutes to seconds, because the SMRT RTs are written in seconds, and we want the inputs of the model to be compatible.

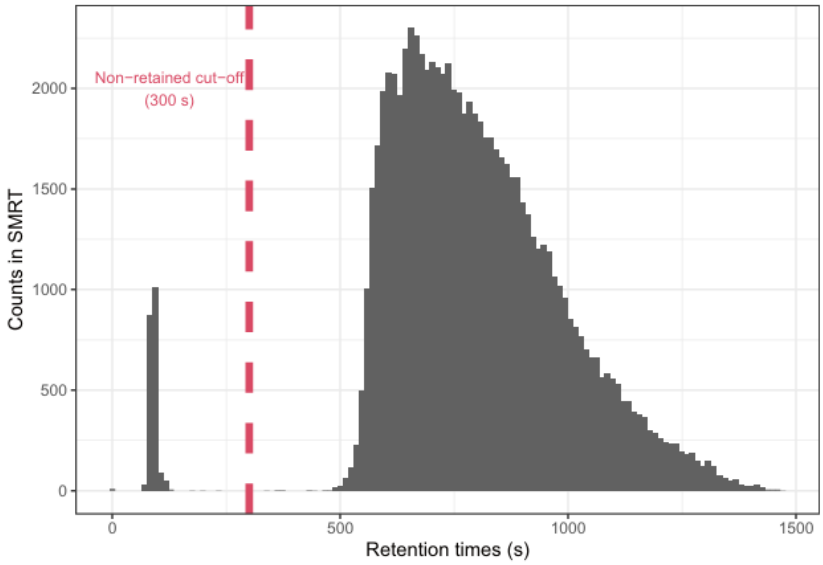

Figure 2.1 Histogram of Retention Times (RTs) for the SMRT dataset. The distribution shows a distinct bimodal shape, which is due to the inclusion of non-retained molecules. In this study, a molecule is classified as non-retained if its Retention Time (RT) is under 300s [83].

2.3 Prediction of retention times with DNNs

As we previously explained the aim of our study is to train a deep learning model on the extensive SMRT dataset, that contains information of mostly synthetic compounds acquired with HPLC-MS, to learn to extract features that influence their retention time. Then, adapt and fine-tune that model to learn to predict RTs from natural compounds and in a different chromatographic method using a few dozen or hundreds of RTs of natural products, from the RepoRT datasets. Finally, we aim to build other DNNs that will be trained directly on the RepoRT datasets in order to compare their performance with the transfer learning approach.

2.3.1 Model creation and hyperparameter optimization

1) DNN trained on SMRT

In the previous study that was based on the SMRT dataset, and from which we took part of the data preprocessing code, the final DNN that was used only had 3 layers [78]. In this case the idea of transfer learning would not be beneficial since all 3 layers include a big part of the knowledge and by unfreezing 1 or 2 layers, we would lose an important part of the required knowledge for the new task.

The new idea was to build a DNN with two different groups of hidden layers. The first one would have more neurons and the second less neurons in order for the first group to have more general knowledge that could be shared for both tasks and the second group would have less parameters that could be fine tunned with less data. The machine learning approach that we used was a DNN that consists of fully connected layers. We did this using the function dense() in which we have to set the number of neurons and the activation function. The input layer has a number of nodes equal to the number of features and the output layer has a single node

because it is a regression problem that returns a single predicted value. We also added dropout, using the function dropout() after each hidden layer. The models were trained using only descriptors, only fingerprints and the combination of both types of features. The code of this implementation is the following:

```
layers = []
    # Input layer
layers.append(Dense(neurons_per_layer1, input_dim=n_features))
    # Intermediate hidden layers
for _ in range(1, number_of_hidden_layers1+1):
    layers.append(Dense(neurons_per_layer1, activation=activation1))
    layers.append(Dropout(dropout_rate1))
    # Intermediate hidden layers
for _ in range(1, number_of_hidden_layers2+1):
    layers.append(Dense(neurons_per_layer2, activation=activation1))
    layers.append(Dropout(dropout_rate2))
    # Output layer
layers.append(Dense(1))
Sequential(layers)
```

As we previously mentioned, even a simple ANN has many hyperparameters that can change the performance of a model. In our study, in all models the chosen optimizer was Adam and for the other hyperparameters instead of simply trying many combinations manually, we used Optuna optimization to select the best ones that provide us with the most promising results for our tasks.

First let's explain how we utilized Optuna. We used the functions trial.suggest_float(), trial.suggest_int() and trial.suggest_categorical() to define the search space. With the first two we define a continuous range for a hyperparameter and Optuna will sample values from this range during each trial. Their difference lies in the type of values they suggest, in one case it is a float and in the other an integer value. The important parameters are the name of the hyperparameter, the lower and the upper bound of the range. With trial.suggest_categorical() we define a categorical search space for a hyperparameter and Optuna will sample one of the specified categories during each trial. The important parameters of the function are the name of the hyperparameter, and the list of the categorical values [78]. For the hyperparameter tuning and the evaluation we used 5-fold nested cross-validation. Nested cross-validation guarantees that different data is used to tune model parameters and to evaluate its performance by means of outer and inner cross-validation loops. The outer loop uses StratifiedKFold to create 5 stratified folds, ensuring that each fold has a representative distribution of the target variable, RTs. During each iteration of the outer loop, the data is split into training and test sets. The splits will be used for averaging the test scores over several data splits. This step ensures that the test data remains unseen during the training and hyperparameter tuning process. Within each fold of the outer loop iteration, there is the inner loop which is executed for hyperparameter optimization. It also creates 5 folds using the RepeatedKFold to split the training data into another train and validation sets. In the inner loop we created a function that defines an objective for Optuna, which includes suggesting hyperparameters, creating a deep neural network (DNN), and performing cross-validation to evaluate the model's performance. The objective function was designed to minimize the Mean Absolute Error (MAE) on the test

splits. We employed 25 trials to explore different sets of hyperparameters. During each trial, the objective function evaluates the model's performance on the test splits and reports the results. We used Optuna's pruning mechanism to stop unpromising trials. After finding the best hyperparameters, the final DNN model is trained on the entire preprocessed training data using these optimal parameters, ensuring robust performance.

For this model, the hyperparameters that were tweaked were the number of hidden layers, the number of neurons per layer, the dropout between the layers, the activation function, the learning rate, the number of epochs and the batch size. The code for implementing the suggestions is the following:

```
def suggest_params(trial):
    params = {
        'lr': trial.suggest_float('lr', 5*10 ** (-6), 10 ** (-4),
        log=True),
        'batch_size': trial.suggest_categorical('batch_size', [8, 16, 32,
        64]),
        'number_of_hidden_layers1': trial.suggest_categorical(
        'number_of_hidden_layers1', [2, 4, 6, 8, 10]),
        'dropout_between_layers1':
        trial.suggest_float('dropout_between_layers1', 0, 0.5),
        'neurons_per_layer1':
        trial.suggest_categorical('neurons_per_layer1', [512, 1024, 2048,
        2500, 4096]),
        'epochs': trial.suggest_int('epochs', 10, 100),
        'activation1': trial.suggest_categorical('activation1', ['relu',
        'leaky_relu', 'gelu', 'swish'])
        'number_of_hidden_layers2': trial.suggest_categorical(
        'number_of_hidden_layers2', [3, 4, 6, 8]),
        'dropout_between_layers2':
        trial.suggest_float('dropout_between_layers2', 0, 0.5),
        'neurons_per_layer2':
        trial.suggest_categorical('neurons_per_layer2', [40, 60, 80])}
    return params
```

The Optuna study showed that the best choice of the number of hidden layers for the first group was either 2 or 10 with 2500 nodes per layer and the second group 3 hidden layers, with 80 nodes per layer. In practice 2 and 10 hidden layers make a big difference in exploring the knowledge and that led us to the decision of instead of using sequential keras API to use the functional API and built a more complex architecture in order to incorporate the advantages of both.

The result was to create a wide and deep neural network for the first group of hidden layers that can learn deep patterns and simple rules. Their outputs concatenate and become the input of the next group of hidden layers [9]. We kept this model and we introduced to it the hyperparameters that were chosen by Optuna for the first study of the sequential DNN.

After creating the models, we had to call the compile() function to specify the loss function we chose the MAE, the optimizer which was ADAM as we previously mentioned and a list of extra metrics that will be computed during the training process, which in our case are the MSE and MAPE. After compiling we have to use the fit() function in order to train the models. In

fit() we have to specify the input and output data, the number of epochs which is the number of complete iterations of the training dataset, the batch size which defines the numbers of training samples that are fed to our DNN each time [56].

```
def fit_dnn(dnn, X, y, optuna_params):
    dnn.compile(
        optimizer=keras.optimizers.Adam(learning_rate=optuna_params["lr"]),
        loss=keras.losses.MeanAbsoluteError(),
        metrics=[
            keras.metrics.MeanSquaredError(),
            keras.metrics.MeanAbsolutePercentageError()
        ],
    )
    dnn.fit(
        x=X,
        y=y,
        batch_size=optuna_params["batch_size"],
        epochs=optuna_params["epochs"],
        verbose=0
    )
    return dnn
```

At the end, we compared the functional topology with the sequential. In the case of the sequential model, we could test the performance setting the first group of hidden layers to either 2 or 10. In our results when we talk about the sequential topology, we will refer to the first group of hidden layers being set to 10. The functional DNN outperformed the sequential and we decided to use this to implement transfer learning on the RepoRT. See the two network topologies in Figure 2.2.

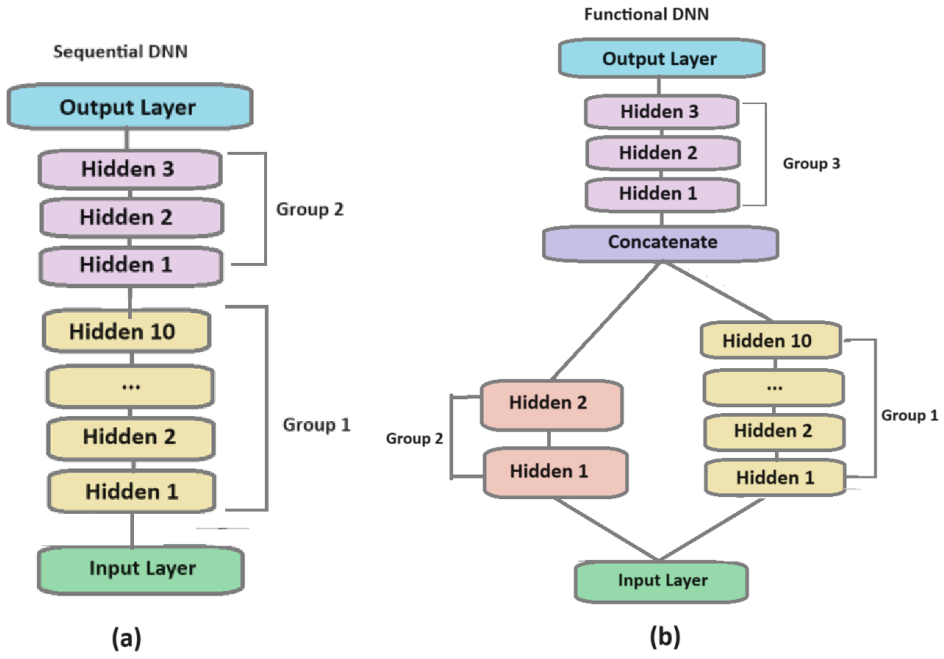

Figure 2.2 Schematic of the two models that we compared. (a) is the sequential model and (b) is the functional and final model that will be used for transfer learning.

2) Transfer learning

We adopted the previous model, we loaded the final configuration and the weights, and we trained it again two times for each of the RepoRT experiments following the transfer learning method described in chapter 1.9.4. First the training was made while we froze the upper hidden layers, in other words the first two groups of hidden layers, allowing only the last group of hidden layers to train. The second training was made after unfreezing them and lowering the learning rate. In this way, the new models learn the basic and generalizable knowledge, which is common in both datasets from the first two groups and the more unique knowledge that is hidden in the last group of hidden layers is acquired with the training process. See Figure 2.3 for better understanding of the training process.

The code for loading the previous model is the following:

```
config_path = f"./results/config.json"
weights_path = f"./results/model.weights.h5"
# Load the JSON configuration
with open(config_path, 'r') as json_file:
   model_config = json_file.read()

# Reconstruct the model architecture from the JSON configuration
model_new = model_from_json(model_config)

# Load the weights from model.weights.h5
model_new.load_weights(weights_path)
```

The code for freezing the hidden layers of the first model is the following:

```
for layer in model_new.layers[:number_of_hidden_layers1+1]:
    layer.trainable = False
```

In the same way, the code for unfreezing the layers is the following:

```
for layer in model_new.layers[:number_of_hidden_layers1+1]:
    layer.trainable = True
```

Then we used Optuna two times. Firstly, to optimize our first part of transfer learning, when only the last group of hidden layers is trainable and secondly after enabling them all to train. In both cases, we utilized an Optuna study for hyperparameter optimization this time without nested cross validation, but with splitting 80% of the data for training and 20% of the data for validation. Since we follow the pretrained model's topology the only tweaked hyperparameters were the learning rate, the number of epochs and the batch size. We did it similarly to the previous code. All the other parameters were fixed to the ones from the pretrained model. The main difference between the two Optuna studies is that the learning rate and epoch suggestions were set to lower numbers on the later. That's because we do not want the weights from the first model to be very tweaked during this training step because this would result in losing knowledge from the previous pretrained model. And of course, we had to compile and fit the model in order to train it.

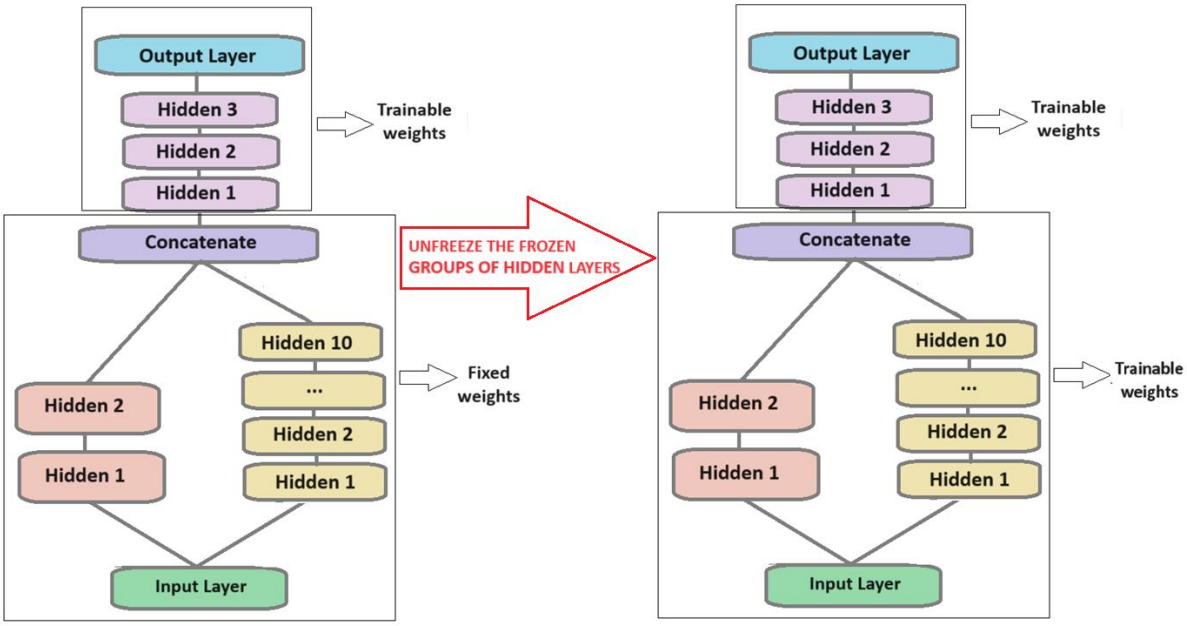

Figure 2.3 Representation of the two training steps for transfer learning

2.3.2 Model evaluation

Finally, we created new state-of-art DNNs regressors for each experiment of the RepoRT which were trained directly on the source data. The main reason of creating these regressors was to show that DNNs do not provide good results when we do not have big input data. So, in the end we want to compare their results with the results of transfer learning approach which is supposed to be solving the problems of data scarcity. The model creation technique is very similar to the first one, although this time we create only one group of hidden layers. Again, we built a DNN that consists of fully connected layers, using the function dense() and we add dropout after each hidden layer. The input layer has a number of nodes equal to the number of features and the output layer has a single node because it is a regression problem that returns a single predicted value. The same hyperparameters were all optimized with Optuna along with 5-fold nested cross validation, the same way as for the first model, assuring that the new models will be built by searching large and complex spaces effectively and without human bias. Finally, we called the compile() and fit() functions to train the DNNs.

For the evaluation of models, the ones trained directly on the RepoRT and the DNNs that we built by leveraging transfer learning to adapt it to the RepoRT, we used Scikit-Learn's K-fold cross validation feature, this is the outer loop of the nested cross validation of the first category of the previously mentioned models. With this technique, we shuffle and split the training set data into k portions, called folds, then we train and evaluate the model K times, picking a different fold for evaluation every time and training on the other K-1 folds. By training and validating the model on different data subsets we can identify whether the model overfits in a specific subset and if it doesn't, we can conclude that the model has a good generalization capability. Also, the fact that every data has an equal chance of being in the training and validation sets reduces the overall bias that could occur if we used a single train-test split, for

example the famous 70%-30% method. So, in general, we split the data, we train and validate, we aggregate the results and finally we find the average performance across all the k folds to obtain the models effectiveness. In our study, we used a 5-fold cross-validation [45] [56]. For evaluation we used most of the loss functions that we previously explained in chapter 1.9.1, MAE, MedAE and MAPE. For the final conclusion we compared the evaluation results of the transfer learning and the DNNs that were trained directly with the RepoRT experiments. For the evaluation, because as we previously explained in chapter 2.2 the RTs had undergone a quantile transformation, we have to turn them into their original state in order to evaluate them. The code for implementing the evaluation is the following:

```
def evaluate_model(dnn, preprocessed_train_split_X, preproc_test_split_y,
preproc_y, fold, features):
    preproc_y_preds = dnn.predict(preprocessed_train_split_X, verbose=0)
    test_split_y =
    preproc_y.inverse_transform(preproc_test_split_y.reshape(-
1,1)).flatten()
    y_preds = preproc_y.inverse_transform(preproc_y_preds.reshape(-1,
1)).flatten()
    # This dictionary creates the evaluation results
    evaluation_dictionary = {
        'mae': mean_absolute_error(test_split_y, y_preds),
        'medae': median_absolute_error(test_split_y, y_preds),
        'mape': mean_absolute_percentage_error(test_split_y, y_preds),
        'fold': fold,
        'features': features
    }
```

3. Results and Discussion

1) First model, trained on SMRT

For the SMRT dataset the best results for each type of feature are shown in Table 3.1. As we can see it is evident that the use of fingerprints alone yielded the best outcomes, as indicated by the lower error metrics. Furthermore, when comparing the best performance achieved with descriptors or a combination of both types of features against the worst performance obtained using only fingerprints, see Table 3.3, we observe that even the least favorable results from the fingerprint model surpass the best results from the other feature sets. Based on this, we decided to focus exclusively on fingerprints for further analysis.

Table 3.1 Best results of each feature

MAE (s)	MedAE (s)	MAPE (%)	FEATURES
44.27	22.32	0.13	Fingerprints
51.35	27.55	0.20	Descriptors
48.49	24.68	0.18	Both types of features

For fingerprints, the hyperparameters selected through Optuna optimization for the sequential topology can be seen in Table 3.2. These are also the hyperparameters we used for the functional DNN.

Table 3.2 Optuna optimization, chosen hyperparameters

Sequential DNN	Optuna: Selected hyperparameters
Learning rate	9*10 ^ (-6)
Batch size	16
Number of epochs	60
Activation Function	Rectified linear unit (ReLU)
Number of hidden layers for group 1	2 or 10
Number of neurons per layer for group 1	2500
Dropout between layers for group 1	0.1
Number of hidden layers for group 2	3
Number of neurons per layer for group 2	80
Dropout between layers for group 2	No dropout

We evaluated the two models by computing the MAE, MedAE, and MAPE for each of the 5 folds. Then we calculated the average error metric that the model presents across all the folds. The results of the sequential model can be seen at Table 3.3 and the results of the functional model can be seen at Table 3.4.

Table 3.3 Evaluation results of the sequential model that was trained on the SMRT dataset

MAE (s)	MedAE (s)	MAPE (%)	FOLD	FEATURES
45.72	23.73	0.14	0	Fingerprints
44.38	21.13	0.26	1	Fingerprints
47.09	24.22	0.13	2	Fingerprints
46.45	23.16	0.44	3	Fingerprints
44.27	22.32	0.13	4	Fingerprints
45.58	**22.91**	**0.22**	**Average value of each error metric**	

Table 3.4 Evaluation of the functional model that was trained on the SMRT dataset

MAE (s)	MedAE (s)	MAPE (&)	FOLD	FEATURES
35.95	19.80	0.04	0	Fingerprints
35.25	19.78	0.04	1	Fingerprints
35.62	19.74	0.04	2	Fingerprints
35.33	19.18	0.04	3	Fingerprints
35.68	19.62	0.04	4	Fingerprints
35.57	**19.62**	**0.04**	**Average value of each error metric**	

By comparing this metrics, we can understand that the functional topology outperformed the sequential and because of that we kept this model in order to use it for transferring knowledge to the new task. In order to get more insights into the performance of the model we did a scatter plot of the actual VS the predicted RTs. We added the regression line of best fit that indicates the general trend of the data, and we calculated the correlation coefficient, see Figure 3.1. The model has great predicting capabilities and the correlation coefficient of the predicted and actual values, which is equal to 0.95, suggest that that the model captures the underlying patterns in the data extremely well and is reliable to make accurate predictions on new, unseen data, assuming that the data distribution remains the same.

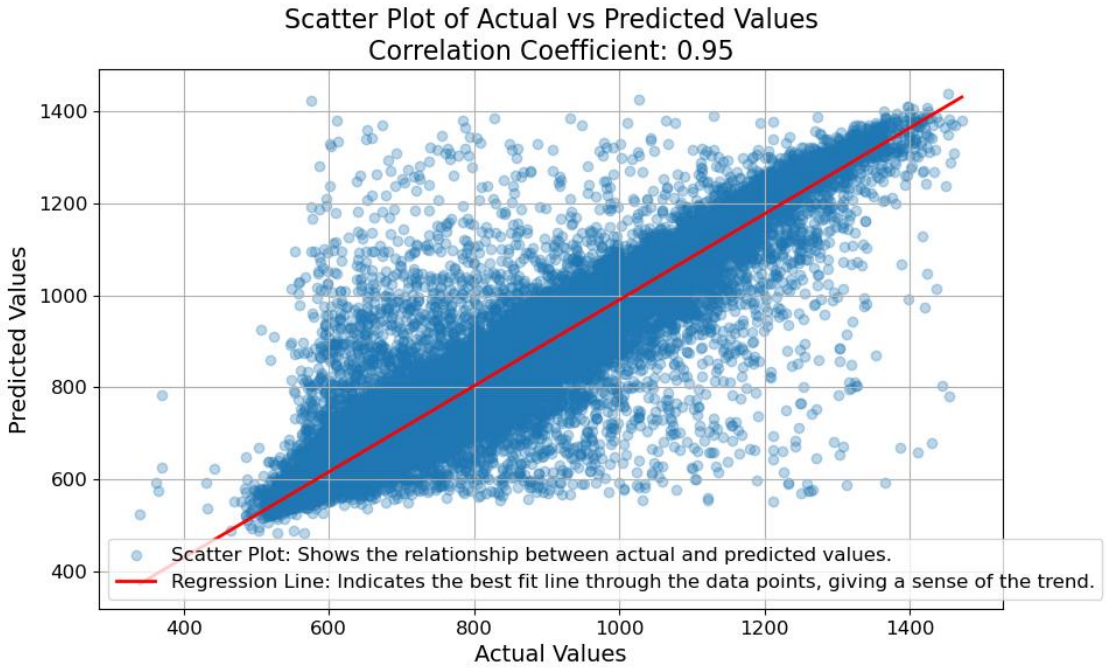

Figure 3.1 Blue data correspond to a scatter plot of the Actual RTs of SMRT vs Predicted RTs by the DNN functional model. The red line is a regression line of the best fit.

2) Transfer learning

As we already described, we trained the model two times. We evaluated the performance of the model both times and we saved the results. Per each RepoRT dataset/experiment we calculated the average error of each metric (MAE, MedAE, MAPE) across all the 5 folds. We then computed the difference between the average values to understand if the model performs better when it is only trained with frozen upper hidden layers or if it performs better after we unfreeze them and train again. We can understand that positive values means that the error of only doing the first step of training is higher than doing both steps. We plotted bar plots of the number of experiments/datasets of RepoRT that had positive value of the difference and the number of the experiments that had negative value, for each of the metrics. As we can see in Figure 3.2 in most datasets the difference is positive for all the error metrics, especially in the case of MAE, indicating that the model performed better, had lower errors, the second time. We also did different bar plots (Figure 3.3) that show the average difference of each error metric per experiment across all folds and we can see that there are a few experiments in which there is a very big negative difference, but they can be translated as outliers. Because of that we continued our study with the results of the second approach.

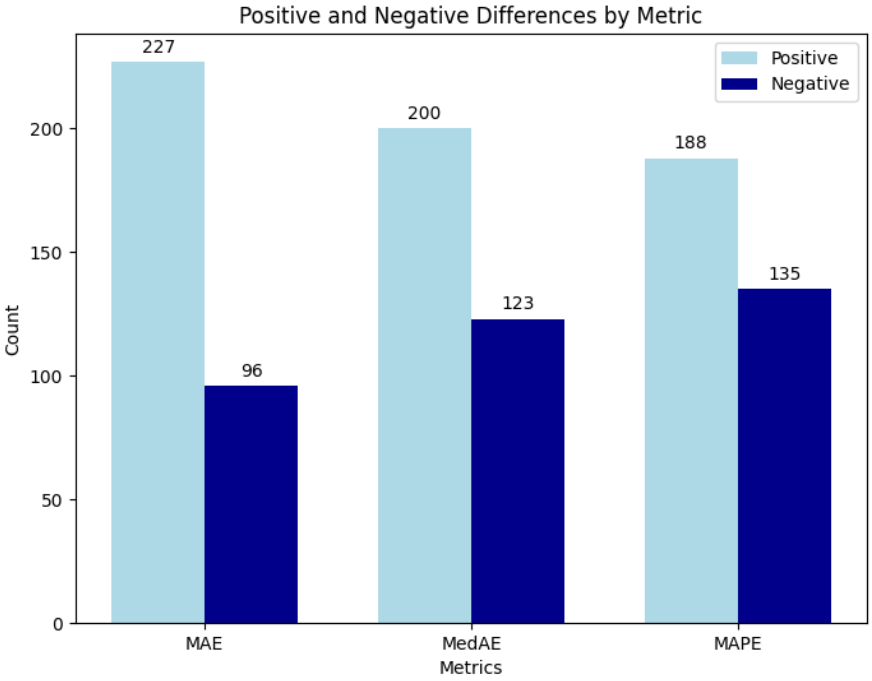

Figure 3.2 Bar plots of the number of experiments that had positive and the number that had negative average difference of each error metric across all the folds

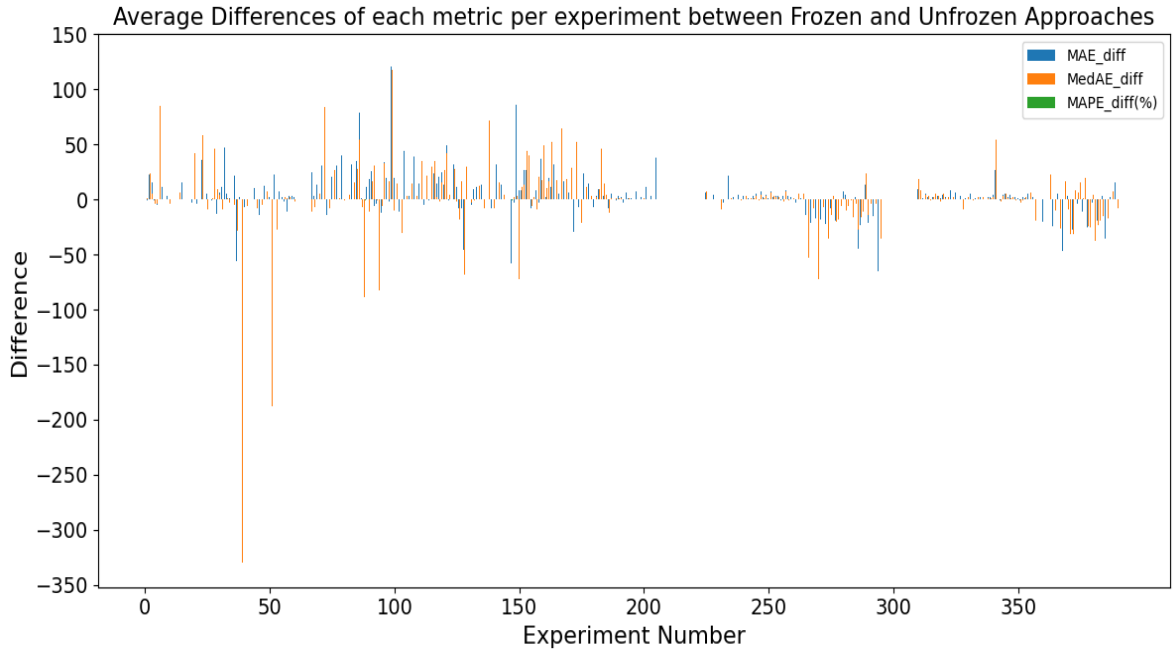

Figure 3.3 Average difference of each error metric per experiment across all folds

In order to understand more our results, for each RepoRT dataset we created three scatter plots of the average error of each metric across all the folds VS the size of each dataset (see Figure 3.4). We wanted to see if there would be a linear relationship between the two parameters, since

as we previously discussed in DNN the models that are trained in more data are mostly more promising.

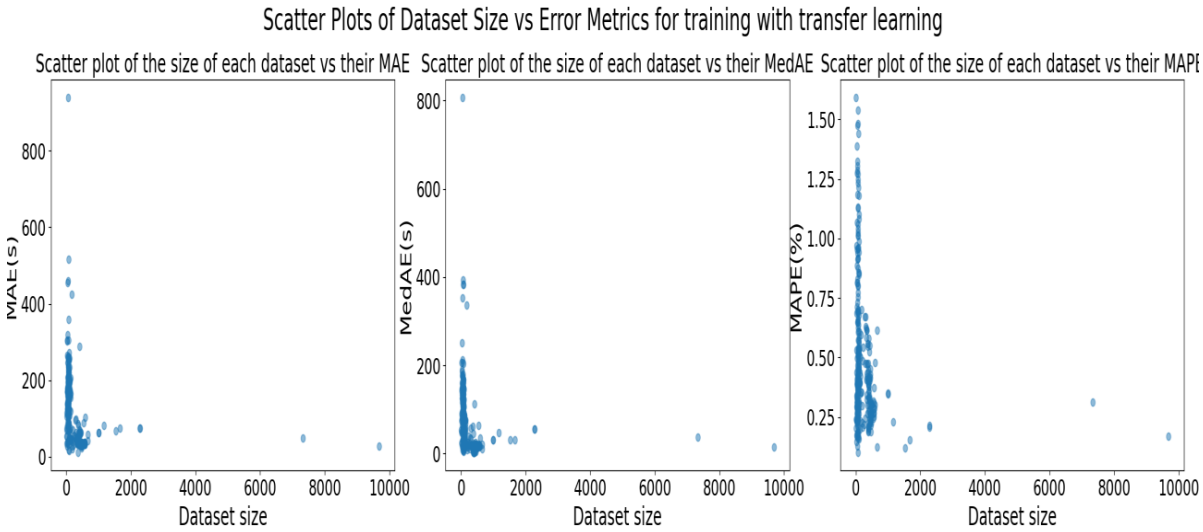

Figure 3.4 Scatter plots of the dataset size of each experiment VS the average error metric across all 5 folds

The observed results were not the expected. One of our thoughts was that this could be due to the fact that the data from SMRT were acquired by untargeted analysis while RepoRT seems to have experiments of both targeted and untargeted analysis which would result in performing better on the same type of analysis, which means that it would perform better in dataset that had longer duration since untargeted analysis requires more time than targeted. Because of that we decided to do a scatter plot of each experiment duration VS their average error metric, see Figure 3.5, and another plot of each experiment duration VS the dataset size of it, see Figure 3.6. From Figure 3.5 we can see in the first plot that the MAE seems to have a linear relationship with the experiment duration, on the second plot we could say that the MedAE also is higher when the duration of the experiment is larger. However, it is important to interpret these comparisons with caution. Since both MAE and MedAE are absolute error metrics that depend directly on time (i.e., RTs of the experiments), it is expected that errors will naturally be larger for experiments with longer durations. This increase is due to the larger numerical values involved, rather than an actual increase in relative error. To obtain a more meaningful comparison, we should focus on the third plot, which presents the percentage error. Unlike the MAE and MedAE, the percentage error is normalized and independent of the absolute duration of the experiments, making it a more reliable metric for assessing model performance across varying experiment lengths. We can see that before 2000 seconds in some cases we have larger errors, even some with over 1.4 MAPE, but after 2000 seconds we don't see any experiments with errors higher than 0.8 MAPE. By this we could interpret that when the duration is longer typically the errors tent to have lower values in general. Based on these results, our assumptions appear to be validated. However, we should consider that the number of experiments that have

duration time over this value is very small, and we are not sure if it is representative. On the other hand, from Figure 3.6 we can interpret that most of the longer experiments correspond to smaller number of metabolites, because for duration bigger than 1500s we do not see datasets with more than 500 metabolites.

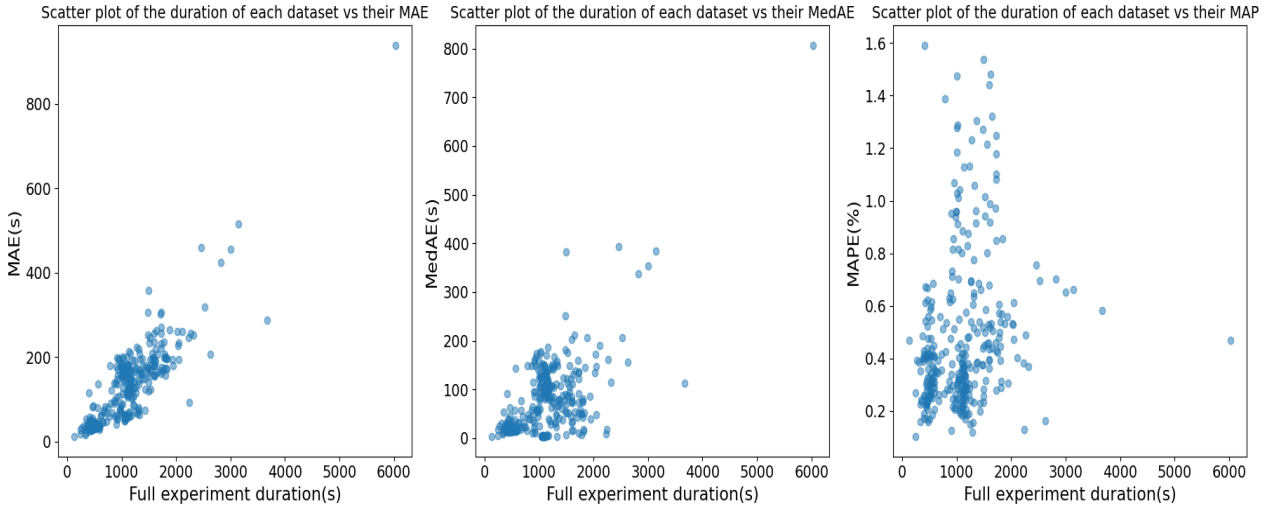

Figure 3.5 Scatter plots of the duration of each experiment VS their average error metric across all folds

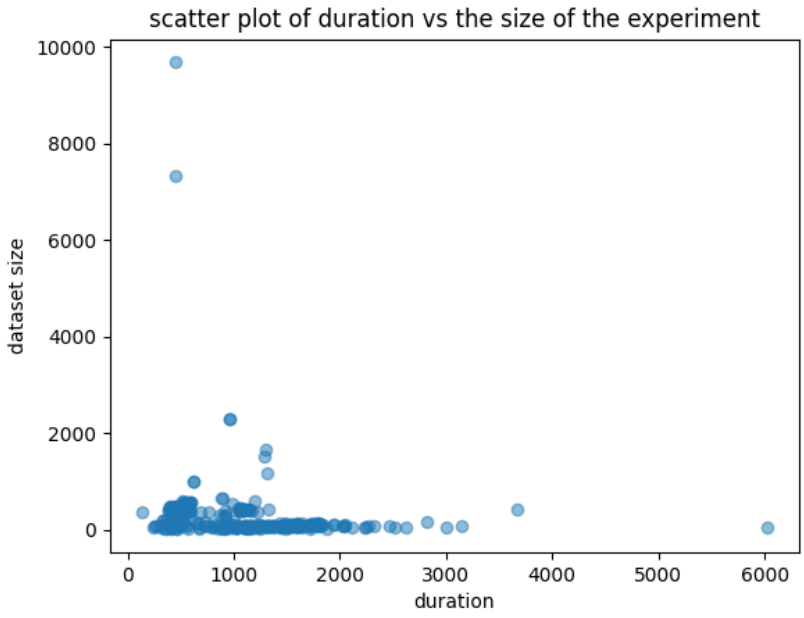

Figure 3.6 Scatter plot of the duration of each experiment VS the dataset size of the experiment

Also, it is important to discuss one incident that occurred while performing a training validation on the RepoRT dataset. The MAPE matric values were not what we first expected to observe, and they were very different than the values of the final model evaluation. The reason behind this is that while doing the training validation, the RTs had undergone a quantile transformation, which as we previously explained transforms them in a way that they follow a standardized normal distribution. So, when using the equation 1.10, if the true RTs of our dataset are close to zero, the denominator is also close to zero resulting to a big number for the fraction and for the loss. On the other hand, on the final model evaluation we use the following function which inverses the transformation and the RTs return to their original form where the values are not distributed close to zero and the fraction does not get a big value.

```
test_split_y = preproc_y.inverse_transform(preproc_test_split_y.reshape(-1,
1)).flatten()
```

3) Comparison between the Transfer learning model and the model that was trained directly on the RepoRT experiments

For the comparisons we used the results we computed the average error metrics across all the experiments and all the folds for both the transfer learning and the direct training approach, see Figure 3.7.

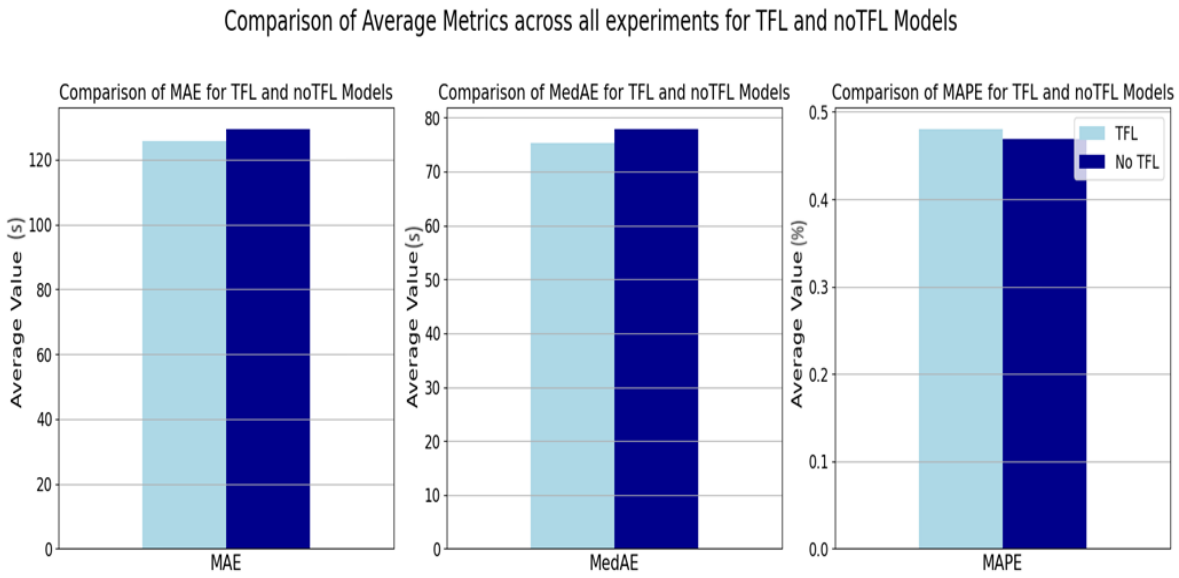

Figure 3.7 Bar plot of the average error metrics across all the experiments and all the folds for both the transfer learning and the direct training approach

The results were not what we initially expected when we thought about our studies aim because as we can see in Figure 3.7 transfer learning seems to have performed better only if we take into account the MAE and MedAE because these are the error metrics that have a lower value for the transfer learning model(labelled as TFL). We thought that these results could be due to the fact that some of the average error metrics could have been skewed by outliers and because

of that we plotted box plots of the same parameters, see Figure 3.8. We want to focus more on the median and the spread of the boxes to understand the typical range and central tendency of errors for each model.

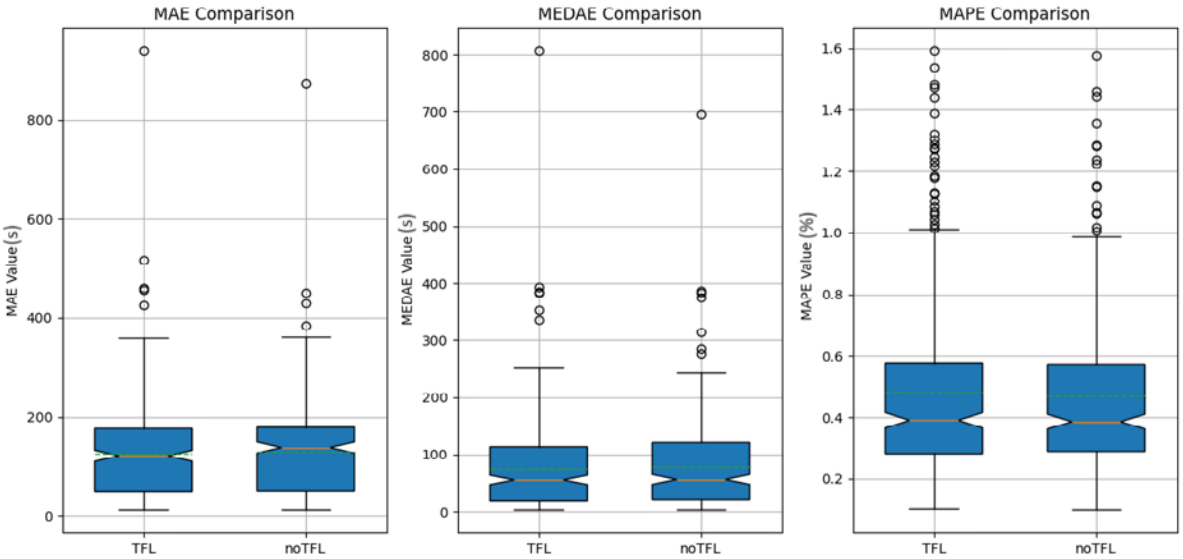

Figure 3.8 Box plots of the average error metrics for all the experiments and all the folds for both the transfer learning and the direct training approach

About the boxplots, the blue box represents the interquartile range (IQR), which spans from the 25th percentile (Q1) to the 75th percentile (Q3) of the data. The median (Q2) is marked by a solid orange line inside the box and the mean value is represented as a dashed green line. The whiskers extend from the edges of the box to show the range of the data. They extend up to 1.5 times the IQR from the edges of the box. Data points beyond this range are considered outliers and are plotted individually. In general, we can see that in the transfer learning model we have more outliers and, in most cases, they are more extreme values. The median values of the error metrics seem to be slightly in a lower height in the case of MAE and MedAE in comparison with the median of the direct trained models. The IQR is also narrower in the same error metrics suggesting that it has less variability in the middle 50% of its data compared to the approach of direct training on RepoRT, while the opposite happens in MAPE metric. The mean is exactly as we had computed earlier but this time, we can also interpret that in most cases the mean value on the MedAE and MAPE is higher than the median (there is right skewness).

In general, the results of Figure 3.7 can be explained by the outliers we see in the box plots of Figure 3.8. We continued with removing the outliers using the IQR method, meaning that data points that lie 1.5 times of IQR above Q3 and below Q1 are outliers and we remove them. From following this process we create the new box plots, Figure 3.9 where we can now clearly see that the IQR in all the error metrics is narrower in the transfer learning method. There are some new outliers in the case of MAPE but removing them again could potentially make us loose many instances.

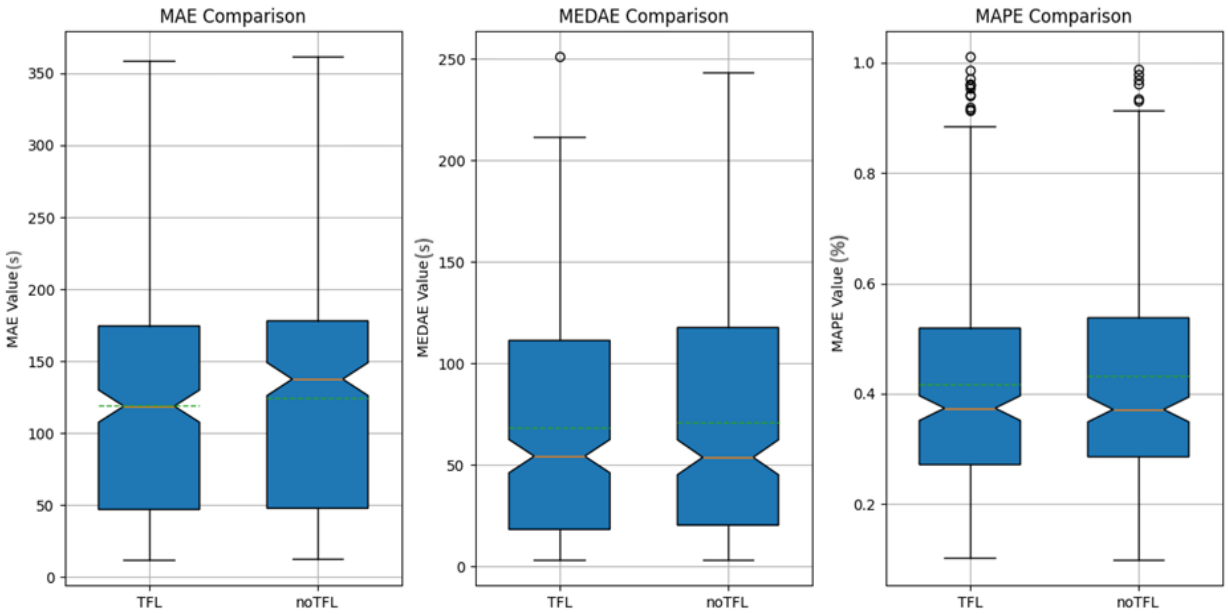

Figure 3.9 Box plots of the average error metrics for all the experiments and all the folds for both the transfer learning and the direct training approach after removing the outliers

With the new, cleared data we plot the bar plot of Figure 3.10. We can interprent that removing the outliers led to clearly showing that the transfer learning method returned overall better results taking into account all the error metrics. However, we should consider that these are the average metric of all the experiments and across all the folds, meaning that in general the transfer leraning model might work better in some experience but in others the direct training could outperform it or it may have very similar behavior.

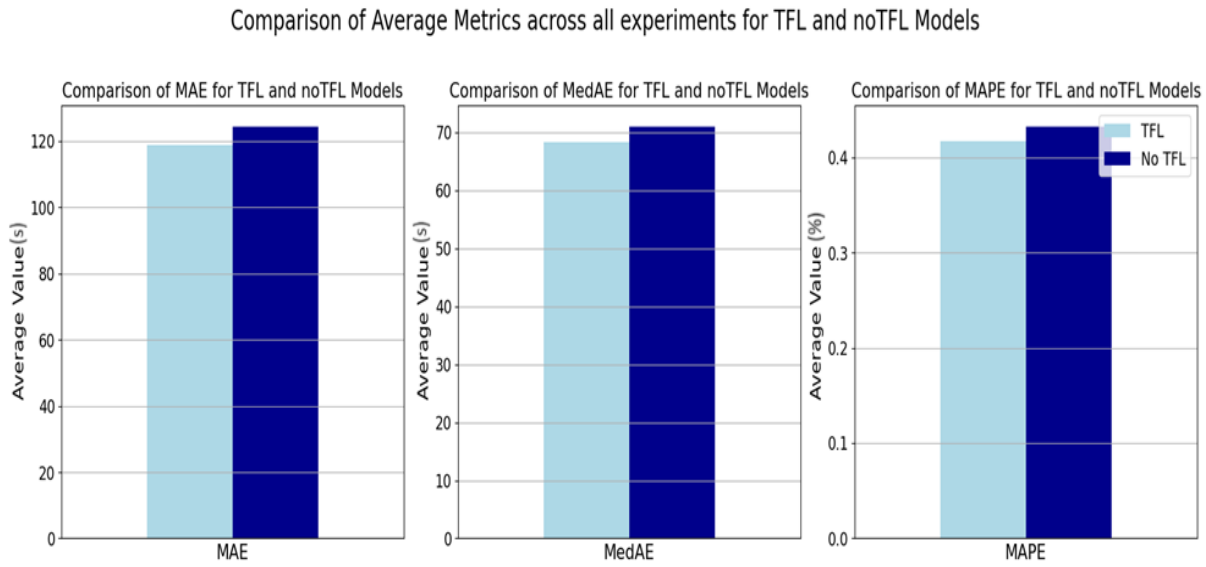

Figure 3.10 Bar plot of the average error metrics across all the experiments and all the folds for both the transfer learning and the direct training approach after removing the outliers

Then we thought it would be a good idea to plot once again another scatter plot of the average error of each metric across all the folds VS the size of each dataset to compare not only the overall performance but across each experiment as well. From Figure 3.11 and we can interpret that both models follow very similar distributions.

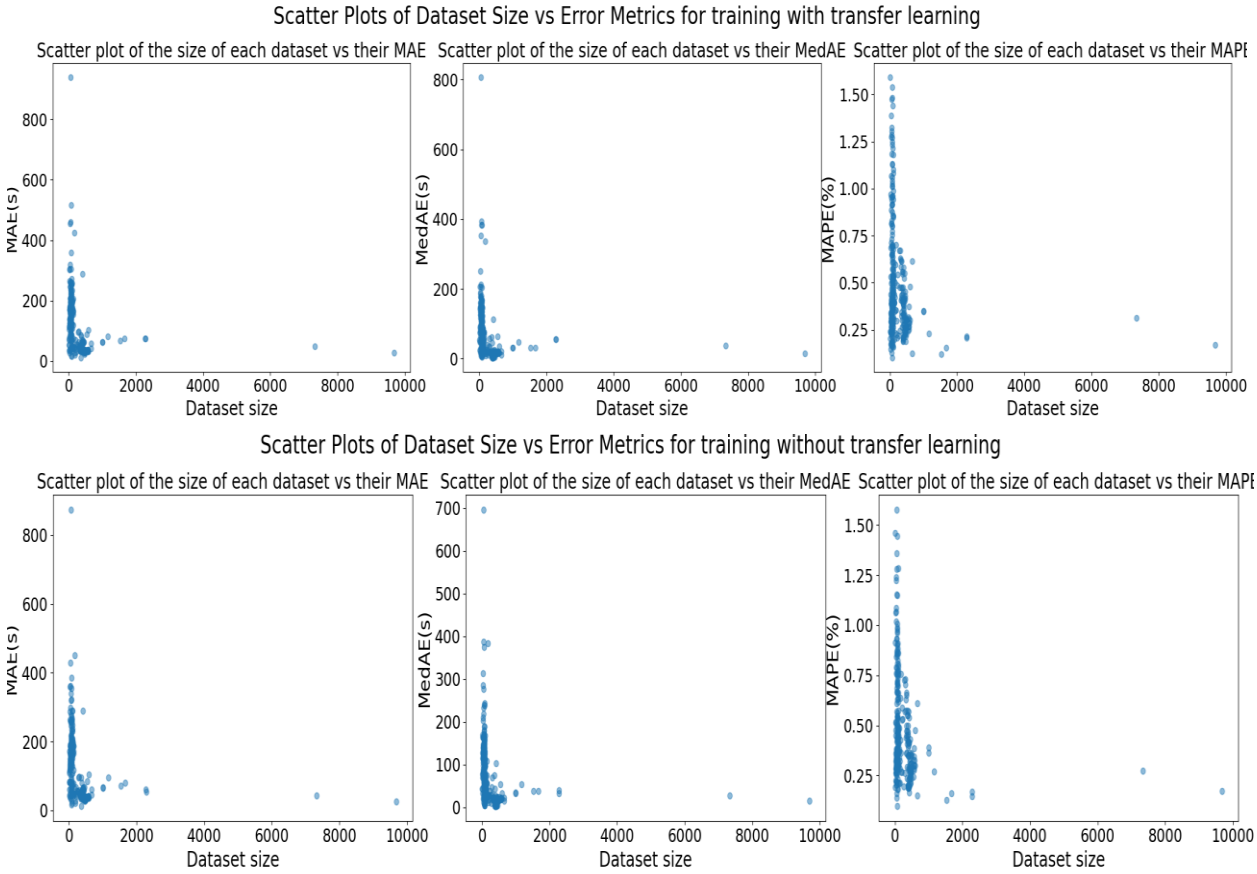

Figure 3.11 Scatter plots of the data size VS the average error metric of training with transfer learning and without

Finally, we could say that one of the clear advantages of transfer learning is the computational time. With this approach, as we previously explained, we only had to do hyperparameter optimization for 3 metrics, the learning rate, the epochs and the batch size while on the other hand for directly training the model without Transfer learning we had to optimize all the hyperparameters. Also, in the first method we did not have to do 5-fold nested cross validation which also requires a lot of time.

4. Conclusions

In this study, we explored the application of deep neural networks (DNNs) for predicting retention times (RTs) of compounds using two distinct approaches: direct training and transfer learning. Initially, we developed a robust DNN architecture on the SMRT dataset, optimizing the hyperparameters using Optuna and 5-fold nested cross-validation. We created a sequential and a functional DNN topology. Our results show a higher performance in predicting RTs of the functional DNN model based on a deep and wide architecture when compared to a more traditional sequential DNN for this problem (see Table 3.3 and Table 3.4).

Transitioning to transfer learning, we leveraged the pre-trained SMRT model to adapt and fine-tune its weights on the natural compound data from the RepoRT dataset. Following two steps, first freezing and then unfreezing layers as training strategies, we effectively transferred knowledge from synthetic to natural compounds, achieving notable improvements in prediction accuracy from step one to step two. Evaluation across various experiments within the RepoRT dataset revealed nuanced performance differences, showing diverse capabilities of the model's ability to generalize and adapt to different experimental conditions, although generally the performance of the model after the transfer learning stage improved (see Figure 3.2). We could guess that the lowest the errors the more similar the experimental conditions of the specific experiment to the one of SMRT.

Comparing the transfer learning approach with direct training on RepoRT experiments, we observed varying degrees of performance across different error metrics (MAE, MedAE, MAPE). While transfer learning demonstrated superior performance in terms of MAE and MedAE the direct training approach showed competitive results in MAPE (see Figure 3.7). Box plot analyses further highlighted the existence of many outliers, the differences in variability and central tendency of error metrics between the two approaches. After removing the outliers, we noticed that transfer learning returned better results, lower errors in all the metrics. This means that transfer learning did not work very well for some experiments which resulted in returning very large errors in our analyses.

Moreover, our study uncovered insights into the impact of experimental duration and dataset size on prediction errors, indicating complex relationships that warrant further investigation. In our study we used one pre-trained model and one final transfer learning training method for all the experiments without giving too much insight in every single one of them because they were 373 in total which does not make it very feasible. Perhaps trying different transfer learning training methods on experiments that returned big errors could potentially lower them. However, it is clear that transfer learning overall worked better and returned promising results in comparison with directly trained DNN models.

Also, notably, the computational efficiency of transfer learning emerged as a significant advantage. Transfer learning in comparison with directly trained DNNs is more computational efficient because it reduces the time required for hyperparameter optimization, as the model starts with a predefined architecture and pre-trained weights which requires fewer hyperparameter optimizations compared to direct training, and therefore less computational resources. This makes transfer learning a practical approach for large-scale applications where computational resources are limited.

In conclusion, this research contributes to advancing predictive modelling in chromatographic studies, showcasing the potential of DNNs and transfer learning to enhance accuracy and efficiency in RT prediction across diverse datasets. Future studies could explore additional datasets, refine model architectures, and investigate further the interplay between experimental variables and prediction outcomes. These efforts will continue to refine our understanding and application of deep learning in analytical chemistry and beyond.

References

[1] D. S. Wishart, (2019), "METABOLOMICS FOR INVESTIGATING PHYSIOLOGICAL AND PATHOPHYSIOLOGICAL PROCESSES", Dep. of Biol. Sciences and Comp. Science, University of Alberta, Edmonton, Alberta, Canada, doi:10.1152/physrev.00035.2018.

[2] Q. Yang, A.-h. Zhang, J.-h. Miao, H. Sun, Y. Han, Yan, Guang-li, F.-f. Wu and X.-j. Wang, (2019),"Metabolomics biotechnology, applications, and future trends: a systematic review" , RSC advances, 9(64), 37245–37257, doi: 10.1039/c9ra06697g.

[3] Baxevanis, A. D., Bader G. D., Whishart D. S., (2020), "Bioinformatics", Hoboken, NJ : Wiley, ISBN 9781119335962 (adobe pdf), 2020 fourth edition.

[4] Y. Chen, E.-M. Li and L.-Y. Xu, (2022), "Guide to Metabolomics Analysis: A Bioinformatics Workflow",15;12(4):357. DOI: 10.3390/metabo12040357.

[5] G. Theodoridis, S. Girousi, G. Zachariadis, A. Zotou and V. Samanidou, (2015),"Βιοαναλυτική Χημεία", http://dx.doi.org/10.57713/kallipos-641, ISB:978-960-603-052-9.

[6] AV. Aderemi, AO. Ayeleso, OO. Oyedapo, E. Mukwevho, (2021),"Metabolomics: A Scoping Review of Its Role as a Tool for Disease Biomarker Discovery in Selected Non-Communicable Diseases", 11(7):418. https://doi.org/10.3390/metabo11070418.

[7] A. Gil-de-la-Fuente, C. Barbas and A. Otero, “Design, validation and implementation of a software tool for metabolites annotation and identification.,” 2019.

[8] "Untargeted Metabolomics Workflows - ES". [Online]. Available: https://www.thermofisher.com/es/es/home/industrial/mass-spectrometry/mass-spectrometry-learning-center/mass-spectrometry-applications-area/metabolomics-mass-spectromet.

[9] V. Spyropoulos, (2015). "Εισαγωγή στην τεχνολογία χειρουργείου, εντατικής και επείγουσας ιατρικής [Chapter 8]", kalipos, http://hdl.handle.net/11419/3030.

[10] D. C. Harris, (2007), "Quantitative Chemical Analysis", 7th ed., Craig Bleyer, ISBN 0-7167-7041-5.

[11] "Types Of Chromatography In Pharmaceuticals 2023", Blog title: Flair Pharma The Knowledge Kit. URL: https://flairpharma.com/types-of-chromatography-in-pharma/.

[12] P. Karkalousos, G. Zoi, C. Kroupis, A. Papaioannou, P. Plageras, V. Spyropoulos, G. Tsotsou and C. Fountzoula, (2015). "Υγρή χρωματογραφία υψηλής πίεσης

(απόδοσης) στην κλινική χημεία. Βασικές αρχές και παραδείγματα (chapter 8)", Kallipos http://hdl.handle.net/11419/5388.

[13] R. E. Ardrey, (2005), "Liquid Chromatography – Mass Spectrometry: An Introduction", John Wiley & Sons, Ltd, ebook ISBN:9780470867297.

[14] N. H, C. S. Reddy, K. Bandarapalle, K. H. Priya, K. Madhavi and K. J. Reddy, (2023). "Review on Chromatographic Fingerprint Analysis of Herbal Medicines", Future Journal of Pharmaceuticals and Health Sciences, 3(1), 69–79. https://doi.org/10.26452/fjphs.v3i1.342.

[15] M. O'Brien, (2018), CHEM133 Mass Spectrometry: Lecture 1[24]. SlidePlayer. URL: https://slideplayer.com/slide/12300829/.

[16] P. Sudhakar, P. Latha and P. V. Reddy, (2016), "Phenotyping Crop Plants for Physiological and Biochemical Traits", Academic Press, ISBN 9780128041109..

[17] VH. Wysocki, KA. Resing, Q, Zhang, G. Cheng, (2005), "Mass spectrometry of peptides and proteins"., 35 (3): 211–22. doi:10.1016/j.ymeth.2004.08.013.

[18] R. Karlsson, A. Thorsell, M. Gomila, F. Salvà-Serra, H. E. Jakobsson, L. Gonzales-Siles, D. Jaén-Luchoro, S. Skovbjerg, J. Fuchs, A. Karlsson and F. Boulund, (2020), "Discovery of Species-unique Peptide Biomarkers of Bacterial Pathogens by Tandem Mass Spectrometry-based Proteotyping". Molecular & Cellular Proteomics. 19 (3): 518–528. doi:10.1074/mcp.RA119.001667.

[19] T. F. Jorge, J. A. Rodrigues, C. Caldana, R. Schmidt, J. T. van Dongen, J. Thomas-Oates and C. António, (2016), "Mass spectrometry-based plant metabolomics: Metabolite responses to abiotic stress". Mass Spectrometry Reviews. 35(5): 620–649 doi:10.1002/mas.21449.

[20] C. H. B. Rao, 2011, LC/MS : AN ESSENTIAL TOOL IN DRUG DEVELOPMENT, Int. J. Adv. Pharm. Anal., DOI:10.7439/ijapa.v1i2.10 .

[21] D. Y. Lin, C. Y. Yu, C. A. Ku and C. K. Chung, (2023), "Design, Fabrication, and Applications of SERS Substrates for Food Safety Detection: Review" Micromachines 14, no. 7: 1343. https://doi.org/10.3390/mi14071343.

[22] A. Vipin, "Fundamentals Of Liquid Chromatography Mass Spectrometry (LC-MS/MS)", Blog title: NorthEast BioLab. URL: https://www.nebiolab.com/complete-guide-on-liquid-chromatography-mass-spectrometry-lc-ms/.

[23] "GenBank Overview., URL: https://www.ncbi.nlm.nih.gov/genbank/".

[24] SK. Burley, HM. Berman, GJ. Kleywegt, JL. Markley, H. Nakamura, S. Velankar, (2017),"Protein Data Bank (PDB): The Single Global Macromolecular Structure Archive". Mol Biol. ;1607:627-641. doi: 10.1007/978-1-4939-7000-1_26.

[25] “METLIN | Scripps Research, URL: https://metlin.scripps.edu/landing_page.php?pgcontent=simple_search”.

[26] “UniProt, https://www.uniprot.org/”.

[27] “KEGG, https://www.kegg.jp/”.

[28] “Human Metabolome Database. Available: https://hmdb.ca/”.

[29] “ChemSpider | Available: https://www.chemspider.com/”.

[30] “MassBank. Available: https://massbank.eu/MassBank/About”.

[31] “LIPID MAPS. Available: https://lipidmaps.org/databases”.

[32] J. Fine, A.K.P. Mann, P. Aggarwal, (2024) "Structure Based Machine Learning Prediction of Retention Times for LC Method Development of Pharmaceuticals". Pharm Res 41, 365–374 . https://doi.org/10.1007/s11095-023-03646-2.

[33] D. Song, T. Tang, R. Wang, H. Liu, D. Xie, B. Zhao, et al., (2024). "Enhancing compound confidence in suspect and non-target screening through machine learning-based retention time prediction". Environmental Pollution, 347, 123763,, https://doi.org/10.1016/j.envpol.2024.123763.

[34] H. A. Noreldeen, X. Liu, X. Wang, Y. Fu, Z. Li, X. Lu, et al., (2018). "Quantitative structure-retention relationships model for retention time prediction of veterinary drugs in food matrixes". International Journal of Mass Spectrometry, 434, 172-178, https://doi.org/10.1016/j.ijms.2018.09.022.

[35] X.A. Domingo, C. Guijas, E. Billings, et al., (2019) "The METLIN small molecule dataset for machine learning-based retention time prediction". Nat Commun.;10(1):5811. . doi:10.1038/s41467-019-13680-7.

[36] Leszczynski, J., Kaczmarek-Kedziera, A., Puzyn, T., Papadopoulos, M. G., Reis, H., Shukla, M. K., (2017). "Handbook of computational chemistry" (2nd ed.). Springer Science. https://doi.org/10.1007/978-3-319-27282-5.

[37] A. Mauri, (2020), "AlvaDesc: A tool to calculate and analyze molecular descriptors and fingerprints. Ecotoxicological QSARs", Springer, New York, pp 801–820.

[38] J. Gasteiger, T. Engel, (2003), “Chemoinformatics–A Textbook”, Wiley-VCH Verlag GmbH & Co. KGaA., ISBN 3-527-30681-1.

[39] Phyo Phyo Kyaw Zin,(2020), "Computing Molecular Descriptors – Part 1", Blog name: The journey of a cheminformatics scientist, URL:https://drzinph.com/computing-molecular-descriptors-intro/?preview=true&_thumbnail_id=206.

[40] Boldini, D., Ballabio, D., Consonni, V., et all., (2024), "Effectiveness of molecular fingerprints for exploring the chemical space of natural products". Journal of Cheminformatics, 16(35). https://doi.org/10.1186/s13321-024-00830-3.

[41] Cereto-Massagué, A., Ojeda, M.J., Valls, C., Mulero, M., García-Vallvé, S., & Pujadas, G. (2015). "Molecular fingerprint similarity search in virtual screening". Methods, 71, 58-63 . DOI: 10.1016/j.ymeth.2014.08.005.

[42] D. Rogers, M. Hahn, (2010), "Extended-Connectivity Fingerprints", J. Chem. Inf. Model., vol. 50, no. 5, pp. 742–754, doi: 10.1021/ci100050t..

[43] A, Cereto-Massagué; Ojeda MJ, Valls C, et al., (2015)," Molecular fingerprint similarity search in virtual screening", Methods, 71, pp 58–63, https://doi.org/10.1016/j.ymeth.2014.08.005.

[44] K. Gao, DD. Nguyen, Sresht V. et al., (2020), "Are 2D fingerprints still valuable for drug discovery?", Phys Chem Chem Phys, 22, (16), pp 8373–8390 , https://doi.org/10.1039/d0cp00305k.

[45] G. Rebala, A. Ravi and S. Churiwala, (2019), "An Introduction to Machine Learning", Springer Nature Switzerland, doi:10.1007/978-3-030-15729-6 , ISBN 978-3-030-15729-6.

[46] X. Yang, Y. Wang, R. Byrne, G. Schneider and S. Yang, (2019), “Concepts of Artificial Intelligence for Computer-Assisted Drug Discovery”, Chem. Rev., vol. 119, no. 18, pp. 10520–10594, doi: 10.1021/acs.chemrev.8b00728..

[47] A. Otero, (2024), "Data mining in biomedicine", CEU San Pablo Madrid, https://ceu.blackboard.com/ultra/courses/_303776_1/outline.

[48] S. Jayatilake, G.U. Ganegoda, (2021), "Involvement of Machine Learning Tools in Healthcare Decision Making", J Healthc Eng, 2021, 6679512 https://doi.org/10.1155/2021/6679512.

[49] T. Jo, (2021), "Machine Learning Foundations Supervised, Unsupervised, and Advanced Learning", Springer Nature Switzerland, ISBN 978-3-030-65900-4 (eBook) https://doi.org/10.1007/978-3-030-65900-4.

[50] K. K. Hiran, R. K. Jain, K. Lakhwani and R. Doshi, (2021), "Machine Learning: Master Supervised and Unsupervised Learning Algorithms with Real Examples",(English Edition). Bpb Publications. [Online]. Available: https://books.google.gr/books?id=4VVDE.

[51] S. IH, (2021), "Machine Learning: Algorithms, Real-World Applications and Research Directions", SN Comput Sci, 2, (3), pp 160 https://doi.org/10.1007/s42979-021-00592-x.

[52] R. Sengupta, D. Sengupta, A. K. Kamra and D. Pandey, (2020)," Artificial intelligence and quantum computing for smarter wireless network", journal of critical reviews, vol 7, issue 19, ISSN-2394-5125, DOI:10.31838/jcr.07.19.21.

[53] M. Negnevitsky, 2002, "Artificial Intelligence: A Guide to Intelligent Systems", Addison-Wesley. https://books.google.gr/books?id=PgxmQgAACAAJ.

[54] A. Dounis, (2023)"ΤΕΧΝΗΤΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ". Πανεπιστήμιο Δυτικής Αττικής, https://eclass.uniwa.gr/modules/document/?course=252.

[55] Z. Meng, Y. Hu and C. Ancey, (2020), "Using a Data Driven Approach to Predict Waves Generated by Gravity Driven Mass Flows" Water 12, no. 2: 600. https://doi.org/10.3390/w12020600.

[56] A. Géron, (2019), "Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems". O'Reilly Media. [Online]. Available: https://books.google.gr/books?id=HHetDwAAQBAJ.

[57] "Regression losses' URL: https://keras.io/2.16/api/losses/regression_losses/#meanabsolutepercentageerror-function".

[58] "'Median Absolute Error - Inside Learning Machines.' ,URL. Available: https://insidelearningmachines.com/median_absolute_error/".

[59] F. M. Salem, (2022), 'Recurrent Neural Networks From Simple to Gated Architectures', ISBN 978-3-030-89929-5 (eBook) https://doi.org/10.1007/978-3-030-89929-5.

[60] A. Khan, M. M. Fouda, D. -T. Do, A. Almaleh and A. U. Rahman, (2023), "Short-Term Traffic Prediction Using Deep Learning Long Short-Term Memory: Taxonomy, Applications, Challenges, and Future Trends," in IEEE Access, vol. 11, pp. 94371-94391, doi: 10.1109/ACCESS.2023.3309601.

[61] S. Singh and A. Mahmood, (2021). "The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures". IEEE Access, 9, 68675-68702..

[62] A. Géron, (2018), "Neural networks and deep learning", O'Reilly Media, Inc., ISBN: 9781492037347.

[63] Amazon Web Service, (2024)"What is Data Augmentation? - Data Augmentation Techniques Explained - AWS," s, Inc., URL: https://aws.amazon.com/what-is/data-augmentation/.

[64] L. Alzubaidi, J. Bai, A. Al-Sabaawi, et al. (2023). "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications", J Big Data 10, 46. https://doi.org/10.1186/s40537-023-00727-2.

[65] Q. Sun, Y. Liu, Z. Chen, T. S. Chua and B. Schiele, (2022), “Meta-Transfer Learning Through Hard Tasks” ,IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 3, pp. 1443–1456, doi: 10.1109/TPAMI.2020.3018506..

[66] W. Guo, Y. Dong and G. F. Hao, (2024), “Transfer learning empowers accurate pharmacokinetics prediction of small samples,” Drug Discov., vol. 29, no. 4, p. 103946, doi: https://doi.org/10.1016/j.drudis.2024.103946.

[67] C. Cai, S. Wang, Y. Xu, W. Zhang, K. Tang, Q. Ouyang, L. Lai and J. Pei, (2020),“Transfer Learning for Drug Discovery”, Journal of Medicinal Chemistry , vol. 63, no. 16, pp. 8683–8694, doi: 10.1021/acs.jmedchem.9b02147.

[68] C. Iorga and V. -E. Neagoe, (2019)," A Deep CNN Approach with Transfer Learning for Image Recognition", 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, pp. 1-6, doi: 10.1109/ECAI46879.2019.9.

[69] S. Ruder, M. E. Peters, S. Swayamdipta and T. Wolf, (2019), "Transfer Learning in Natural Language Processing". Conference of the North American Chapter of the Association for Computational Linguistics. https://aclanthology.org/N19-5004/.

[70] N. M. Khan, N. Abraham, M. Hon and L. Guan, (2019),"Machine Learning on Biomedical Images: Interactive Learning, Transfer Learning, Class Imbalance, and Beyond", IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), San Jose, CA, USA, pp. 85-90, doi: 10.1109/MIPR.2019.00023..

[71] A. Dalby, J.G. Nourse, W.D. Hounshell, A.K. Gushurst, D.L. Grier, B.A. Leland, J. Laufer, (1992), "Description of several chemical structure file formats used by computer programs developed at molecular design limited", J Chem Inf Comput Sci 32(3):244–255.

[72] A. St-Gelais, (2016), "Natural vs Synthetic: Clarifying the Terms", Laboratoire PhytoChemia, URL: https://phytochemia.com/en/2016/03/17/natural-vs-synthetic-clarifying-the-terms/.

[73] S.R. Johnson, B.M. Lange, (2015), "Open-access metabolomics databases for natural product research: present capabilities and future potential". Front Bioeng Biotechnol. 4;3:22. doi: 10.3389/fbioe.2015.00022. PMID: 25789275; PMCID: PMC4349186..

[74] “ https://github.com/KatsaraVasiliki/Vasiliki_regressor.git”.

[75] “scikit-learn: machine learning in Python — scikit-learn 1.5.0 documentation.” . Available: https://scikit-learn.org/stable/.

[76] “'Why TensorFlow', TensorFlow. URL: https://www.tensorflow.org/about”.

[77] “'Keras: Deep Learning for humans', URL: https://keras.io/”.

[78] “Optuna - A hyperparameter optimization framework,” https://optuna.org/.

[79] “Alvascience: AlvaDesc (software for Molecular Descriptors Calculation). https://www.alvascience.com”.

[80] “Alvascience: alvaDesc Molecular Descriptors, (2021), https://www.alvascience.com/alvadescdescriptors/”.

[81] S. Böcker and M. Witting, (2024), "RepoRT", https://github.com/michaelwitting/RepoRT.git.

[82] F. Kretschmer, E.M. Harrieder, M.A. Hoffmann, et al., (2024) "RepoRT: a comprehensive repository for small molecule retention times". Nat Methods 21, 153–155. https://doi.org/10.1038/s41592-023-02143-z.

[83] C. García, A. Gil-de-la-Fuente, C. Barbas and O. A., (2022). "Probabilistic metabolite annotation using retention time prediction and meta-learned projections". J Cheminform 14, 33, https://doi.org/10.1186/s13321-022-00613-8.

[84] “https://github.com/grfone/regressor”.