Diplomarbeit

Deep learning-based metabolite quantification in proton magnetic resonance spectroscopy of the brain

Author:
Tobias Klein, B.Sc.

Supervisors:
Em. Univ. Prof. DI Dr. Gerald Badurek
Vienna University of Technology
Assoc. Prof. DI Dr. Wolfgang Bogner
Medical University of Vienna

October 15, 2020

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references.

I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database where it shall also remain after examination, to enable comparison with future theses submitted. Further rights of reproduction and usage, however, are not granted here.

This paper was not previously presented to another examination board and has not been published.

Signature:

Name (in capitals):

Date of submission:

Dedication

I dedicate this thesis to my greatest supporters, my parents Pia and Thomas Klein, and my grandma, Felicitas Berg, who always have my back. Without their love and support, I would never have come this far. I thank you and I love you.

Ich widme diese Arbeit meinen Eltern, Pia und Thomas Klein, und meiner Großmutter, Felicitas Berg, ohne deren unerbitterliche Unterstützung und Liebe dies niemals möglich gewesen wäre. Ich danke Euch und liebe Euch.

Abstract

Proton magnetic resonance spectroscopy of the brain is a noninvasive technique to extract neurochemical information from the brain and allow the analysis of primary and secondary brain tumors and metabolic diseases. Acquired spectra are inevitably degraded due to many factors, such as line-broadening, low signal-to-noise ratio, overlapping metabolic signals, and variability in spectral baselines. Current software for spectral fitting and metabolite quantifications are based on nonlinear least squares fitting algorithms. These approaches are not suitable for all MR metabolic signals and are, moreover, very time consuming. Thus, magnetic resonance spectroscopic imaging still is rarely used in daily clinical routine regardless of its high potential.

This thesis investigates how deep learning can accelerate spectral fitting and metabolite quantification. A convolutional neural network (CNN) called Superfit and consisting of two serialized autoencoders, was developed to remove the baseline from the original spectra and to disassemble the baseline free spectra into their metabolic components. For training, validation, and testing of Superfit, almost 70,000 spectra were simulated based on a basis set of the metabolites of interest and *in vivo* spectral baselines from earlier studies.

The trained Superfit was capable of disassembling the metabolites and quantifying the concentrations of the total metabolites with a mean absolute percentage error of 8.99% $\pm$ 7.41%. Furthermore, Superfit could fit and quantify over 10000 spectra in less than 40s implying that subminute metabolite quantification via deep learning is possible.

Acknowledgments

I would like to express my deep and sincere gratitude to my thesis supervisors, Em. Univ. Prof. DI Dr. Gerald Badurek, Vienna University of Technology, and Assoc. Prof. DI Dr. Wolfgang Bogner, Medical University of Vienna, for giving me the opportunity to perform research and providing invaluable guidance throughout this thesis. It was a great privilege and honor to work and study under their guidance. I am extremely grateful for what they have offered me. I would also like to thank DI Stanislav Motyka for his help, professionalism, patience, and jokes, which made it a pleasure to work here.

Contents

1 Introduction

In this very first chapter, after a brief introduction to the physical basics of nuclear magnetic resonance (NMR), the basic concepts of proton magnetic resonance (MR) imaging (MRI) and magnetic resonance spectroscopy (MRS) are described. The following section follows with a focus on metabolic quantification in MRSI, especially via LCModel, which is still the method primarily used for quantification. Finally, the concept of deep learning and deep neural networks, the tools that used in the Methods chapter of this thesis, is introduced. For a more detailed introduction to the physical and technical NMR-related part, "*In Vivo NMR Spectroscopy*" by de Graaf [1] is highly recommended, whereas readers interested in further information about LCModel should consult Provencher's "*Estimation of Metabolite Concentrations from Localized in Vivo Proton NMR Spectra*" [2]. A broad introduction to deep learning and its tools is presented in "*Deep Learning*" by Goodfellow et al. [3].

1.1 Introduction to MR Spectroscopic Imaging

MRSI is a noninvasive technique to extract biochemical information like metabolite concentrations from a body tissue of interest, whereas MRI obtains only information about the structure of the tissue. Thus, MRSI has a great potential for clinical and scientific purposes. For example, it can be used to examine prostate cancer, as reported by Müller-Lisse et al. [4]. The main medical focus of MRSI is set on the brain: Cecil and colleagues used MRSI spectra to investigate many metabolic diseases, including lysosomal disorders, peroxisomal disorders, mitochondrial disorders, white-matter disorders, disorders of amino and organic acid metabolism, and some miscellaneous disorders [5]. One of the most prominent MR-visible metabolites is the neuronal marker N-acetylaspartate (NAA). A reduced concentration in NAA is associated with white mat-

ter diseases and brain tumors [6]. Creatine (Cr) and phosphocreatine (PCr) are brain metabolites, which play a role in the energy metabolism. Although their concentrations may be reduced by pathologies like brain tumors, their concentrations are considered to be stable and, thus, can be used to calculate the concentrations of other metabolites [6]. Phosphocholine (PC) and glycerophosphocholine (GPC) are involved in the synthesis of membrane; their concentrations correlate with the malignancy of tumors [6]. Other prominent MR-visible metabolites include myo-inositol (m-Ins), glutamate (Glu) and glutamine (Gln). The neurotransmitter with the highest inhibitory effect is gamma aminobutyric acid (GABA) and is related to several nervous system disorders; it was reproducibly quantified quantified by Bogner et al. [7].

In MR spectroscopy, as written in the word itself, the goal is to observe spectra which enable conclusions to be derived about the concentration of a certain molecule. This observation is achieved by the effect of the magnetic resonance of nuclei. Therefore, before getting to the basics of MRI and MRS, it makes sense to understand the physical principles of NMR, which will be described in a semi-classical way in the following.

1.1.1 Physical Basics of NMR

NMR relies on the quantum mechanical concept of nuclear spin. If a particle rotates about a fixed point, its motion is referred to as the angular momentum. In quantum mechanics, the angular momentum L of elementary particles is assumed to have discrete values and is quantized and the amplitude specified by

$$L = \hbar\sqrt{I(I+1)} \tag{1.1}$$

with $\hbar$ being the reduced Planck's constant (Planck's constant h divided by 2π), and I being the either integral or half-integral spin number. The direction of the rotation, i.e., the direction of angular momentum, is defined by the second quantum number m, which can have $2I+1$ values given by

$$m = I, \quad I-1, \quad I-2, \quad \ldots, \quad -I,$$

with respect to a given direction z. For such given direction, L_z denotes

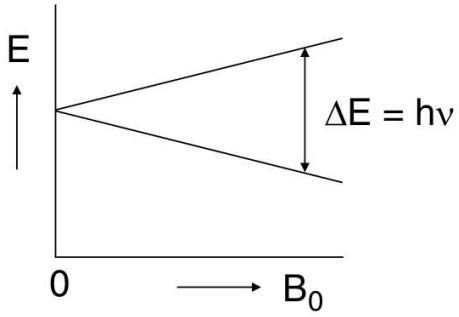

Figure 1.1: The two energetic states of a particle with spin $I = 1/2$ split when an external magnetic field B_0 is applied. This effect is called Zeeman Splitting [1].

the corresponding part of the angular momentum and is specified by

$$L_z = \hbar m \tag{1.2}$$

In NMR, especially in medical context, nuclei of major importance are those with a spin number $I = 1/2$, primarly ^{1}H and ^{31}P.

Related to the angular momentum, elementary particles further have a magnetic moment $\boldsymbol{\mu}$,

$$\boldsymbol{\mu} = \gamma \mathbf{L}, \tag{1.3}$$

where γ denotes the gyromagnetic ratio. Analogously to L_z, the component of the magnetic moment with respect to the longitudinal axis z, can be quantized as

$$\mu_z = \gamma L_z = \gamma \hbar m \tag{1.4}$$

If an external magnetic field B_0 is applied, magnetic energy E is acquired by the particle:

$$E = -\mu_z B_0 = -\gamma \hbar m B_0 \tag{1.5}$$

Hence, for particles with spin number $I = 1/2$, only two energy levels, $m = \pm 1/2$, exist and the energy difference, ΔE, is shown in Figure 1.1 and specified by

$$\Delta E = \gamma \hbar B_0. \tag{1.6}$$

If an oscillating magnetic field perpendicular to μ_z with a frequency of ν_0 is applied, the energy arising from the electromagnetic wave is given by

$$E = 2\pi \hbar \nu_0. \tag{1.7}$$

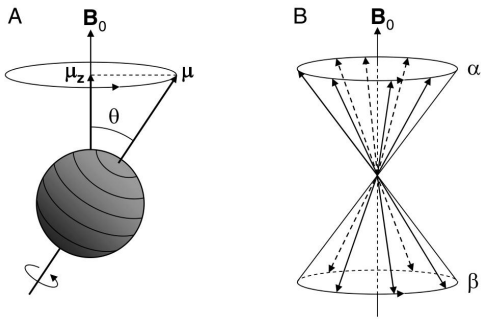

Figure 1.2: (A) A particle with a spin $I = 1/2$ has a precessing magnetic moment with a Larmor frequency ν_0 and an angle θ between μ and μ_z.. (B) The position of the magnetic moment depends on m and is parallel, i.e. has low energy, for $m = +1/2$ and is denoted as α. The antiparallel state $m = -1/2$ has high energy and is denoted as β. The possible positions of μ form the surface of two cones [1].

If both, ΔE given by equation (1.6) and E given by equation (1.7) have the same energy, the fundamental resonance phenomenon of NMR is achieved and the important Larmor equation is obtained by

$$\nu_0 = \left(\frac{\gamma}{2\pi}\right) B_0 \tag{1.8}$$

weher ν_0 is called the Larmor frequency, which denotes the frequency at which a particle's magnetic moment precesses around the external magnetic field B_0.

Considering just one particle, there is just one energetic state, as show in Figure (1.2 A). But, in a sample, there is a whole population of particles and the corresponding spins and realizations (see Figure (1.2 B)).

The magnetization M of a sample is the sum over all magnetic moments within the sample. To detect a sample's nuclear magnetization M, the net longitudinal magnetization M_0 in equilibrium induced by B_0 must be rotated toward the transverse plane by applying a second magnetic field B_1. B_1 oscillates in radio frequency (RF) and is usually applied as an RF pulse, i.e. turned on for a certain time and then turned off again. Due to B_1, the initially equilibrial longitudinal magnetization M_0 receives a torque and starts rotating towards the transverse plane (see Figure 1.3). This generates a transverse magnetization that rotates around B_0 at the Larmor frequency, ν_0. This process is called excitation and induces an electromotive force (EMF) that can be measured by the receiver coil surrounding the sample. In NMR, the magnetization M

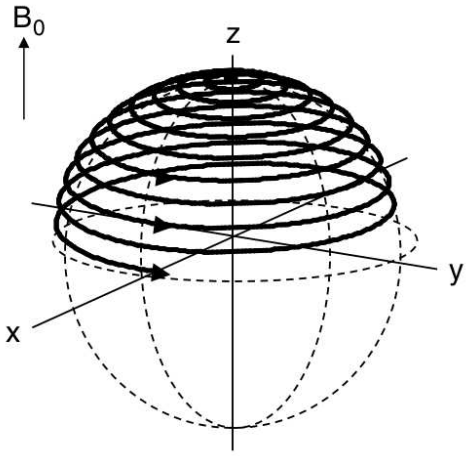

Figure 1.3: The initially longitudinal magnetization by B_0 rotates toward the XY-plane due to B_1 [1].

itself is not measured, but rather, the relaxation or decay back to the equilibrium is measured right after excitation through the RF pulse.

The magnetization of a sample in a laboratory frame is described by the so-called Bloch equations:

$$\begin{aligned}
\frac{dM_x(t)}{dt} &= \gamma[M_y(t)B_0 - M_z(t)B_{1y}] - \frac{M_x(t)}{T_2} \\
\frac{dM_y(t)}{dt} &= \gamma[M_z(t)B_{1x} - M_x(t)B_0] - \frac{M_y(t)}{T_2} \\
\frac{dM_z(t)}{dt} &= \gamma[M_x(t)B_{1y} - M_y(t)B_{1x}] - \frac{M_z(t) - M_0}{T_1}
\end{aligned} \tag{1.9}$$

The first part of each equation specifies the excitation process, and the second part part pictures the relaxation process. B_{1x}, and B_{1y} denote the components of B_1 in the direction of the x- and y-axis, respectively. The transverse components of M (M_x and M_y) relax with a different time constant T_2 than the longitudinal component M_z which relaxes with time constant T_1.

1.1.2 Basics of NMR Spectroscopy

In NMR spectroscopy, the signal of interest decreases according to the transverse magnetization right after the RF pulse is detected by the receiver coil. Due to the transverse relaxation time T_2, the EMF measured in the receiver coil decreases exponentially as a function of time. This decay is called free induction decay (FID). However, because of the sample's macroscopic and microscopic inhomogeneities in B_0, we see a distribution

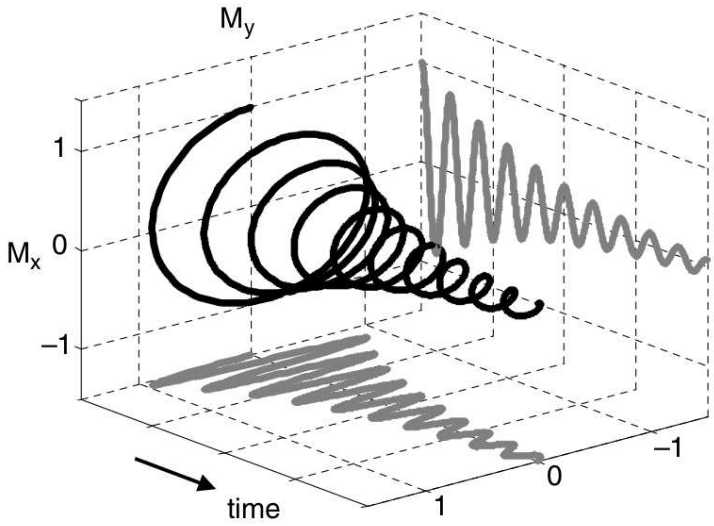

Figure 1.4: The FID as a complex function of time detected as a real and an imaginary FID component on the x- and y-axis respectively [1].

of locally different Larmor frequencies. In praxis, this results in a faster decrease of the transverse magnetization compared to only T_2 relaxation:

$$M_{xy}(t) = M_{xy}(0) \exp\left(\frac{-t}{T_2^*}\right) \tag{1.10}$$

with M_{xy} being the transverse magnetization right at the end of the RF pulse and t denoting the time. T_2^* denotes the the adapted transverse relaxation constant described by:

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{1}{T_2'} \tag{1.11}$$

where T_2' is inversely proportional to the mentioned macroscopic field inhomogeneity.

Fourier Transform NMR

Since the transverse magnetization M_{xy} rotates, its motion and thus the FID is a complex function over time:

$$M_{xy} = M_x + iM_y \tag{1.12}$$

$$\begin{aligned} M_x(t) &= M_0 \cos[(\omega_0 - \omega)t + \phi] \exp\left(\frac{-t}{T_2^*}\right) \\ M_y(t) &= M_0 \sin[(\omega_0 - \omega)t + \phi] \exp\left(\frac{-t}{T_2^*}\right) \end{aligned} \tag{1.13}$$

$\omega_0 - \omega$ describes the reduced precessional frequency. ϕ denotes the phase at $t = 0$. The receiving FID signal contains all the relevant information and M_x and M_y as the real and imaginary part of the FID are detected separately using the so-called quadrature detection. However, this time-domain data is usually converted to frequency-domain data, i.e. the spectrum. This is achieved by a Fourier transformation:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \exp(-i\omega t) dt \quad \text{or} \quad F(\nu) = \int_{-\infty}^{\infty} f(t) \exp(-i2\pi\nu t) dt \tag{1.14}$$

Fourier transforming the time-domain signal yields the real and imaginary part of the spectrum:

$$\begin{aligned} R(\omega) &= A(\omega) \cos\phi - D(\omega) \sin\phi \\ I(\omega) &= A(\omega) \sin\phi + D(\omega) \cos\phi \end{aligned} \tag{1.15}$$

with

$$\begin{aligned} A(\omega) &= \frac{M_0 T_2^*}{1 + (\omega_0 - \omega)^2 T_2^{*2}} \\ D(\omega) &= \frac{M_0 (\omega_0 - \omega)^2 T_2^{*2}}{1 + (\omega_0 - \omega)^2 T_2^{*2}} \end{aligned} \tag{1.16}$$

where $A(\omega)$ and $D(\omega)$ represent the absorption and dispersion parts of Lorentzian lineshapes. In ideal circumstances, the phase ϕ equals zero and the absorption and dispersion match the real and imaginary part, respectively. However, in praxis, both parts are mixed and a so called phasing must be done by:

$$\begin{aligned} A(\omega) &= R(\omega) \cos\phi_c + I(\omega) \sin\phi_c \\ D(\omega) &= I(\omega) \cos\phi_c - R(\omega) \sin\phi_c \end{aligned} \tag{1.17}$$

with ϕ_c being the phase correction performed according to:

$$\phi_c = \phi_0 + (\omega_0 - \omega)\phi_1 \tag{1.18}$$

Chemical Shift

Typically, the examined sample does not consist of only one type of spin that shows one resonance frequency. Instead, nuclei of the same element resonate at different frequencies since not only the gyromagnetic ratio γ and the external magnetic field, but also the chemical environment of the nuclei within a molecule, affects the resonance frequency. This effect is referred to as the chemical shift and is represented by a dimensionless shielding constant, σ. Thus, the resonance condition must be updated by modifying the equation (1.8) to:

$$\nu = \left(\frac{\gamma}{2\pi}\right) B_0(1-\sigma) \tag{1.19}$$

Since this expression depends on the magnetic field strength, it is rather cumbersome to compare to chemical shifts with different external magnetic fields. Therefore, chemical shifts are expressed in parts per million (ppm) and conveniently are defined by:

$$\delta = \frac{\nu - \nu_{ref}}{\nu_{ref}} \cdot 10^6 \tag{1.20}$$

with ν being the frequency of the investigated compound and ν_{ref} representing the frequency of a reference compound. Common reference compounds include 3-(trimethylsilyl) propionate (TSP) or 2,2-dimethyl-2-silapentane-5-sulfonate (DSS), which both resonate independently of pH and temperature at a certain frequency and are usually placed adjacent to the sample.

Digital Fourier Transform NMR

Two indicators of the spectral quality are the signal-to-noise ratio (SNR) and the full width at half maximum (FWHM) $\nu_{1/2}$ which is $1/(2\pi T_2^*)$ for the absorption. In digital Fourier transform NMR, SNR, and FWHM can be improved by multiple manipulations of the spectrum. By averaging n consecutive, identical observations, the SNR is increased by a factor of $\sqrt{n}$.

Another modification to improve the SNR is the so-called time-domain filtering, which multiplies the original FID with a filter function:

$$f_{filtered}(t) = f_{original}(t) \cdot f_{filter}(t) \tag{1.21}$$

The idea is to attach importance to the beginning of the FID, which has a higher SNR than the end of the FID. This can be achieved by an exponentially decreasing function that apodizes the end of the FID. A commonly used filter function is exponential weighting:

$$f_{filter}(t) = \exp\left(\frac{-t}{T_w}\right) \tag{1.22}$$

which becomes a decreasing monoexponential function for $T_W > 0$. In addition to the SNR improvement, the FWHM also becomes larger since the resultant T^*_{2W} becomes:

$$\frac{1}{T^*_{2W}} = \frac{1}{T^*_2} + \frac{1}{T_W}$$

Another widely used filter function is the so-called Lorentz-Gaussian filtering defined by:

$$f_{filter}(t) = \exp\left(\frac{+t}{T_L}\right) \exp\left(\frac{-t^2}{{T_G}^2}\right) \tag{1.23}$$

While the first exponential function graduates the exponential decay (and the resultant Lorentzian lineshape) due to T^*_2 relaxation, the second exponential function induces apodization of the end of the FID signal, but also a Gaussian lineshape of the spectrum. For $T_L = T^*_2$ the Lorentzian lineshape disappears completely and is replaced by a fully Gaussian lineshape for a fairly long T_G. Gaussian lineshapes are steeper and thus improve the resolution in the frequency-domain.

FIDs, theoretically, are continuous, analog signals. However, for practical reasons, they are transformed to discrete signals by an analog-to-digital converter (ADC), which in a signal of N points over an acquisition time, T_{acq}. Due to the use of discrete Fourier transformation, the spectrum will also contain N points with a spectral resolution of $1/T_{acq}$. One way to improve the spectral resolution is to increase the acquisition time. But, as mentioned above, for higher t, the SNR decreases rapidly and the additional achieved signal would mostly contain noise. By simply filling an FID with zeros rather than actually measuring for a longer time, the SNR can be improved significantly while also improving the spectral resolution.

1.1.3 Basics of MR Imaging

Magnetic resonance imaging (MRI) has proven to be a powerful but non-invasive and nondestructive technique with which to generate images of living objects. These images obtain the spatial boundaries and distribution of different tissues within the object, which must be known before using any localization technique for in vivo MRS. Furthermore, many MRS techniques apply the same principles underlying MRI, thus, this subsection will discuss MRI principles.

Magnetic Field Gradients

Rather than spectral information, the resonance condition of equation (1.8) is used to provide spatial information by using magnetic field gradients. While the magnetic field is always directed in one direction, the gradients point in the direction in which the strength of the magnetic field changes. Magnetic field gradients can be used to make the external magnetic field position-dependent by:

$$B(\mathbf{r}) = B_0 + \mathbf{r}\mathbf{G} \tag{1.24}$$

where $B(\mathbf{r})$ is the total magnetic field at position $\mathbf{r}$ and $\mathbf{G}$ represents the magnetic field gradient. Consequently, the resonance condition must be modified and the resonance frequency becomes dependent on position $\mathbf{r}$:

$$\omega(\mathbf{r}) = \gamma B(\mathbf{r}) = \gamma B_0 + \gamma \mathbf{r}\mathbf{G} \tag{1.25}$$

Using magnetic field gradients, all three dimensions can be encoded.

Slice Selection

To avoid a considerable expenditure of time due to independently encoding all three dimensions, 3D imaging is reduced to several 2D images which results in the desired 3D image. Such a 2D slice is selected by simultaneously combining an RF pulse, which excites only a certain frequency range, with a magnetic field gradient in one dimension, e.g. z. The resulting thickness of the slice, Δz, is specified by:

$$\Delta z = \frac{\Delta \omega}{\gamma G_z} \tag{1.26}$$

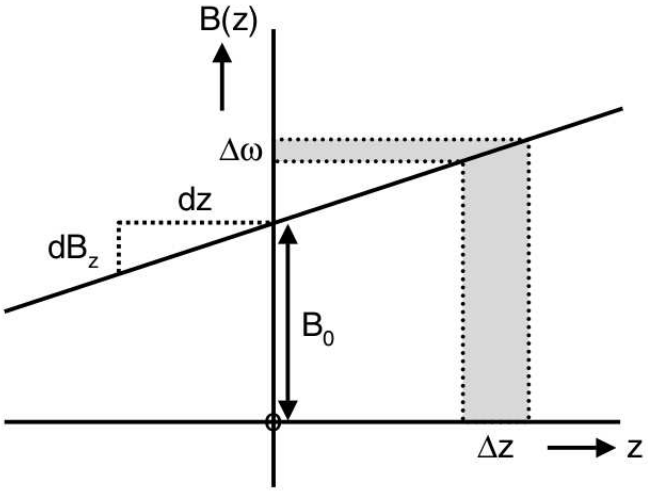

Figure 1.5: The relation between slice thickness, bandwidth, and the slope of the z-component of the magnetic field gradient [1]

.

with the bandwidth of the RF pulse, $\Delta\omega$, and the magnetic field gradient in the z-direction, G_z.

Frequency Encoding

Frequency encoding provides a favorable method for imaging along one of the two dimensions of the selected slice. The idea is to make the signal frequency position dependent on a magnetic field gradient before and during acquisition, thus generating a 1D projection. First, an encoding gradient, G_1, is applied, generating a specific phase shift of the transverse magnetization dependent on position $\mathbf{r}$:

$$\phi_1(\mathbf{r}, t) = \gamma \mathbf{r} \int_0^t \mathbf{G}_1(t')dt' \tag{1.27}$$

Consequently, due to the position-dependent phase shift, spins rotate differently at each position and the frequencies, $\omega(\mathbf{r})$, rely on the phase shift according to:

$$\omega(\mathbf{r}) = \frac{d\phi(\mathbf{r}, t)}{dt} \qquad \text{with} \qquad \phi(\mathbf{r}, t) = \phi_1(\mathbf{r}, t_1) + \gamma \mathbf{r} \int_0^t \mathbf{G}_2(t')dt' \tag{1.28}$$

where t_1 is the length of $\mathbf{G}_1$ and $\mathbf{G}_2$ denotes the second so-called readout gradient which has the opposite sign and double the amplitude of the

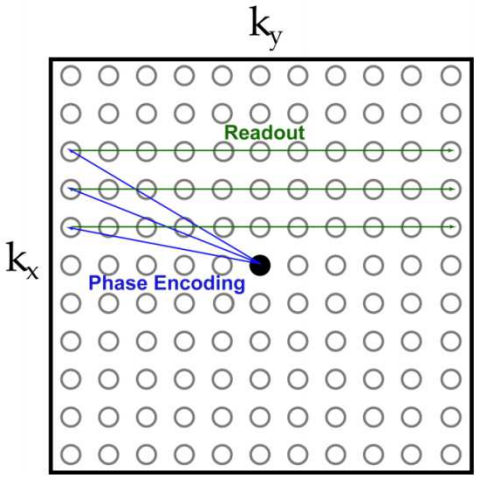

Figure 1.6: The construction of the k-space image slice by means of phase encoding and frequency encoding (readout) [8].

first one. During $\mathbf{G}_2$ the signal is acquired and Fourier transformation obtains the spatial spin distribution.

Phase Encoding

To obtain the second spatial coordinate of the slice, phase encoding is performed: the phase of the signal is encoded as a function of $\mathbf{r}$ prior to the signal acquisition. This is accomplished by repeating the same frequency-encoding procedure with appropriately modified phases. The frequency encoding can be described as the readout of a row, and the phase encoding as the selection of the row position.

The image projection, which is generated by frequency and phase encoding, is the built-in spatial-frequency domain and is called k-space. The image itself is then 'reconstructed' via 2D Fourier transformation.

k-Space

The observed transverse magnetization, $M_{xy}(t)$, in the context of a magnetic field gradient dependent on time, $\mathbf{G}(t)$, can be described as:

$$M_{xy}(\mathbf{G},t) = \int_{-\infty}^{+\infty} M_0(\mathbf{r}) e^{+i\gamma\mathbf{r}\int_0^t \mathbf{G}(t')dt'} d\mathbf{r} \tag{1.29}$$

with $M_0(\mathbf{r})$ denoting the spin density at position $\mathbf{r}$ and t the time after a 90° pulse. Let $\mathbf{k}(t)$,

$$\mathbf{k}(t) = \gamma \int_0^t \mathbf{G}(t')dt' \tag{1.30}$$

define the spatial frequency variable. Then equation (1.29) reduces to:

$$M_{xy}(\mathbf{k}(t)) = \int_{-\infty}^{+\infty} M_0(\mathbf{r}) e^{+i\mathbf{k}(t)\mathbf{r}} d\mathbf{r} \tag{1.31}$$

The achieved equation (1.31) illustrates that the inverse Fourier transformation of the spin density equals the time-domain signal that generates the 2D k-space of the spatial frequencies. Consequently, the dimensions of the k-space correspond to the dimensions of the image. However, every pixel in the k-space maps to the whole image and every pixel in the image maps to the whole k-space.

1.1.4 Basics of MR Spectroscopic Imaging

The principles of MRSI are very similar to the above-discussed phase encoding of MRI. In particular, the basic pulse sequences are analogous to the MRI spin echo sequences. The main difference between MRSI and MRI is the additional axis for the chemical shift dispersions: after selecting a slice, for both transverse dimensions phase encoding is performed and for each point in the k-space an FID signal is acquired. A (spatial) Fourier transformation with respect to the applied phase-encoding gradient will obtain the spatial distribution; Fourier transformation of the localized time-domain signal will then reveal the frequency-domain spectrum. Considering the x-direction, the total acquired time-domain signal, $S(t)$, equals the sum of the time-domain signal from all elementary volume elements, $s(x,t)dx$, from each point x in the sample:

$$S(t) = \int_{-\infty}^{+\infty} s(x,t)dx \tag{1.32}$$

Applying the k-space formalism and Fourier transformation, the total spectrum of the sample, $F(\omega)$, obtained by:

$$F(\omega) = \int_{-\infty}^{+\infty} S(t)e^{-i\omega t}dt = \int_{-\infty}^{+\infty} f(x,\omega)dx \tag{1.33}$$

To obtain the spatial distribution, $f(x,\omega)$, a phase-encoding gradient, G_x, is applied that includes a phase shift on each elementary volume element. The entire spectrum can then be described by:

$$F(G_x,\omega) = \int_{-\infty}^{+\infty} f(x,\omega)e^{i\gamma x G_x t}dx \tag{1.34}$$

Using the k-space formalism, i.e., $k_x = \gamma G_x t$, equation (1.34) becomes:

$$F(k_x,\omega) = \int_{-\infty}^{+\infty} f(x,\omega)e^{ik_x x}dx \tag{1.35}$$

The phase-modulated spectra of the whole sample, $F(k_x,\omega)$, represents the inverse Fourier transformation of the spatially distributed spectra, $f(x,\omega)$, from the individual volume elements. $f(x,\omega)$ can hence be revealed by Fourier transformation of $F(k_x,\omega)$:

$$f(x,\omega) = \int_{-\infty}^{+\infty} F(k_x,\omega)e^{-ik_x x}dk_x \tag{1.36}$$

This illustrate only one of the three spatial dimensions, but it can easily be extended to all of them by independently and subsequently applying three orthogonal gradients. The volume elements are then calculated as:

$$f(x,y,z,\omega) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(k_x,k_y,k_z,\omega)e^{-i(k_x x+k_y y+k_z z)}dk_y dk_y dk_z \tag{1.37}$$

Although written as continuous infinite integrals, in praxis, the k-space is sampled at discrete positions over finite time points and the corresponding spectra are obtained by a discrete Fourier transformation.

1.2 Metabolite Quantification in MR Spectroscopy: LCModel

The main intention of MR spectroscopy is to determine the concentrations of the molecules under observation. The concentration of a molecule is directly proportional to the relative integral of its spectrum in the frequency domain. Thus, a molecule's concentration can be quantified by determining its spectrum. In praxis, the acquired spectrum consists of the individual metabolic spectra, the spectral baseline, the macromolecule baseline, and the spectral noise. Therefore, the acquired spectrum must be noise-corrected, the baseline removed, and the metabolic spectra disassembled. It is essential to obtain quantification results that are accurate and reliable. To obtain the disassembled, noise-corrected, and baseline-free spectra, there are several methods and algorithms. For brain MRSI, a broadly used commercial software is LCModel. In this chapter, the basic concept of LCModel and its limitations, as well as its disadvantages, will be described.

1.2.1 Basic Concept

The basic concept and origin of the name 'LCModel' is to analyze a spectrum as a linear combination of a basis set of complete model spectra of metabolite solutions. LCModel uses a parametric, constrained regularization method, which attempts to choose the best compromise between bias and artifacts. That can lead to instabilities due to finding the smoothest lineshape and baseline consistent with the data. The only necessary input of LCModel is the time-domain data; no user interaction is required apart from setting several (admittedly optional) starting-parameters and soft constraints.

The in vivo spectra, as well as the N_M *in vitro* model spectra in the basis set, are first zero-filled in the time-domain. The time domain *in vitro* spectra, $m_l(t)$, for time t and $1 \leq l \leq N_M$, correspond to their frequency-domain counterparts

$$M_l(\nu; 0, 0) = FFT(m_l(t)).$$

To account for the differences between the *in vivo* and *in vitro* spectra, the basis sets are modified. First, for T_2 broadening and due to possi-

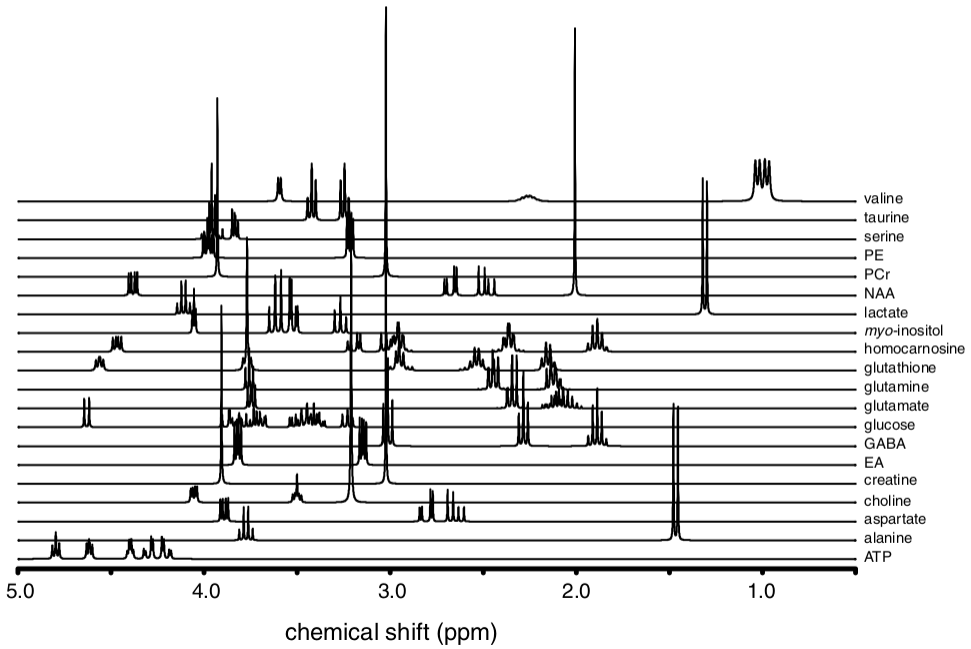

Figure 1.7: The basis set used by LCModel with the names of the metabolites on the y-axis and the chemical shift on the x-axis [2].

ble referencing errors, the broadening parameters, γ_l, and the shifting parameters, ϵ_l, are applied in the time-domain.

$$M_l(\nu;\gamma_l,\epsilon_l) = \mathrm{FFT}(m_l(t)\exp(-(\gamma_l + i\epsilon_l)t)) \tag{1.38}$$

where FFT denotes the fast discrete Fourier transform. For the whole basis set, a convolution with vector $\boldsymbol{S}$ of $2N_S$ lineshape parameters S_n corresponding to field inhomogeneities, eddy currents, and frequency instabilities is performed; to achieve the different possible concentrations, the model spectra are then multiplied by the respective concentration parameter, C_l.

$$\sum_{l=1}^{N_M} C_l \sum_{n=-N_S}^{N_S} S_n M_l(\nu_{k-n};\gamma_l,\epsilon_l)$$

Due to the appearance of macromolecules in the *in vivo* spectra, a baseline must be added. This baseline is represented as a basis set of N_B cubic B-splines, $B_j(\nu)$. Zero- and first-order phase corrections are achieved by the respective parameters, ϕ_0 and ϕ_1. Thus, the discrete *in vivo* frequency-domain spectrum data points, $Y(\nu_k)$, are represented by LCModel

as:

$$\hat{Y}(\nu_k) = \exp[-i(\phi_0+\nu_k\phi_k)][\sum_{j=1}^{N_B}\beta_j B_j(\nu_k)+\sum_{l=1}^{N_M} C_l \sum_{n=-N_S}^{N_S} S_n M_l(\nu_{k-n};\gamma_l,\epsilon_l)] \tag{1.39}$$

With the constraints

$$C_l \geq 0, \quad \gamma_l \geq 0 \quad \text{and} \quad \sum_{n=-N_S}^{N_S} S_n = 1. \tag{1.40}$$

To estimate the metabolite concentrations and their uncertainties, a constrained, nonlinear, least squares analysis is performed to minimize the criterion

$$\begin{aligned}\frac{1}{\sigma^2(Y)}\sum_{k=1}^{N}(Re(Y(\nu_k)-\hat{Y}(\nu_k)))^2 + \|\alpha_S R_S \mathbf{S}\|^2 + \|\alpha_B R_B \boldsymbol{\beta}\|^2 \\ + \sum_{l=1}^{N_M}\left(\frac{(\gamma_l-\gamma_l^0)^2}{\sigma^2(\gamma_l)} + \frac{\epsilon_l^2}{\sigma^2(\epsilon_l)}\right) = \text{minimum},\end{aligned} \tag{1.41}$$

where the first term corresponds to the common least squares criterion of the real parts of both the *in vivo* and modeled spectra. The second and third terms are the regularization terms for the lineshape parameters and the β_j of the cubic B-splines, denoted as vector $\boldsymbol{\beta}$, respectively, with R_S, R_B being the particular regularizer matrices. The last sum of the criterion constitutes the prior normal probability distributions of the broadening and shifting parameters, γ_l and ϵ_l.

1.2.2 Limitations and Problems

As do most of the state-of-the-art approaches for quantification, LCModel also uses a nonlinear least squares fitting method. Since all of these nonlinear least squares analyses are iterative processes and one whole brain scan usually contains several thousands of spectra, all these analyses are computationally intensive and, depending on the volume size and resolution, may take several hours.

Another problem is the software packages themselves. Bhogal et al. compared four research institutions; each obtained the exact *in vivo* spectra of 30 individuals and had to process these with LCModel. All four institutions returned different results due to individual preferential ini-

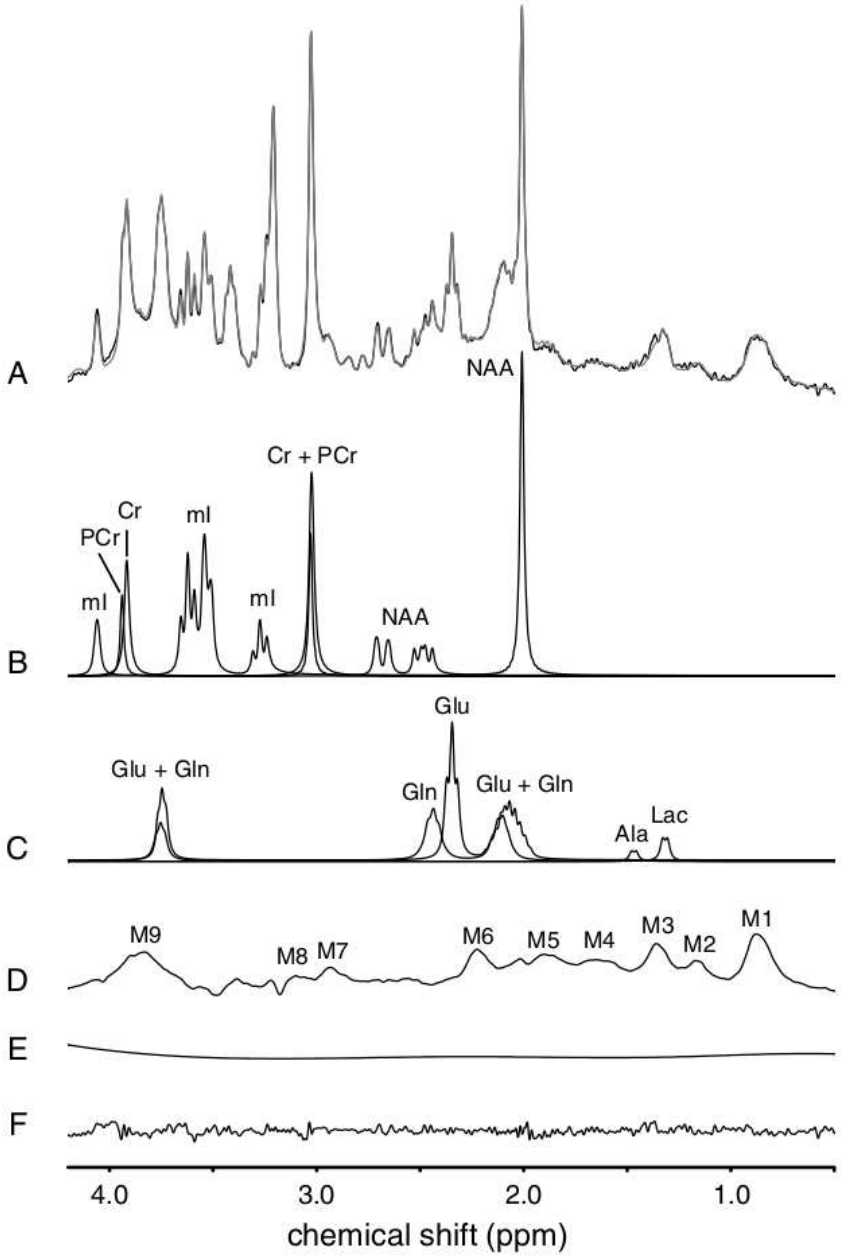

Figure 1.8: The different compounds of a spectrum disassembled by LCModel: (A) Original (gray) and fitted (black) spectra. (B)+(C) Individual metabolite resonances. (D) Reconstructed macromolecule baseline. (E) Reconstructed spectral baseline. (F) Spectral noise [1].

tialization parameters as well as optimized standards and constraints [9].

1.3 Introduction to Deep Learning

This section is a broad introduction to deep learning and its tools as presented in "*Deep Learning*" by Goodfellow et al. [3].

1.3.1 Motivation

In the past few years, machine learning has had an enormous impact on science and technology since computers are able to handle the huge amount of data required for it. Deep learning, in particular, as possibly most prominent part of the machine learning family, has been applied broadly, e.g., computer vision, speech recognition, and medical image analysis. Some of the first medical applications for deep learning appeared in the early 1990s, e.g., lung nodule detection [10], micro calcification mammography [11], or classification of breast tissue [12].

The basic concept of deep learning involves deep neural networks, artificial neural networks inspired by neuroscience. The first artificial neuron was described by Warren McCulloch and Walter Pitts in 1943. They created a mathematical model for a single biological neuron [13]. In 1958, Frank Rosenblatt illustrated a similar model, but focused on its application, such as pattern recognition, and called it a perceptron [14].

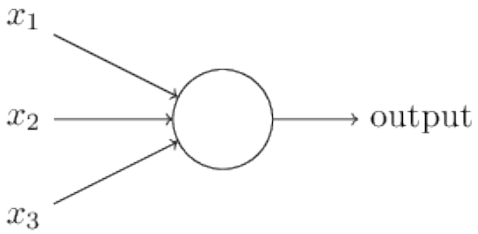

Figure 1.9: Simple artificial neuron with three inputs, x_1, x_2, and x_3. The processing of the input occurs in the center, resulting in the output [15].

An artificial neuron or perceptron consists of its inputs, x_i, the corresponding weights, w_i, the activation function, ϕ, and the output, y, of the neuron:

$$y = \phi(\mathbf{w}^T\mathbf{x}) = \phi\left(\sum_i w_i x_i\right),$$

with $\mathbf{w}$, $\mathbf{x}$ the vectors of the weights and inputs, respectively. The activation function, ϕ, describes whether the neuron is activated or not (which

usually equals 0) and is often a nonlinear threshold function. Rosenblatt's perceptron used this structure with the heavyside function as the activation function, and created a binary classifier. This perceptron is the simplest neural network and approximates a target function that discriminates between two classes based only on the input values. This illustrates the basic idea; but, more complex and nonlinear problems need more complex structures.

1.3.2 Feedforward Neural Networks

Deep neural networks are designed to approximate some target function $f^* : \mathbf{X} \to \mathbf{Y}$, $f^*(\mathbf{x}) = \mathbf{y}$ by a mapping $f(\mathbf{x}) = f(\mathbf{x}; \mathbf{\Theta}) = \hat{\mathbf{y}}$ with inputs $\mathbf{x}$ in the input data set $\mathbf{X}$, the desired output $\mathbf{y} \in \mathbf{Y}$, and function parameters $\mathbf{\Theta}$. $\mathbf{X}$ and $\mathbf{Y}$ hereby form the training set $\mathcal{T} = \mathbf{X}, \mathbf{Y}$. To get as close as possible, the parameters are optimized by training the network with training data. This optimization or learning depends on the network's output $\hat{\mathbf{y}}$ and the target function's output $\mathbf{y}$. The learning is classified as supervised and unsupervised learning. Supervised learning requires $\mathbf{y}$ to be targets or labels associated with the input $\mathbf{x}$. This association is made prior and the network is trained to clearly match the input with the corresponding label or target, e.g., like a classifier. Unsupervised learning requires less prior knowledge and tries to learn unknown properties of the input data set to estimate its probability distribution explicitly or implicitly to perform synthesis or denoising of the input. To quantify the disparity of the network f and the desired function f^*, an overall loss function, $J(\mathbf{X}; \mathbf{\Theta})$, over the whole training must be defined. Typically, the loss, J, relies on a per-example loss, $L(f(\mathbf{x}; \mathbf{\Theta}), \mathbf{y})$, and decomposes as a sum of those losses:

$$J(\mathbf{X}; \mathbf{\Theta}) = \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y} \in \mathcal{T}} L(f(\mathbf{x}; \mathbf{\Theta}), \mathbf{y}) = \frac{1}{N} \sum_{\mathbf{x},\mathbf{y} \in \mathcal{T}} L(f(\mathbf{x}; \mathbf{\Theta}), \mathbf{y}),$$

with N being the number of elements in $\mathcal{T}$. The loss, $J(\mathbf{\Theta}) = J(\mathbf{X}; \mathbf{\Theta})$, needs to be minimized with respect to $\mathbf{\Theta}$ and an optimal approximation, f, of f^* is determined by the training criterion

$$\mathbf{\Theta}^* = \arg\min_{\mathbf{\Theta}} J(\mathbf{\Theta}).$$

Deep neural networks differ according to their flow of information. In

feedforward neural networks (FNN), information always moves in one direction from input to output like a directed acyclic graph, whereas in recurrent neural networks, loops are possible and information may flow in both directions. Since the architecture used in this thesis is a feedforward neural network, attention is paid to this kind of structure.

While a perceptron consists of only one function, FNNs are a chain of two or more functions called layers. The layers between the input and the output layer are referred to as hidden layers and their number corresponds to the depth of a network. Given three layers f^1, f^2 and f^3 and input, $\mathbf{x}$, the FNN is a function $f(\mathbf{x}) = f^3(f^2(f^1(\mathbf{x})))$. Each layer f^i typically has its individual parameters, $\mathbf{\Theta_i}$, describing an affine transformation function, K, on the layer's input $\mathbf{x}_i$ and, in addition to this an activation function, ϕ_i, resulting in

$$f^i(\mathbf{x}_i) = f^i(\mathbf{x}_i; \Theta_i; \phi_i) = \phi_i(K(\mathbf{x}_i; \mathbf{\Theta}_i)).$$

The individual output of each layer is usually a vector and each element of this vector is called a unit or a node and is the result of a vector-to-scalar function operating in parallel. The transformation function, $K(\mathbf{x}_i; \mathbf{\Theta_i})$, is referred to as the kernel (function) of the layer. The basic kernel used in FNNs is a simple matrix multiplication with weights for matrix, $\mathbf{W}$, and bias, $\mathbf{b}$. For such a kernel, $K(\mathbf{x}_i; \mathbf{W}_i, \mathbf{b}_i) = \mathbf{W}_i^T\mathbf{x}_i + \mathbf{b}_i$, the layer

$$f^i(\mathbf{x}_i) = \phi_i(\mathbf{W}_i^T\mathbf{x}_i + \mathbf{b}_i),$$

is called a fully connected layer (FCL).

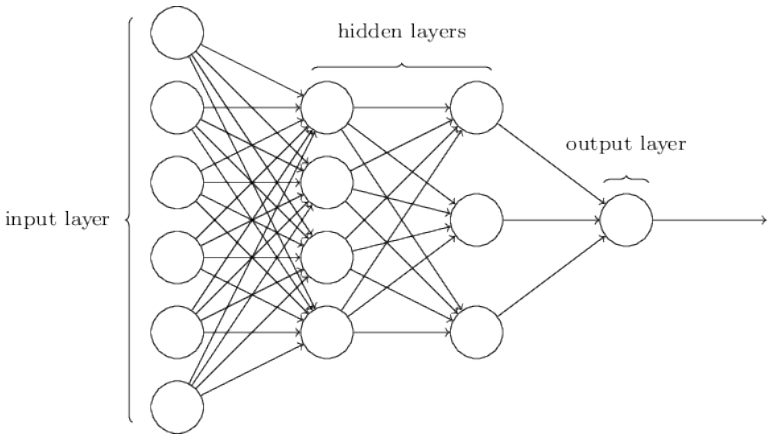

Figure 1.10: Typical architecture of a simple feedforward neural network with an input layer, two hidden layers and one output [15].

Given an FNN of the structure of Figure 1.10, the network's function becomes

$$\hat{y} = f(\mathbf{x}) = f^3(f^2(f^1(\mathbf{x}))) = \phi_3(\mathbf{W}_3^T\phi_2(\mathbf{W}_2^T\phi_1(\mathbf{W}_1^T\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3)$$

approximating the target function $f^*(\mathbf{x}) = y$. All three layers are FCLs.

Commonly used activation functions are threshold functions. The most prominent one is probibly the

$$\text{rectifier:} \qquad a(x) = \max\{x, 0\},$$

and its modification

$$\text{leaky rectifier:} \qquad a(x) = \max\{x, 0\} + \min\{\alpha x, 0\},$$

with $\alpha > 0$ as the slope of negative values. Units with the rectifier or leaky rectifier as their activation function are called rectified linear units (ReLU) and leaky rectified linear units (leakyReLU) respectively. Other very often used activation functions are the logistic function, the hyperbolic tangent and many more.

1.3.3 Learning and Backpropagation

When an FNN f with parameters $\boldsymbol{\Theta}$ is fed with an input $\mathbf{x}$, information flows forward to the output layer computing every layer's output one after the other, which results in a final output, $\hat{\mathbf{y}}$, of the network and a loss, $J(\boldsymbol{\Theta})$. This procedure from the input $\mathbf{x}$ to the loss $J(\boldsymbol{\Theta})$ is called forward-propagation. During training, when the parameters, $\boldsymbol{\Theta}$, are optimized, a forward-propagation is performed to calculate the loss, $J(\boldsymbol{\Theta})$. To know how to manipulate $\boldsymbol{\Theta}$, the information of the loss flows backward through the layers to compute the gradient of the whole network, f. The gradient of a network is the transposition of the first derivative of the loss function, J, with respect to $\boldsymbol{\Theta}$ for a given input, $\mathbf{x}$. The negative gradient $-\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta})$ characterizes the direction of largest decrease of J with respect to $\boldsymbol{\Theta}$. This so-called backpropagation or backprop is the fundamental step of training the model. This backprop that uses a learning technique which relies on the network's gradient is therefore called gradient-based learning.

Gradient Descent Optimization

Gradient Descent Optimization is an iterative learning algorithm that updates the network's parameters at each epoch of iteration until the criterion or loss, $J(\boldsymbol{\Theta})$, converges, or a maximum of epochs is reached. By knowing the gradient, $\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta})$, following its direction to the minimum of J will yield an optimal $\boldsymbol{\Theta}^*$. This principle is used by gradient descent optimization, an iterative optimization process that forms the basis for nearly all optimization algorithms used in deep learning. Let $\boldsymbol{\Theta}(t)$ be the state of the parameters, $\boldsymbol{\Theta}$, at time point t. Then, at each time step, $t+1$, the parameters are updated as follows and

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) + \Delta\boldsymbol{\Theta}(t) = \boldsymbol{\Theta}(t) - \gamma \mathbf{g}$$

denotes the learning rule of the gradient descent optimization algorithm with the scalar $\gamma > 0$ the learning rate and

$$\mathbf{g} = \nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}(t))$$

denotes the gradient.

In praxis, training sets can be extremely huge, and thus, the parameter update of one time step happens to be computationally expensive. Therefore, at each epoch, only a random minibatch, $\mathcal{B} \subset \mathcal{T}$, of the training set is used. This variation is called stochastic gradient descent optimization and is the most popular optimization used in deep learning. The learning rule becomes then

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) - \gamma \hat{\mathbf{g}},$$

with

$$\hat{\mathbf{g}} = \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y} \in \mathcal{B}} L(f(\mathbf{x}; \boldsymbol{\Theta}), \mathbf{y})$$

denoting the estimated gradient.

The standard stochastic gradient descent learning corresponds only to the gradient of the last epoch. Therefore it may happen that the learning oscillates and thus converges slowly. Including the gradients of the past, the stochastic gradient descent with momentum implements a velocity term called 'momentum' to update the parameters. The learning rule is given by

$$\boldsymbol{\Theta}(t+1) = \boldsymbol{\Theta}(t) + \mathbf{v}(t+1),$$

with

$$\mathbf{v}(t+1) = \alpha \mathbf{v}(t) - \gamma \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}\in\mathcal{B}} L(f(\mathbf{x};\boldsymbol{\Theta}),\mathbf{y})$$

being the momentum at epoch $t+1$, with hyperparameter $\alpha \in [0,1)$ determining the influence of the previous gradients.

Backprop Algorithm

The analytical term of the gradient $\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta})$ may be mathematically straightforward; its numerical computation, however, is computationally quite expensive. The backprop algorithm provides a rather inexpensive solution.

The basic idea of brackprop is to calculate the partial derivatives of $J(\boldsymbol{\Theta})$ with respect to every parameter in $\boldsymbol{\Theta}$ by applying the chain rule for derivatives of composite functions recursively, beginning at the final layer. Revisiting the FNN of Figure 1.10, for simplification, the bias may be assumed to be zero and all activation functions to be the same ϕ. Let $\mathbf{x}$ again be the input vector, $\mathbf{h}^1$ and $\mathbf{h}^2$ be the output vectors of the hidden layers before applying the activation function, ϕ, and $\hat{y}$ the output of the last layer. Let w_k denote the weights of the last layer; then, the partial derivates with respect to w_k are calculated as

$$\frac{\partial L}{\partial w_k} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_k}.$$

Next, the partial derivatives with respect to the weights, w_{kj}, from h_j^1 to h_k^2 are calculated as

$$\frac{\partial L}{\partial w_{kj}} = \frac{\partial L}{\partial \phi(h_k^2)} \frac{\partial \phi(h_k^2)}{\partial h_k^2} \frac{\partial h_k^2}{\partial w_{kj}} = \frac{\partial L}{\partial \hat{y}} w_k \frac{\partial \phi(h_k^2)}{\partial h_k^2} \frac{\partial h_k^2}{\partial w_{kj}}.$$

Analogously the partial derivatives with respect to the weights, w_{ji}, from x_i to h_j^1 are calculated. This illustrates how the backprop works for FCLs.

1.3.4 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a special class of FNNs. CNNs are used for any data with a known topology, such as images or time series data. In general, a CNN is a regular FNN with at least one convolutional layer.

Convolution Operation

Similar to fully connected layers, a convolutional layer consists of an affine transformation followed by an activation function. In contrast to a fully connected layer, the affine transformation function, K, of a convolutional layer is the eponymous mathematical operation convolution

$$s(t) = (g * h)(t) = \int g(\tau)h(t-\tau)d\tau$$

of two functions g and h. Since data is always discrete, the discrete convolution is the proper operation:

$$s(t) = (g * h)(t) = \sum_{\tau=-\infty}^{\infty} g(\tau)h(t-\tau).$$

The input data, I, and the kernel, K, of convolutional layers are usually one- or multi-dimensional tensors with certain entries stored in every point of the tensor. I and K can be seen as a finite and convex subset in reference to the infinite d-dimensional space. Thus, I and K can be assumed to be zero everywhere but in the points stored with values. Thus, it appears that the infinite sum of the discrete convolution can be expressed as a sum over a finite number of elements. Let I and K be two-dimensional, then

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

denotes the discrete 2D-convolution over input I and kernel K.

Convolutional Layer

The kernel K in convolutional layers is mostly referred to as the filter and the output of the convolution as the feature map. Filters used in CNNs usually are much smaller than the input, I. Due to this, for every node in the input, there is only one region of the filter's shape that is affected by that neuron. The layer obtains sparse local connectivity. This means that local patterns or features in I are enforced and only appear in the same region of the output. Thus, the output is also called the feature map. Another property of smaller filters is parameter-sharing, since only one tensor is used for all nodes in I. A great benefit of this is the decrease

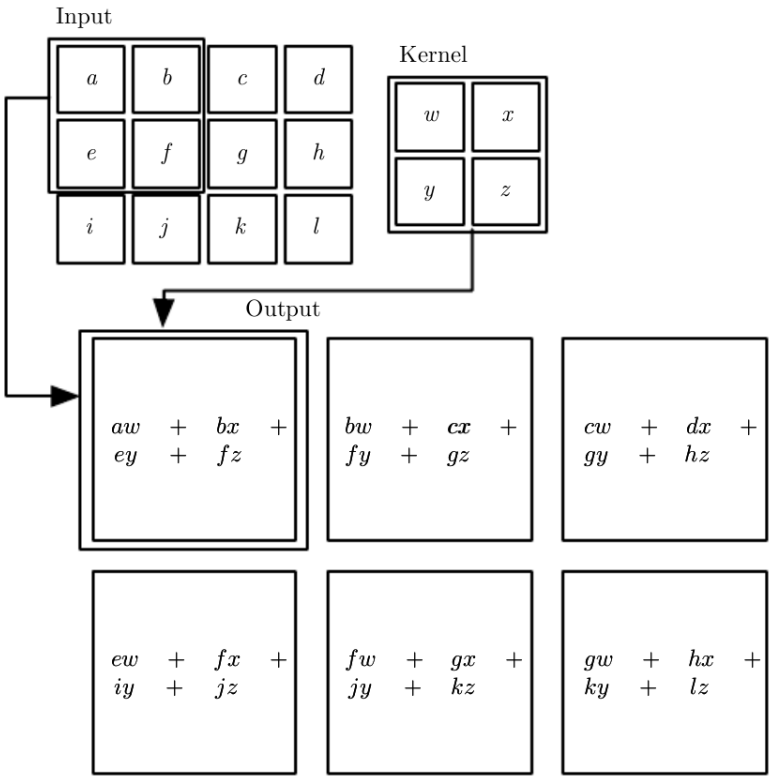

Figure 1.11: Example of 2-D convolution of a 3×4 input and a 2×2 kernel resulting in a 2×3 feature map [3].

of memory needed to store the filter's values compared to fully connected layers. Furthermore, a mathematical consequence of parameter-sharing is translation equivariance; for a layer, f^i, its input $\mathbf{x}_i$ and a translation function, T, then holds the expression $f^i(T(\mathbf{x}_i)) = T(f^i(\mathbf{x}_i))$.

The shape of the filter used in a convolutional layer is an important hyperparameter and affects the shape of the feature map. It further defines the subspace of the input, which defines a single point in the feature map. This subspace is known as the receptive field of a unit. Another hyperparameter that influences the size of the feature map is the step size of the filters, i.e., the number of points the filter moves in the input at each step of the convolution. This hyperparameter is called stride. Yet another hyperparameter to be chosen is zero-padding or simply padding. Zero-padding of the input adds zeros at the borders of the input volume to prevent the shape of the input and to balance the loss of informational influence of the values at the border. The padding parameter typically is logical, deciding whether or not zeros should be added to prevent the original shape (or ratio) of the input.

In deep learning praxis, usually more than one filter per layer is used

to gain more features of the input. This results in a new dimension called channels, with one channel for each filter and feature map. So the output consists of several feature maps that form a feature space. In conclusion, all hyperparameters' filter shape, number of filters, stride and padding define the shape of the feature space. Let I be an input tensor of shape $M \times N \times C$, with C channels and $M \times N$ data. Further, let $K \times L$ be the shape of the F filters in a convolutional layer with stride S. Then

$$\frac{M-K}{S}+1 \times \frac{N-L}{S}+1 \times C \cdot F$$

defines the size of the feature space without zero-padding and

$$\frac{M}{S} \times \frac{N}{S} \times C \cdot F$$

defines the size of the feature space with zero-padding. A stride of 1 preserves the data shape; a stride of 2, for instance halves each dimension. The process of decreasing the dimensions of the input is called downsampling and is an often-used property of convolutional layers. It also is a property and aim of a layer commonly used subsequently to a convolutional layer: pooling.

Pooling

The pooling layer is used right after a convolutional layer to further modify its outcome. The pooling layer performs a summary statistic of the neighborhood of a node and replaces it with the statistic's outcome. Commonly used pooling functions involve the maximum of a neighborhood (called max-pool), its average, the L^2 norm, or a weighted average depending on the distance from the node in the center. Similar to convolutional layers, a pooling size that defines the rectangular neighborhood to be analyzed and a stride are hyperparameters to be set for a pooling layer. With strides greater than 1, downsampling can be achieved. Another property of pooling is translational invariance: small translational changes in the input do not change the pooling layer's outcome.

A convenient structure used in CNNs is the blockwise use of one or more convolutional layers that includes an activation function and a pooling layer at the end of the block. Figure 1.12 shows a typical CNN architecture with three convolutional layers, each followed by a max-pool layer.

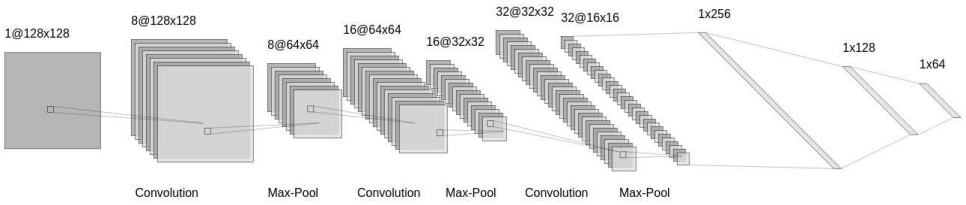

Figure 1.12: A typical CNN with two convolutional layers and two max-pool layers followed by three dense layers.

Backpropagation

The backpropagation of convolutional layers works basically as indicated above. A big advantage is due to parameter-sharing the relatively low number of partial derivatives to be computed for such a layer. As pooling layers do not use any trainable parameters, no updates must be performed here.

1.3.5 Autoencoders

Autoencoders are a special class of artificial neural networks that use unsupervised learning. The aim of autoencoders is to adopt a representation (so-called encoding) of the input data in a usually lower dimension and then reconstruct it in the original space (decoding). Autoencoders thus consist of an encoding function, $f(\mathbf{x})$, and a decoding function, $g(f(\mathbf{x}))$. Autoencoders are built to learn the properties of an input to synthesize it or to denoise a corrupted input. Therefore autoencoders are not about the output of the network itself, but rather about learning the properties and features that represent the input.

In most real-world cases, the decoding function, g, is a predefined function with little or no parameters to learn. It can be a model function or regression function using prior knowledge of the data. Hence, the decoding function is usually not the crucial part unlike the encoding function.

Because their hidden layers have smaller dimensions than the input, so-called undercomplete autoencoders use the principle of downsampling to delete useless features and keep those that have a larger impact on the decoding function.

1.3.6 Training, Initialization, and Regularization

Training

The goal in deep learning is to learn an approximation, $f(\mathbf{x})$, of the true function, $f^*(\mathbf{x})$, to fulfill the condition

$$f(\mathbf{x}) \approx f^*(\mathbf{x}), \quad \forall \mathbf{x} \in \mathbf{\Omega},$$

with $\mathbf{\Omega}$ being the population of all possible observations. In real life, it is neither possible to acquire all possible observations nor computationally feasible to train a network on such an infinitely large population set. Thus, deep learning attempts to learn on a finite subset, $\mathbf{X}^{(Train)} \subset \mathbf{\Omega}$, a training set, to achieve a generalization on the whole population $\mathbf{\Omega}$. A large enough $\mathbf{X}^{(Train)}$ must be chosen with independent and identically distributed $\mathbf{x}$ to obtain generalization.

Also, the network must be designed properly. A too-small number of trainable parameters may result in large training costs that cannot be minimized, and f^* cannot be approximated. This case is called underfitting. A too-large number, on the other hand, may result in a very small training cost. But, if new unknown input data is fed to the network, the cost might be very high, known as overfitting. The capacity the number of trainable parameters defined by the network's depth and width, determines whether the network can achieve generalization or not. The capacity also describes the hypothesis space, the space of all possible functions the network can adopt.

To quantify a network's performance on unknown data, another subset of $\mathbf{\Omega}$, the so-called test set, is needed. After the network is trained, the test set is fed forward to compute its test cost. During training, the training loss, $J(f(\mathbf{X}^{(Train)}), f^*(\mathbf{X}^{(Train)}))$, of the network is minimized; but the goal is to rather minimize the test loss, $J(f(\mathbf{X}^{(Test)}), f^*(\mathbf{X}^{(Test)}))$.

Regularization

One way to achieve generalization is to choose the perfect capacity. This is rather difficult, as assuming a reasonable range of capacity requires solid prior knowledge. Another way to improve the learning is to modify the learning algorithm. Those modifications are referred to as regularization.

An easy modification is the early-stopping method. A maximum number of epochs is scheduled with regard to when to stop the training. But, stopping at the optimal time point is more or less luck.

A common regularization is to use a third set, a validation set. All three sets must not overlap and have to be independent and identically distributed. During the training process, backpropagation and weight updates are performed based on the training set and training error. However, the validation error must be minimized. The learning algorithm, therefore, stops when the minimum of the validation error is accomplished assuming the generalization error to be minimized as well.

Yet another method forces the weights to decay by adding a regularization term to the loss. The regularization term depends on the network's parameters, $\mathbf{\Theta}$, and a decay parameter, λ. The training criterion becomes

$$J(\mathbf{\Theta}) = \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}\in\mathscr{T}} L(f(\mathbf{x};\mathbf{\Theta}),\mathbf{y}) + \lambda R(\mathbf{\Theta}),$$

with R being the regularization term. A broadly used regularization term is the L^2 norm of the parameters, $\mathbf{\Theta}$.

Initialization

When training is established, a parameter state at epoch zero must be initialized. This initialization influences if, how fast, and at what level the learning algorithm converges. An easy way is to set all parameters to a constant. But, since many layers are often designed similarly and units have the same activation function, equal parameters would result in equal unit outputs. Consequently, input as well as gradient patterns can get lost, making it difficult for the algorithm to learn. Thus, randomized initialization is generally recommended. One common initialization method is to assume the parameters to be uniformly distributed. The range of the uniform distribution depends on the number of inputs and outputs of a layer. Another initialization method depends on the Gaussian distribution. While, in the majority of cases, the mean is set to zero, the standard deviation depends on the input and/or output of a layer. For example, the Glorot normal initializer assumes, for the weights of an FCL, a Gaussian distribution with a default mean of zero and a standard deviation

$$SD = \sqrt{\frac{2}{M_{in} + M_{out}}},$$

with M_{in}, M_{out} being the number of input and output units of the layer [16]. The underlying Gaussian distribution i,s in praxis, often truncated to avoid extreme outliers. To conclude, random parameter initialization is recommended. However, one explicit way of initialization does not exist.

2 Methods

2.1 Data Simulations

All spectra were simulated using self-made MATLAB scripts [17]. The spectra consisted of a metabolite part and a baseline part. While the baseline part was provided from formerly processed *in vivo* data, the metabolite spectra were generated using a basis set of the single metabolite spectra. The metabolites used in the simulation were: NAA; NAAG; PC; GPC; Cr; PCr; m-Ins; Glu; and Gln. The basis set was simulated using jMRUI [18] without apodization with fixed T2 values for each metabolite. After the metabolite part was simulated, a Gaussian noise with random SNR was added as well as the baseline part.

2.1.1 Simulation of Metabolic Spectra

The foundation of the metabolic part of the spectra is the basis set. The basis set was derived from further studies on the 3T whole-body MR scanner (Prisma-fit, Siemens Healthcare, Erlangen, Germany) with a 32-channel head coil. The initial FID signals and the time vector had a length of 637 points and were zero-filled to a total size of 4,096 points. Each basis FID m_i was then normalized by dividing by the maximum magnitude of the summed basis set. The normalized basis set was saved and had a size of $9 \times 4,096$ for the nine metabolites.

To generate a total of $N = 75,000$ spectra, a metabolite concentration matrix, $\mathbf{C}$ of size $N \times 9$, was built containing random metabolite specific concentration factors following a uniform distribution, with ranges of the minimum and maximum concentrations factors of the respective metabolite.

First, by multiplying $\mathbf{C}$ with the basis set, a matrix of $N = 75,000$ raw FIDs of length 4,096 was created. Second, for the spectral lineshape, a mixture of Lorentzian and Gaussian broadening (called Voigt

Metabolite Concentration Ranges (in mmol L^{-1})		
Metabolite	Lower Bound	Upper Bound
Cr	4.5	10.5
GPC	0.25	1.0
Gln	3.0	6.0
Glu	6.0	12.5
m-Ins	4.0	9.0
NAA	7.5	17
NAAG	0.5	2.5
PC	0.5	2.0
PCr	3.0	5.5

Table 2.1: Metabolite concentration ranges according to De Graaf [1].

broadening) was applied in the time domain:

$$\mathrm{FID}_{\mathrm{Voigt}} = \mathrm{FID}_{\mathrm{raw}} \odot \exp\left(\frac{-\mathbf{t}}{L}\right) \odot \exp\left(\frac{-\mathbf{t}^2}{G^2}\right),$$

where $\mathbf{t}$ refers to the time vector. $L \in [0.05, 0.25]$ and $G \in (0, 1]$ are the Lorentzian and Gaussian parameters respectively. For each raw FID, random values for L and G were sampled from the uniform distribution in the particular ranges. The operator $\odot$ represents the Hadamard or Schur product, i.e., the element-wise multiplication of two vectors or matrices of equal shape. To obtain realistic shapes, only those spectra were kept with a linewidth of ∼5 to ∼12.3 Hz.

Third, the signals were Fourier transformed:

$$S_{\mathrm{Voigt}} = \mathrm{FFT}(\mathrm{FID}_{\mathrm{Voigt}})$$

and the zero-frequency component was shifted to the center of the array. Fourth, zero-order, and first-order phase shifts were performed by applying

$$S_{\mathrm{raw}} = |S_{\mathrm{Voigt}}| \odot \exp(i(\phi(S_{\mathrm{Voigt}}) + (\phi_0 + \phi_1 \mathbf{t}))),$$

with $\phi(S_{\mathrm{Voigt}})$ denoting the angle of S_{Voigt} ϕ_0 and ϕ_1 being the zero-order and first-order phase shift. Both were sampled from a uniform distribution with a range of$[-\pi, \pi]$. Normalizing the resulting spectra

according to Parseval's theorem,

$$S_{real} = \text{real}\left(\frac{1}{\sqrt{4096}} S_{\phi_1}\right)$$

denotes the noise-free metabolic part of the spectra; only the real part of the spectra was used.

Before adding the noise, too flat and broken spectra were removed; only spectra with an FWHM (measured at the Cr peak at about 3.0 ppm) between 5 and 12.3Hz were kept, which resulted in $N = 69,970$ spectra.

The SNR for each spectra was set randomly by applying a uniform distribution in the range of 6.8 to 18.2 with a mean SNR of 12 and an SD SNR of 2. The SNR referred to the maximum peak of the noise-free metabolic spectra. Thus, the absolute amplitude of the maximum peak of each spectra was identified and divided by the particular SNR value to obtain the standard deviation, σ_{noise}, of the Gaussian noise of each spectrum. A gaussian noise with mean zero and SD, σ_{noise}, was added.

To simulate a shift in frequency, each spectrum was shifted randomly in a range from about -5 to 5 Hz. At the end, each spectrum was normalized to $[-1, 1]$ and cropped to $1,024$ data points in a range from $\sim$5 to $\sim$1.5 ppm.

2.1.2 Baseline Acquisition

The baselines were provided from formerly fitted spectra of in-house previously acquired images. From LCModel-fitted images, the baseline part was obtained and upsampled to the desired size of 1,024. All baselines were then standardized and each baseline normalized to $[-1, 1]$ so that either the maximum or minimum of each baseline was 1 or -1, respectively.

2.1.3 Preprocessing of Simulated Spectra

After the metabolite part and baseline part were prepared, both were added together at a ratio of $1 : 0.8$, respectively. This ratio was manipulated randomly in a range of $+/-25\%$. Afterward, all spectra were normalized to $[-1, 1]$. The spectra were then shuffled randomly and split into a training set of 41,745 spectra and a validation set of 8,946 spectra

and a test set of 10,335 spectra.

The training, validation, and test set, as well as the time vector and the basis set, were saved and then imported into the Python script of Superfit [19].

For the simulated test spectra, the individual SNRs per metabolite were calculated by dividing the value of the highest peak of each metabolic spectrum by the SD of the spectral noise.

2.2 Implementation of Superfit

2.2.1 Concept

The basic idea is to build an autoencoder, such that the input spectrum is denoised, and separated into the single metabolite spectra and the baseline. In a first stage, the baseline features of the spectrum are encoded. Then, a decoder reconstructed the baseline to subtract it from the input spectrum, resulting in a baseline-free spectrum. This was used as the input spectrum for the second stage, which encoded the metabolic feautures to then model the metabolites in a final decoding step. By adding the baseline and the metabolites, for each input spectrum, a reconstruction was built and compared to the input to train the network.

In 2019, Gurbani et al. [20] built an analogous structure and showed, that such a trained network can remove the baseline and quantify simple metabolic singlets. Superfit has a similar structure and also uses the baseline reconstruction via upsampling and convolution with a coiflet low-pass filter like that introduced by Gurbani et al. [20]. Moreover, Superfit was also built to manage the j-coupled appearances of the major metabolites visible in brain MRS.

Baseline Model

To model the baseline, a total of 32 baseline parameters have to be achieved by Superfit. These parameters act as the baseline's nodes assuming a degree of freedom of 32. The vector

$$\boldsymbol{\theta}_B = (\beta_1, \quad \ldots, \quad \beta_{32})$$

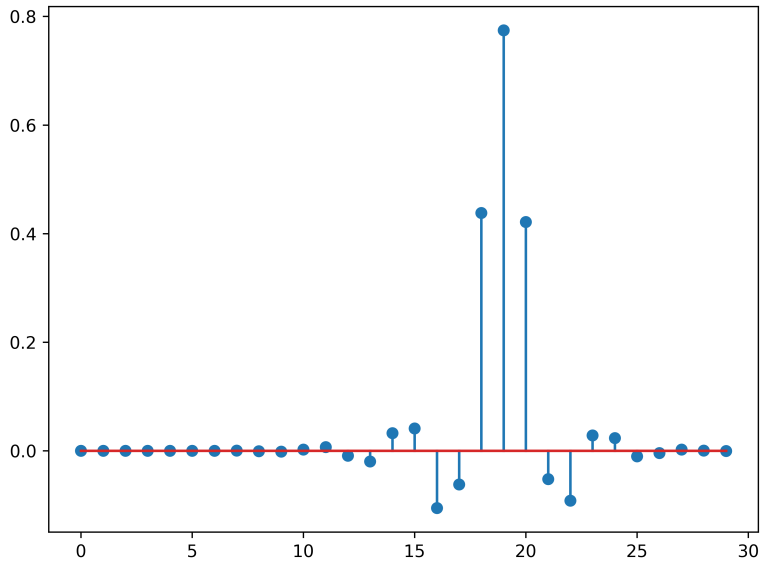

Figure 2.1: Coiflet5 low-pass filter.

of the 32 baseline parameters is upsampled by zipping with a vector of equal length containing only zeros:

$$\boldsymbol{\beta}^{64} = (\beta_1, \quad 0, \quad \beta_2, \quad 0, \quad \ldots, \quad \beta_{32}, \quad 0)$$

followed by a convolution:

$$\boldsymbol{\beta}^{64}_{*} = \boldsymbol{\beta}^{64} * \text{coif5},$$

with coif5 being the fifth-level coiflet low-pass filter [21].

This process is repeated until a vector of a total length of $2,048$ is achieved. To avoid the edge effects of upsampling and convolution (zero values and zero slope), only the central $1,024$ points are used, resulting in the reconstructed baseline, $\boldsymbol{\beta}$.

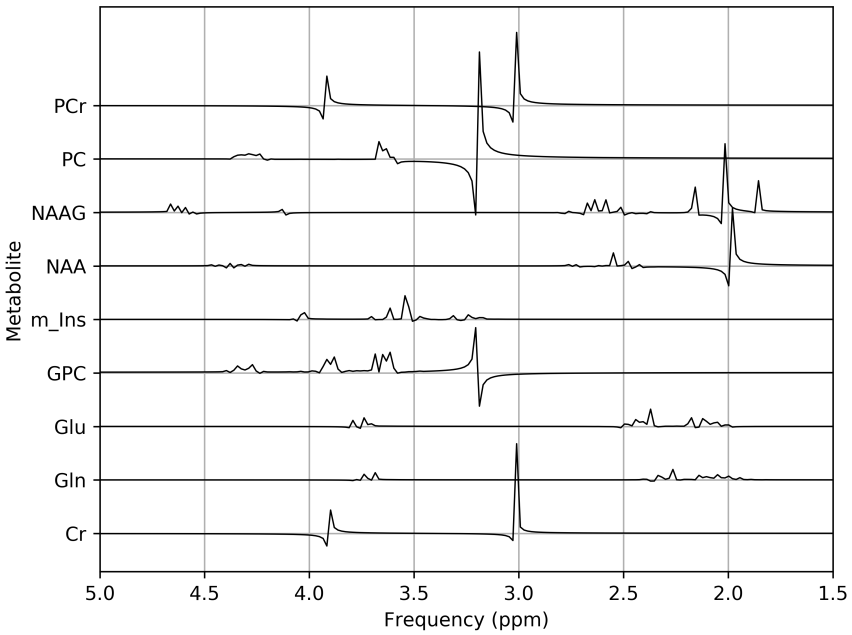

Figure 2.3: The basis set used for the simulation and the reconstruction in the metabolite model.

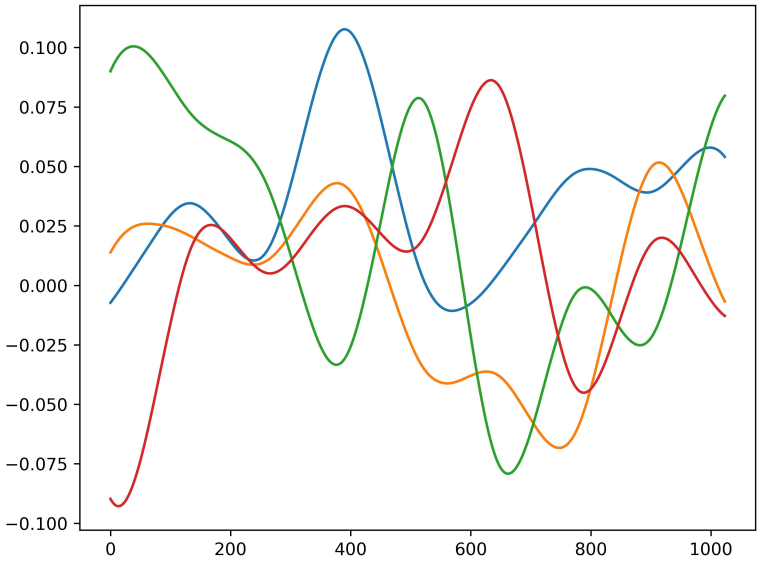

Figure 2.2: Four reconstructed baselines with random input parameters. All plots are in arbitrary units on the y-axis and indeces on the x-axis.

Metabolite Model

To model the metabolic part, the basis set of the simulation was modified for each spectrum by

$$\mathbf{M} = \frac{1}{\sqrt{P}} \left| \sum_{i}^{M} C_i \cdot \mathrm{FFT} \left[\mathbf{M}_i \cdot \exp(\omega) \cdot \exp\left(\frac{-t}{T_L}\right) \cdot \exp\left(\frac{-t^2}{{T_G}^2}\right) \right] \right| \cdot \exp(\mathrm{i}(\phi + (\phi_1 \cdot t + \phi_0))), \tag{2.1}$$

with C_i being the concentration factor for each of the M metabolites, $\mathbf{M}_i$ of the basis set. t denotes the time vector, T_L and T_G denote the Lorentzian and Gaussian lineshape parameters, respectively. ϕ denotes the angle of the spectrum, and phi_0 and ϕ_1 represent the zero- and first-order phasing. ω denotes the shift in frequency defined by

$$\omega = 2\pi\omega_L\omega_s 10^{-6},$$

with ω_L denoting the Larmor frequency and ω_s the shift in frequency in ppm. P stands for the total number of points of the spectrum. The resulting metabolic spectrum was cropped to $1,024$ points in a range from 4.5 to 1.5 ppm. While the concentration parameters, C_i, differ between every metabolite, T_L, T_G, ϕ_0, ϕ_1 and ω_s are assumed to be equal for all metabolites in one spectrum. Having nine metabolites of interest thus leads to a vector $\boldsymbol{\theta}_M$ of 14 metabolite-related parameters that must be computed by Superfit.

To conclude, Superfit is trained to determine a total of 46 parameters to model and reconstruct the noise-free spectrum. By reconstructing the spectrum with two separate models, the input spectrum can be disassembled into the baseline and each single metabolite of interest- which allows its quantification.

2.2.2 Network Architecture

Superfit was designed and optimized using tensorflow 2.0 on one GPU [22].

Structure

The network consisted of two serialized autoencoders for baseline and metabolite encoding and reconstruction. The baseline encoder consisted

of three convolution blocks followed by a flattening layer and two dense layers, which resulted in the 32 baseline parameters. Applying the model above, the baseline was reconstructed and subtracted from the input. The resultant baseline-free spectrum was passed to the metabolite encoder. The metabolite encoder consisted of four convolution blocks, one flattening layer and three dense layers, which resulted in the 14 metabolite parameters. By using the metabolite model, the metabolic resonances were reconstructed. Finally, the baseline and metabolite reconstructions were added, resulting in the reconstructed spectrum.

Convolution Blocks

Each convolution block consisted of four repetitions of a convolutional layer, followed by batch normalization and a leaky rectifier activation function. At the end of each convolution block, a max pooling was performed.

2.2.3 Hyperparameters

The input shape of the network was $batch\ size \times 1,024 \times 1 \times$ None. The batch size was 64, the size of each spectrum was 1024×1 (real part) and the last dimension represents the channel dimension, which was 'None' initially. For the three convolution blocks of the baseline encoder, 32, 64, and 128 filters were used; for the four blocks of the metabolite encoder 32, 64, 128, and again, 128 filters were used, respectively. All filters had a size of 13×1 and the convolutional layers were applied with zero-padding and a stride of 1. Max pooling was done with a pool size of 2×1 with zero-padding and a stride of 2. Leaky ReLUs with $\alpha = 0.5$ were used as the activation functions in the convolutional layers, as well as in the dense layers. For the final dense layer, the identity function was used as the activation function. All layers were used without bias and initialized according to Glorot normal initialization. To prevent overfitting, all kernels were regularized using L2-Norm with $\lambda = 0.01$.

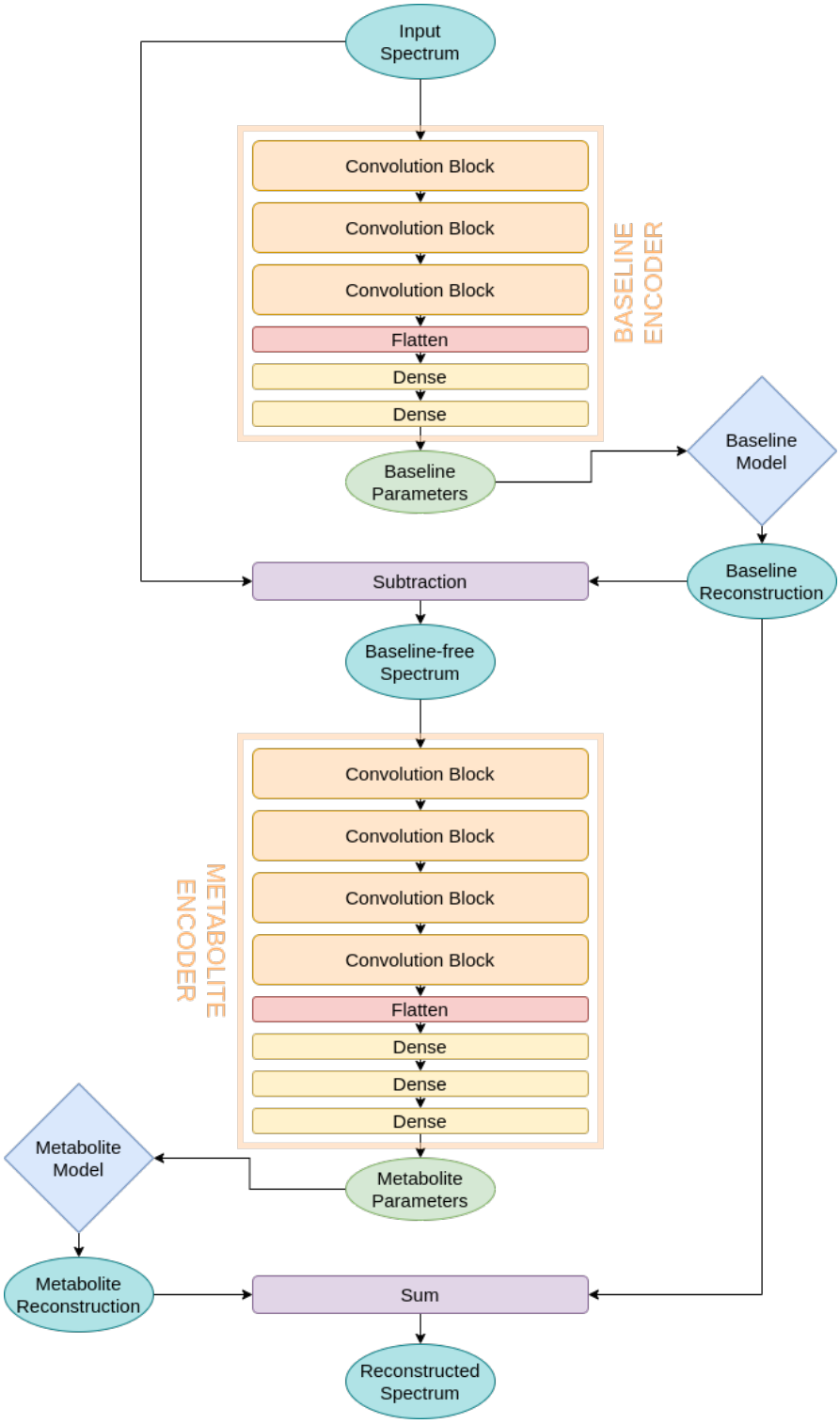

Figure 2.4: Structure of Superfit.

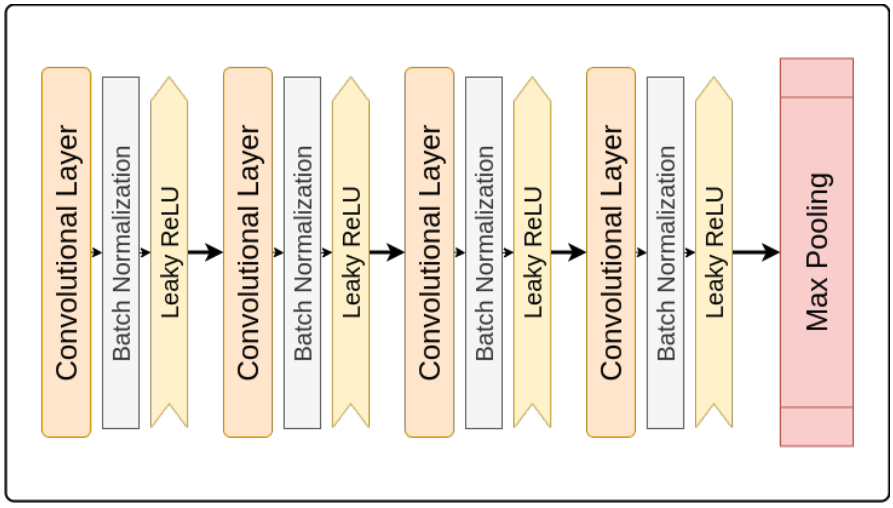

Figure 2.5: Convolution Block.

2.2.4 Training, Validation, and Testing of Superfit

The chosen loss object was the root mean squared error (RMSE) defined as:

$$\mathrm{RMSE}(y_{predicted}, y_{true}) = \sqrt{\frac{1}{1024}\sum(y_{predicted} - y_{true})^2} \tag{2.2}$$

The loss function was defined as the mean of the per-element RMSE within one batch. Superfit was trained using stochastic gradient descent with a decaying learning rate and Nesterov momentum. The momentum was set to 0.83. The initial learning rate was set to 0.0001 and decayed by a factor of 0.5 after an initial step size of 25 epochs; the step size grew by 50 after every step with regard to the approximately exponential decay of a loss function over time.

At the end of each epoch, the loss was computed for the validation set. The training process continued until the validation loss converged or the maximum epoch of 500 was reached.

After training, the test loss was calculated for the test set. For all spectra in the test set, the model parameters were calculated, and, using the metabolic model, the metabolite-wise spectra were reconstructed.

2.3 Evaluation of Superfit in Metabolite Quantification

To compare the relative metabolite quantifications estimated by Superfit with the ground truth, every metabolic component for all test spectra was reconstructed separately to calculate the individual relative metabolite concentrations. For all relative metabolite concentrations, the absolute percentage errors (APE) and the corresponding mean absolute percentage errors (MAPE) and SD were calculated. APEs were also calculated for total NAA (tNAA), total choline (tCho), total creatine (tCr) and total glutamate and glutamine (Glx). Furthermore, median APEs (MedAPEs) were calculated for all metabolites. For the ratios of tCho, tCr, Glx and m-Ins with respect to tNAA, and for the ratios NAAG/NAA, PC/GPC, Cr/PCr, and Gln/Glu, the individual MAPEs and the corresponding Pearson correlation coefficients were calculated.

The expected concentration ranges of the metabolic concentrations with respect to tNAA were calculated as the mean ± twice the SD.

To compare the results of Superfit with the ground truth, the correlations were plotted and the corresponding Bland-Altman plots were created [23]. Furthermore, a paired t-test was performed. A p-value < 0.05 was considered to be statistically significant.

3 Results

3.1 Simulated Spectra

The simulation created 59,635 spectra for the training process: 41,745 training spectra and 8,946 validation spectra. For testing and statistics, another 10,335 spectra were simulated.

The individual SNRs per metabolite were 10.57 ± 4.79 for tNAA, 4.76 ± 1.81 for tCho, 11.12 ± 2.63 for tCr, 3.14 ± 1.32 for Glx and 4.70 ± 1.87 for m-Ins. The individual SNR distribution per metabolite is shown in Figure 3.1. The FWHM of the simulated spectra was 7.39Hz ± 1.90Hz and is illustrated in Figure 3.2.

Figures 3.3, 3.4 and 3.5 show spectra of a minimal SNR of 7, a mean SNR of 12, and a maximum SNR of 18, including all metabolic components, baseline, and noise.

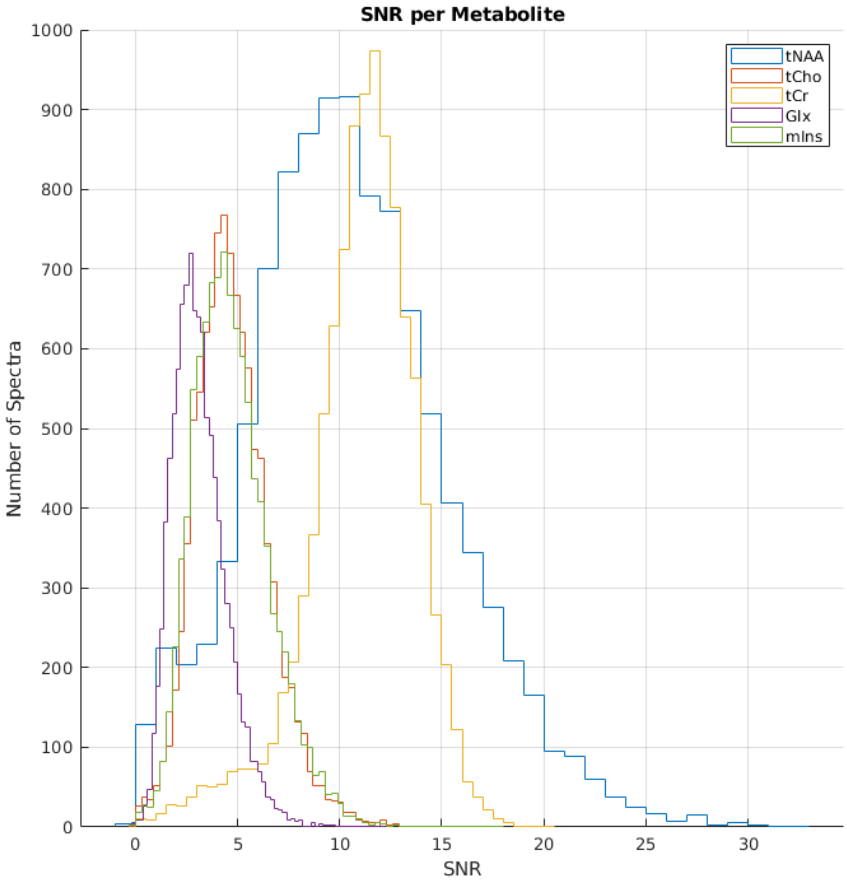

Figure 3.1: Individual SNR distribution per metabolite

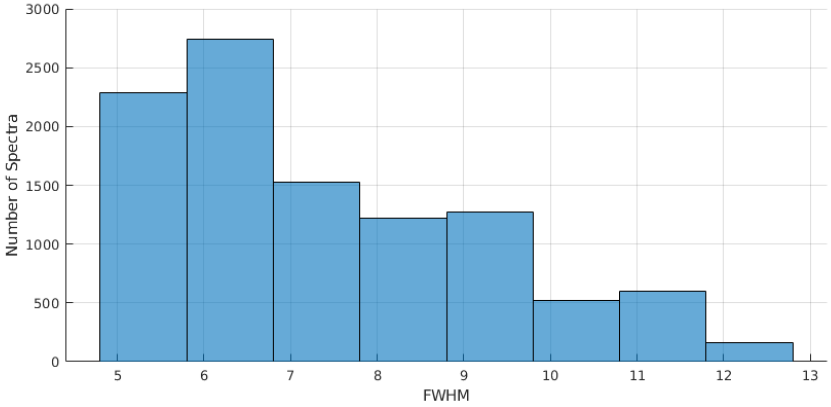

Figure 3.2: Histogram of FWHM of the simulated test spectra.

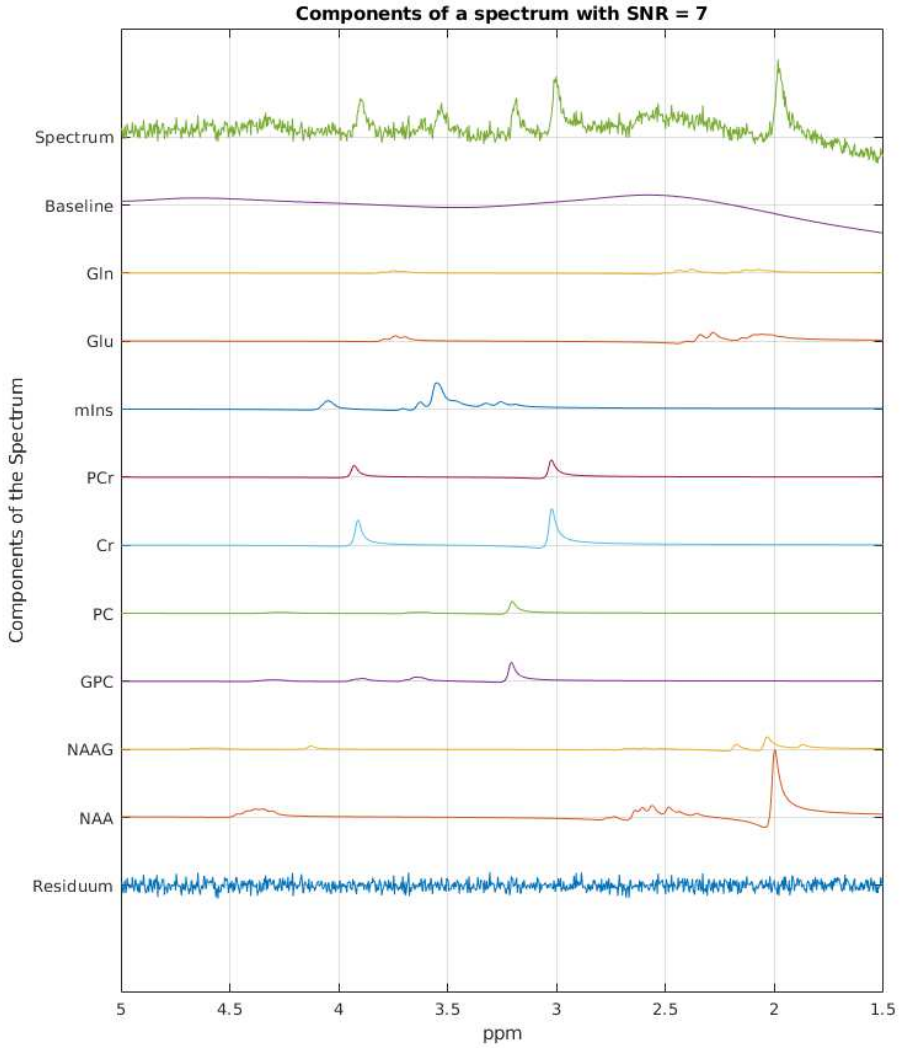

Figure 3.3: Spectral components of a spectrum with SNR = 7. The green spectrum on the top illustrates the final input spectrum containing all metabolic components, the baseline, and the noise. The second line shows the baseline, followed by the individual metabolite spectra, and, finally, the noise.

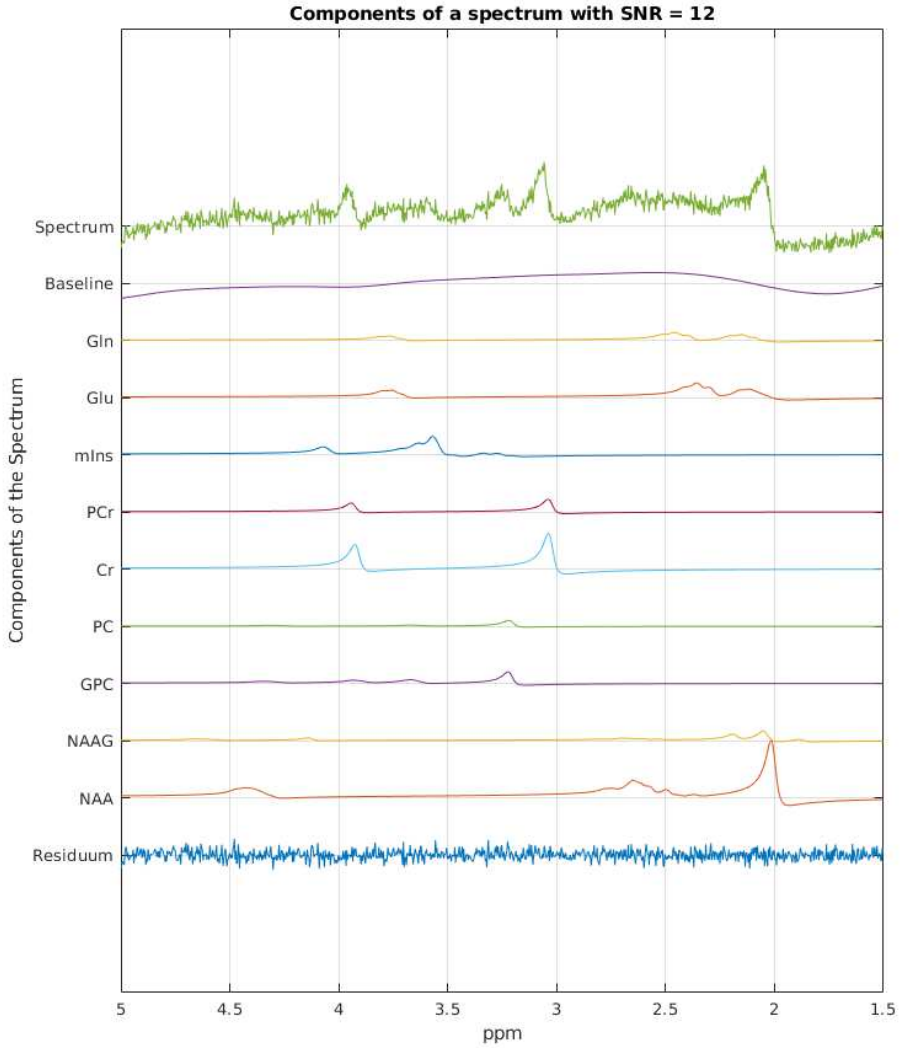

Figure 3.4: Spectral components of a spectrum with SNR = 12. The green spectrum on the top illustrates the final input spectrum containing all metabolic components, the baseline, and the noise. The second line shows the baseline, followed by the individual metabolite spectra, and, finally, the noise.

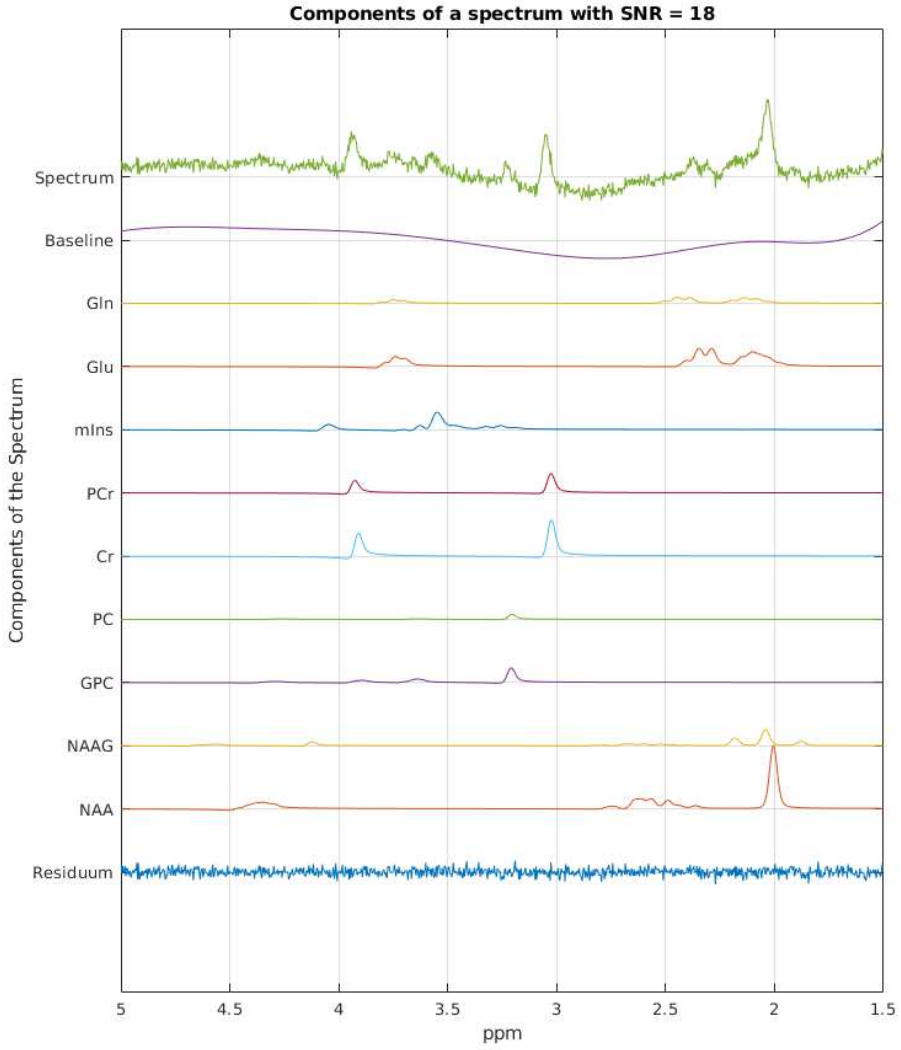

Figure 3.5: Spectral components of a spectrum with SNR = 18. The green spectrum on the top illustrates the final input spectrum containing all metabolic components, the baseline, and the noise. The second line shows the baseline, followed by the individual metabolite spectra, and, finally, the noise.

3.2 Training and Performance of Superfit

The training process stopped at the 501st epoch because a maximum number of 500 epochs was defined.

The optimal generalization loss (RMSE of all true and noisy data) was 0.0193547. The RMSE loss and the accuracy (1 - mean absolute error) were 0.0198515 and 98.4174% for the training set, 0.0198196 and 98.4204% for the validation set, and 0.0197522 and 98.4262% for the test set, respectively. The SD of the test RMSE was 0.008886539. For loss and accuracy history during the training process, see Figure 3.6 and 3.7.

In the test set, the minimum RMSE of a spectrum was 0.004032349, the RMSE closest to the mean was 0.0197522, and the maximum was 0.08043192. For all three RMSEs, input and reconstruction are illustrated in Figures 3.9, 3.9 and 3.10.

The total elapsed run time for fitting the test set (N = 10,335) was 39s on a PC using one GPU [GeForce GTX 1050/PCIe/SSE2]: 1s for loading and preprocessing the spectra, 2s for building the network and model, 29s for computing the metabolite (and baseline) parameters by Superfit, and another 7s to reconstruct the baseline and metabolites.

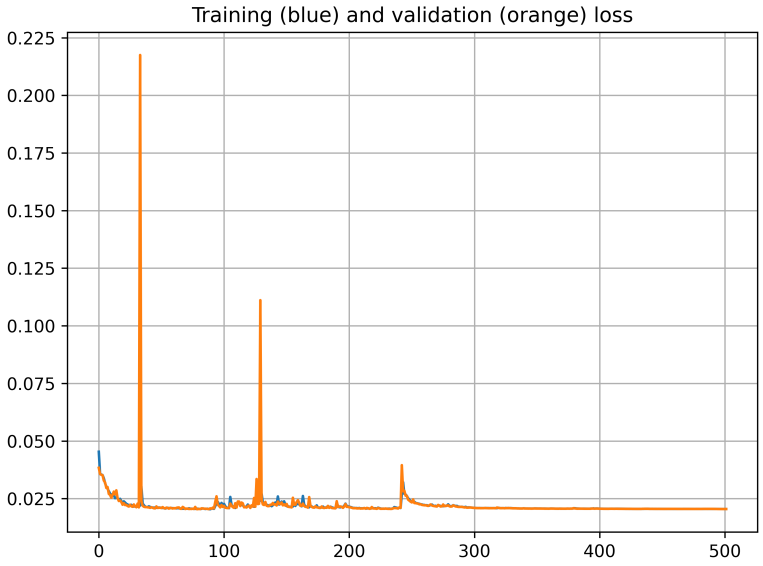

Figure 3.6: Loss history of the training of Superfit showing a rapid decrease at the beginning and three peaks illustrating crucially wrong weightings during the training process. After the third peak, the loss converges.

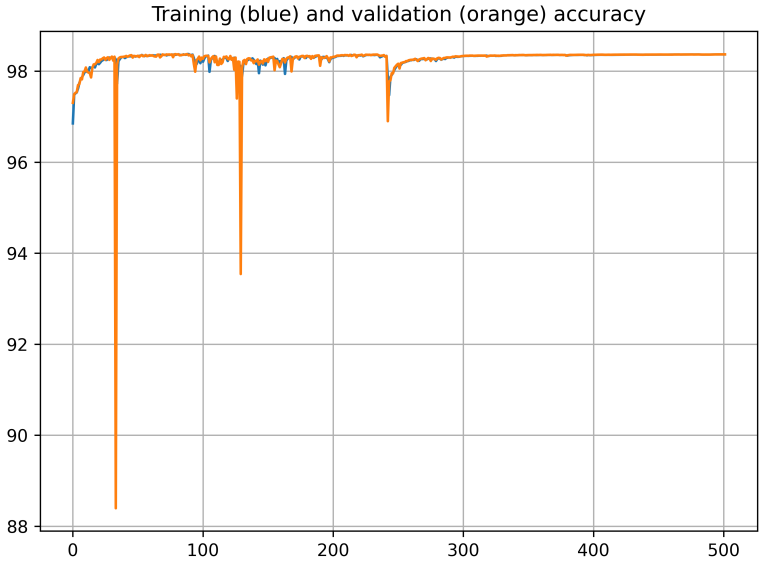

Figure 3.7: Accuracy history of the training of Superfit showing a rapid increase at the beginning. Three times the accuracy crashed, but then converges.

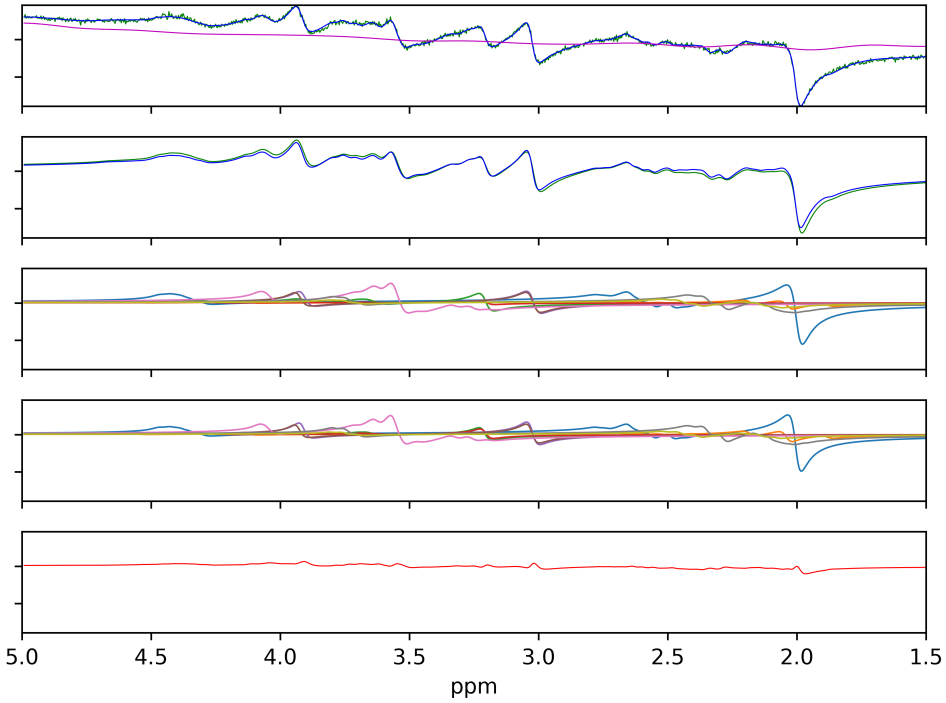

Figure 3.8: Reconstructions of Superfit for a spectrum with a minimum RMSE. From top to bottom: 1) noisy input spectrum (green), Superfit reconstruction (blue), and reconstructed baseline(pink); 2) metabolite-only ground truth (green) and reconstruction (blue); 3) ground truth metabolic components; 4) reconstruction of metabolic components; 5) difference between the input and reconstructed spectrum. For such a low SNR and loss, the input spectrum was fit very accurately.

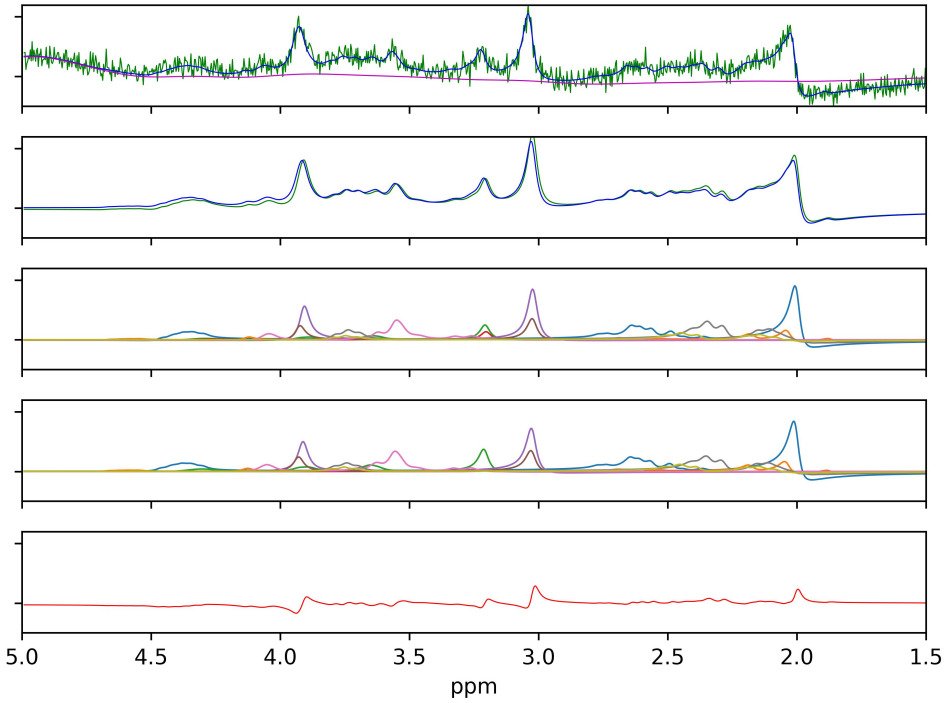

Figure 3.9: Reconstructions of Superfit for a spectrum with mean RMSE. From top to bottom: 1) noisy input spectrum (green), Superfit reconstruction (blue), and reconstructed baseline(pink); 2) metabolite-only ground truth (green) and reconstruction (blue); 3) ground truth metabolic components; 4) reconstruction of metabolic components; 5) difference between the input and reconstructed spectrum. Even though the SNR seems to be high, the input spectrum could be fit.

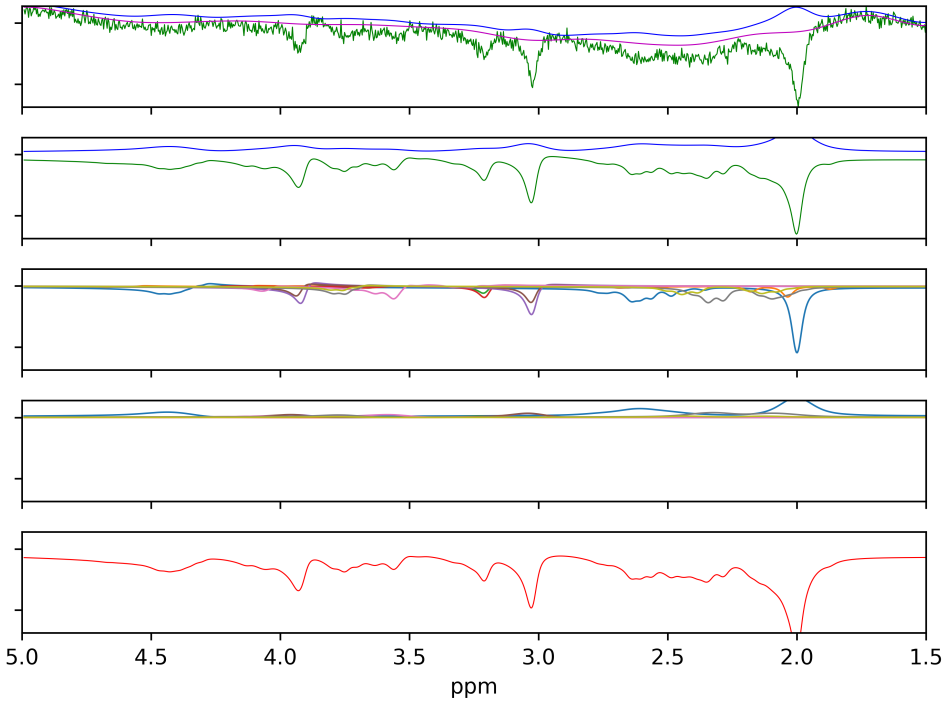

Figure 3.10: Reconstructions of Superfit for a spectrum with maximum RMSE. From top to bottom: 1) noisy input spectrum (green), Superfit reconstruction (blue), and reconstructed baseline(pink); 2) metabolite-only ground truth (green) and reconstruction (blue); 3) ground truth metabolic components; 4) reconstruction of metabolic components; 5) difference between the input and reconstructed spectrum. The input spectrum could not be fit; the reconstructed spectrum appears as a rather flat line, which results in a high difference.

3.3 Metabolite Quantification

The resulting relative metabolite quantifications, the MAPEs, and the MedAPEs from all simulated test spectra by using the implemented metabolite reconstruction are shown in Table 3.1. The overall MAPE for all nine metabolites was 18.50% ± 16.10%. NAAG, GPC, PC, and Gln had a MAPE > 20%. Excluding these, the overall MAPE was 12.32% ± 9.81%. For NAA (6.4% ± 5.58%) and m-Ins (9.19% ± 7.78%), the MAPEs were ¡ 10%. For tNAA (6.08% ± 5.30%), tCho (9.77% ± 8.51%), and tCr (5.75% ± 5.34%), the MAPEs were < 10%; for Glx (14.14% ± 10.10%) it was < 20%. The MedAPEs, which are more robust to bias, were, in general, lower than the calculated MAPEs. The MedAPEs were 4.95% for NAA, 25.70% for NAAG, 22.15% for GPC, 39.28% for PC, 10.08% for Cr, 14.49% for PCr, 7.40% for m-Ins, 14.04% for Glu, and 22.31% for Gln. For tNAA, tCho, tCr, and Glx, the MedAPEs were 4.75%, 7.74%, 4.43%, and 12.58%, respectively.

Metabolite Concentrations				
Metabolite	Ground Truth	Prediction	MAPE (in %)	MedAPE (in %)
NAA	14.40 ± 4.13	14.54 ± 4.20	6.40 ± 5.58	4.95
NAAG	2.96 ± 1.26	2.78 ± 0.96	34.29 ± 33.35	25.70
GPC	3.71 ± 1.33	3.76 ± 1.51	27.98 ± 24.31	22.15
PC	1.81 ± 0.84	1.75 ± 4.13	53.06 ± 50.95	39.28
Cr	9.18 ± 3.29	8.97 ± 0.98	12.17 ± 9.76	10.08
PCr	5.22 ± 1.62	5.45 ± 3.36	18.16 ± 15.12	14.49
m-Ins	7.99 ± 2.37	7.98 ± 1.41	9.19 ± 7.78	7.40
Glu	8.39 ± 2.08	8.01 ± 2.27	15.66 ± 10.85	14.04
Gln	4.11 ± 1.00	3.27 ± 1.38	27.84 ± 22.50	22.31
tNAA	16.54 ± 4.28	16.60 ± 1.25	6.08 ± 5.30	4.75
tCho	5.45 ± 1.74	5.44 ± 4.43	9.77 ± 8.51	7.74
tCr	14.25 ± 4.33	14.28 ± 1.77	5.75 ± 5.34	4.43
Glx	12.00 ± 2.39	10.90 ± 4.42	14.14 ± 10.10	12.58

Table 3.1: Mean metabolite concentrations and SD of ground truth and prediction, the respective MAPEs, and the MedAPE.

The MAPEs of the metabolite concentrations are also illustrated as boxplots in Figure 3.11. Figures 3.12 - 3.20 show the scatter plots for the true and predicted concentrations per metabolite (except for GPC, PC, PCr, and Cr) and the corresponding Bland-Altman plots. These plots illustrate the similarity of prediction and ground truth; the blue

line highlights the mean difference between the two 'measurements' and the red line highlights the limits of agreement, which were ±2 SD.

For m-Ins, the p-value was 0.4149, so the true and predicted concentrations of m-Ins are not significantly different. The same applied for tCho with a p-value of 0.0553. For all other metabolites, the corresponding p-values were < 0.05; thus, their true and predicted concentrations are significantly different.

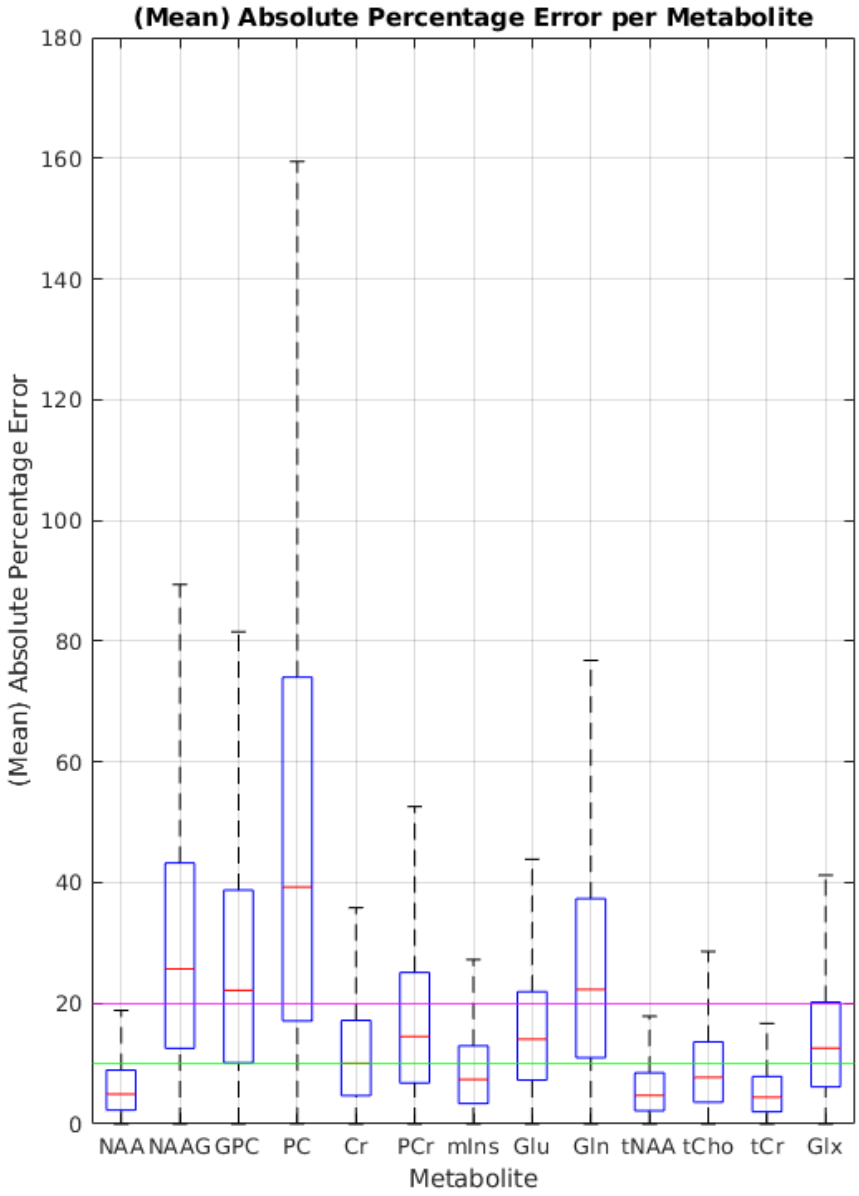

Figure 3.11: (M)APEs per metabolite illustrated as boxplots. The blue boxes show the interquartile range, the red lines inside the boxes represent the corresponding medians. The dashed whiskers show the range of ±2.7 SD, which covers about 99.7% of the data. The green and the red line highlight 10% and 20%, respectively.

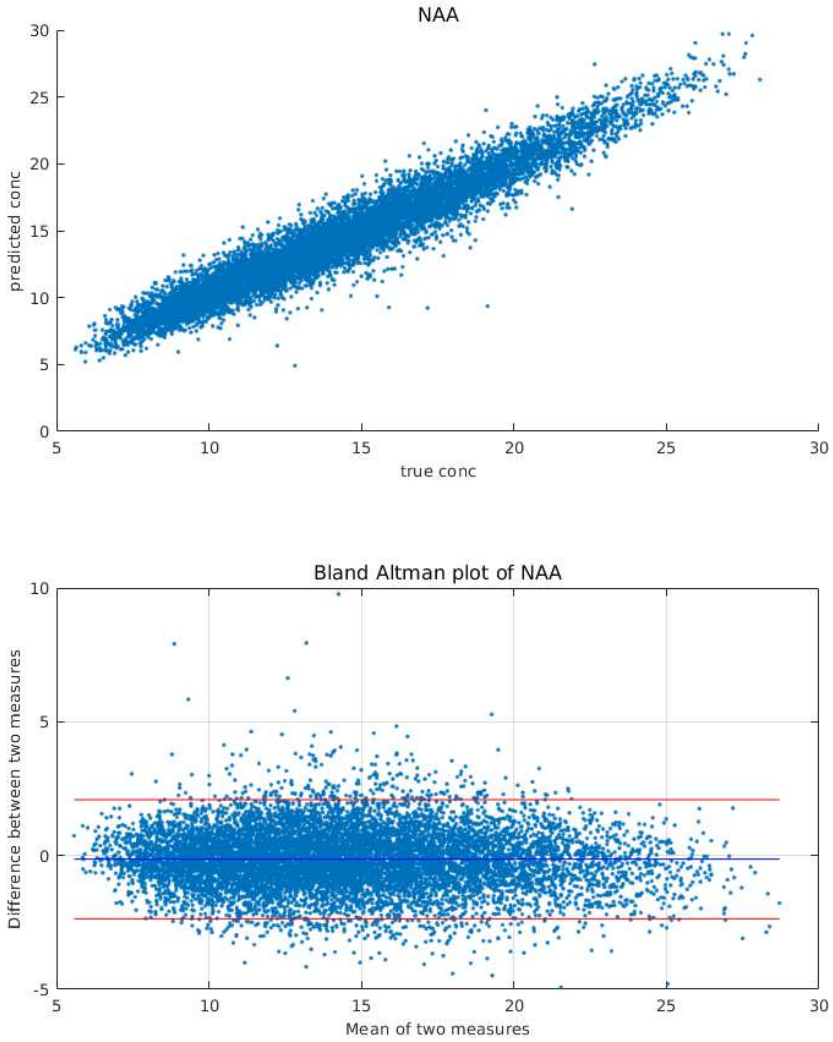

Figure 3.12: Correlation and Bland-Altman plot for NAA. For NAA, the p-value was < 0.0001.

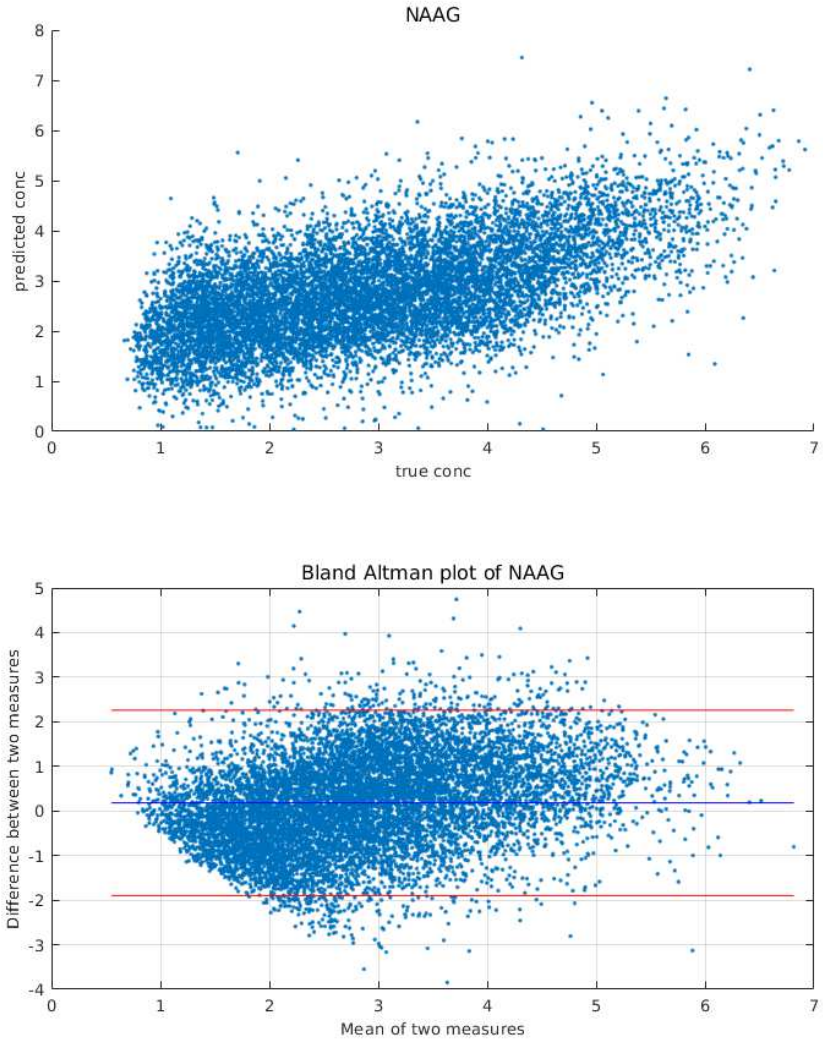

Figure 3.13: Correlation and Bland-Altman plot for NAAG. For NAAG, the p-value was < 0.0001.

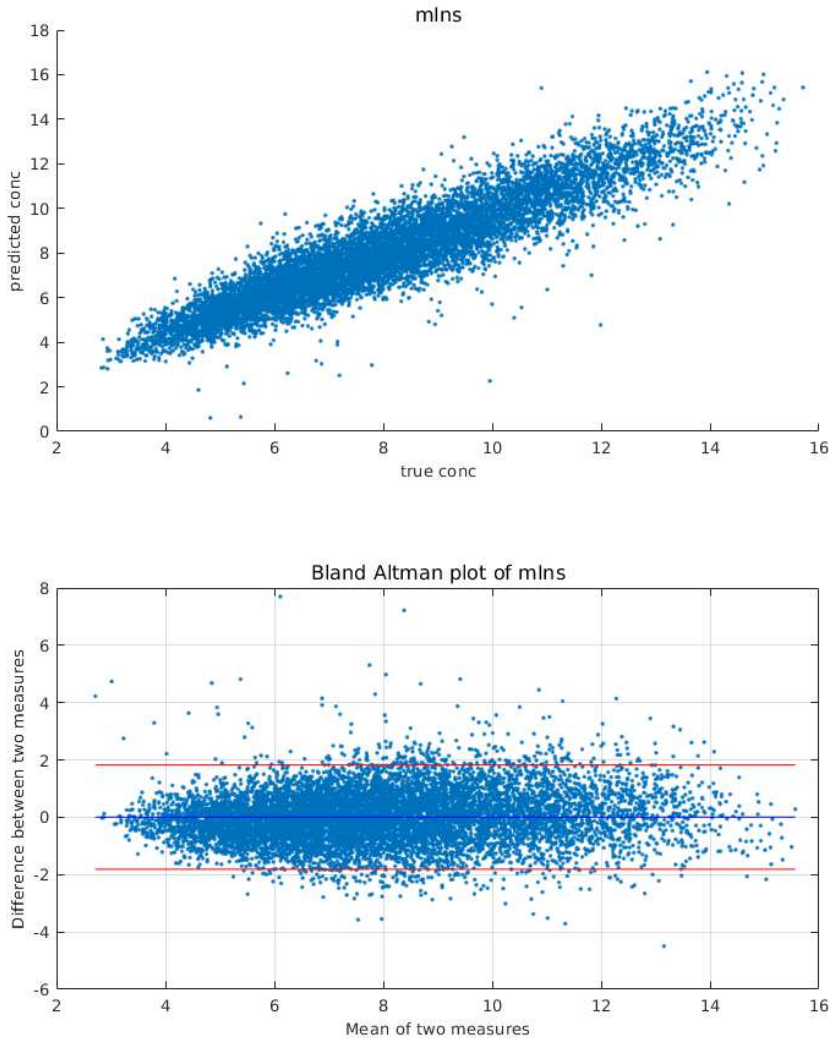

Figure 3.14: Correlation and Bland-Altman plot for m-Ins. For m-Ins, the p-value was $p = 0.4149$.

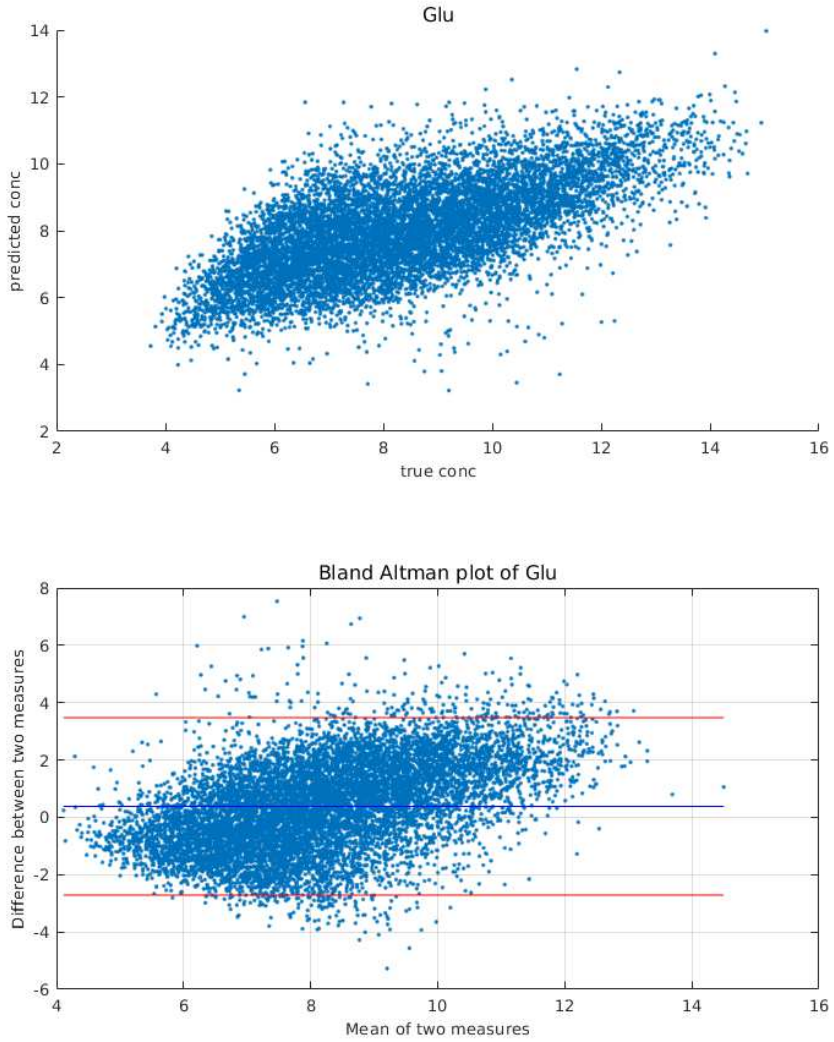

Figure 3.15: Correlation and Bland-Altman plot for Glu. For Glu, the p-value was < 0.0001.

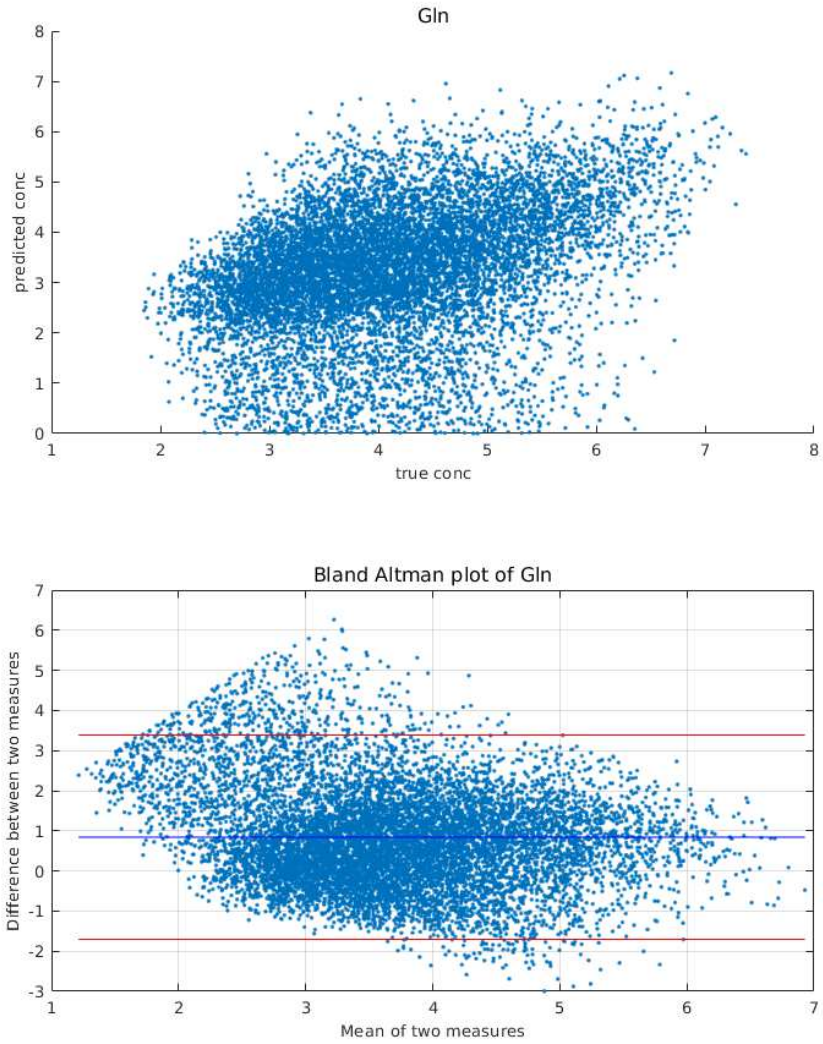

Figure 3.16: Correlation and Bland-Altman plot for Gln. For Gln, the p-value was < 0.0001.

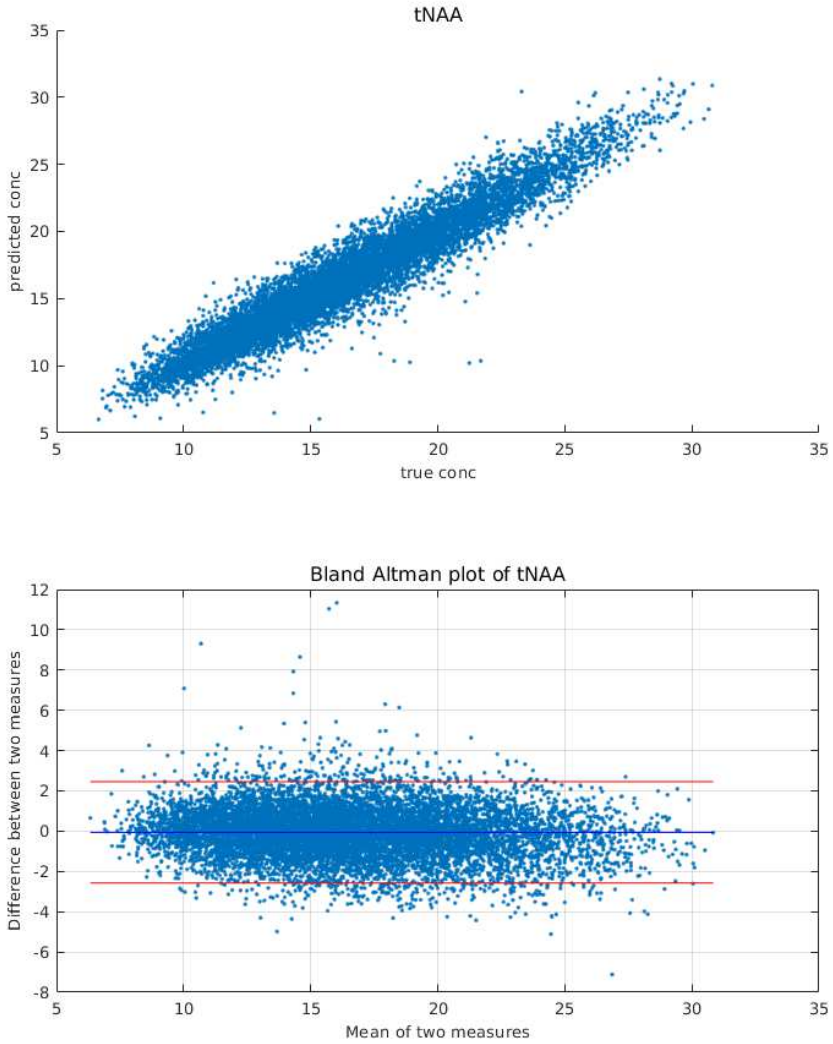

Figure 3.17: Correlation and Bland-Altman plot for tNAA. For tNAA, the p-value was < 0.0001.

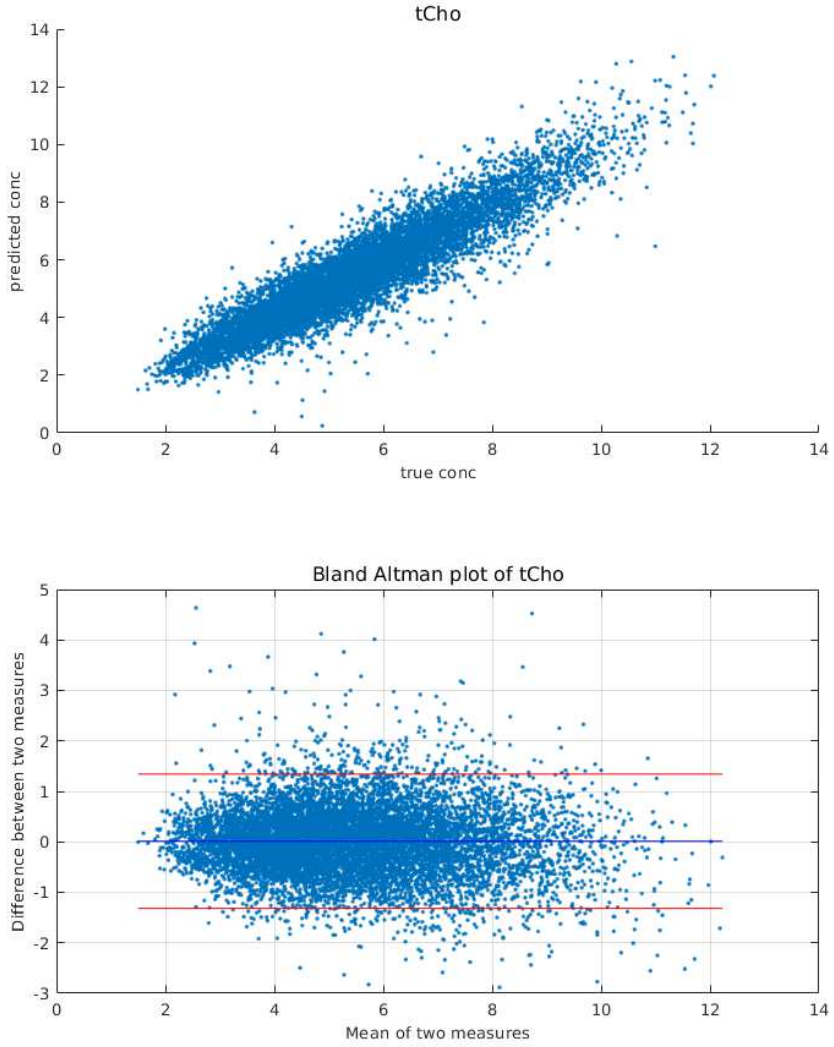

Figure 3.18: Correlation and Bland-Altman plot for tCho. For tCho, the p-value was $p = 0.0553$.

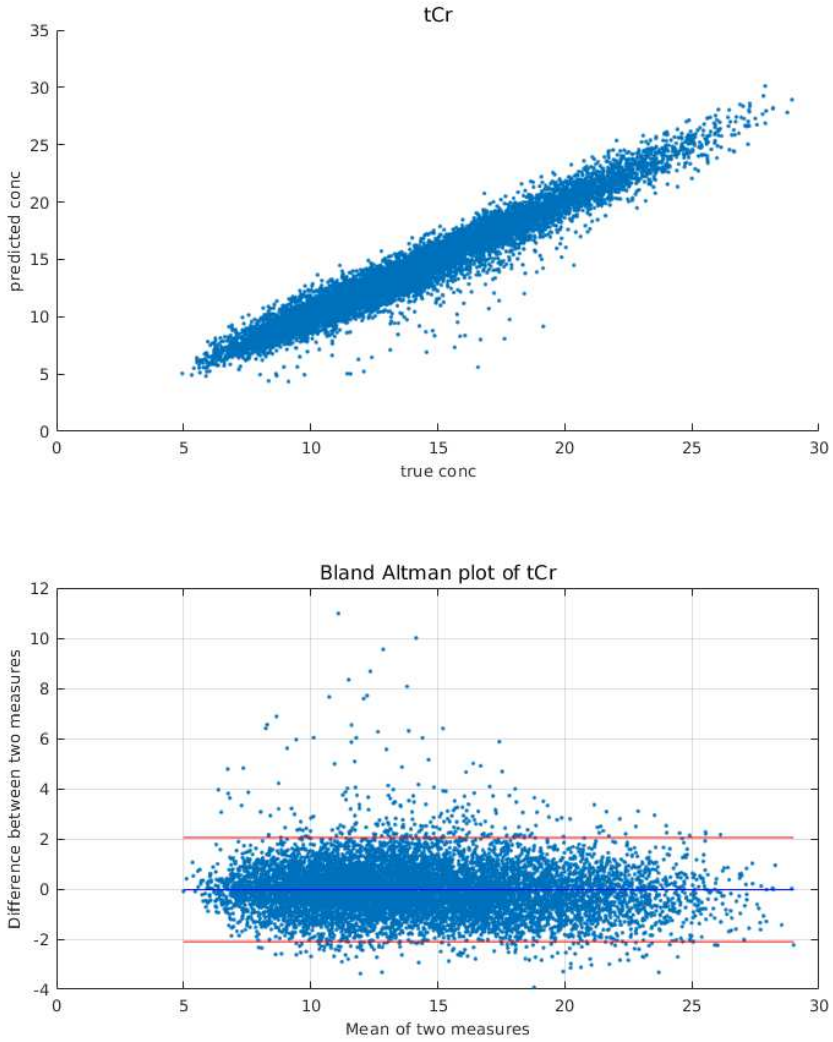

Figure 3.19: Correlation and Bland-Altman plot for tCr. For tCr, the p-value was $p = 0.0063$.

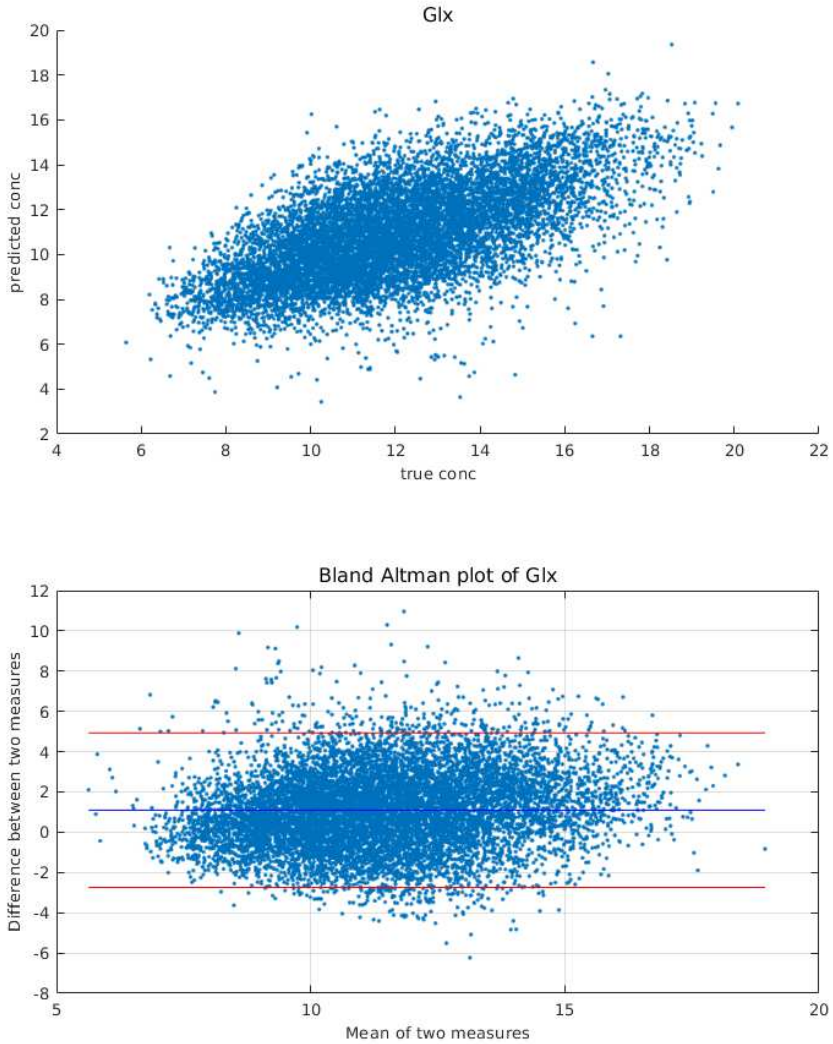

Figure 3.20: Correlation and Bland-Altman plot for Glx. For Glx, the p-value was < 0.0001.

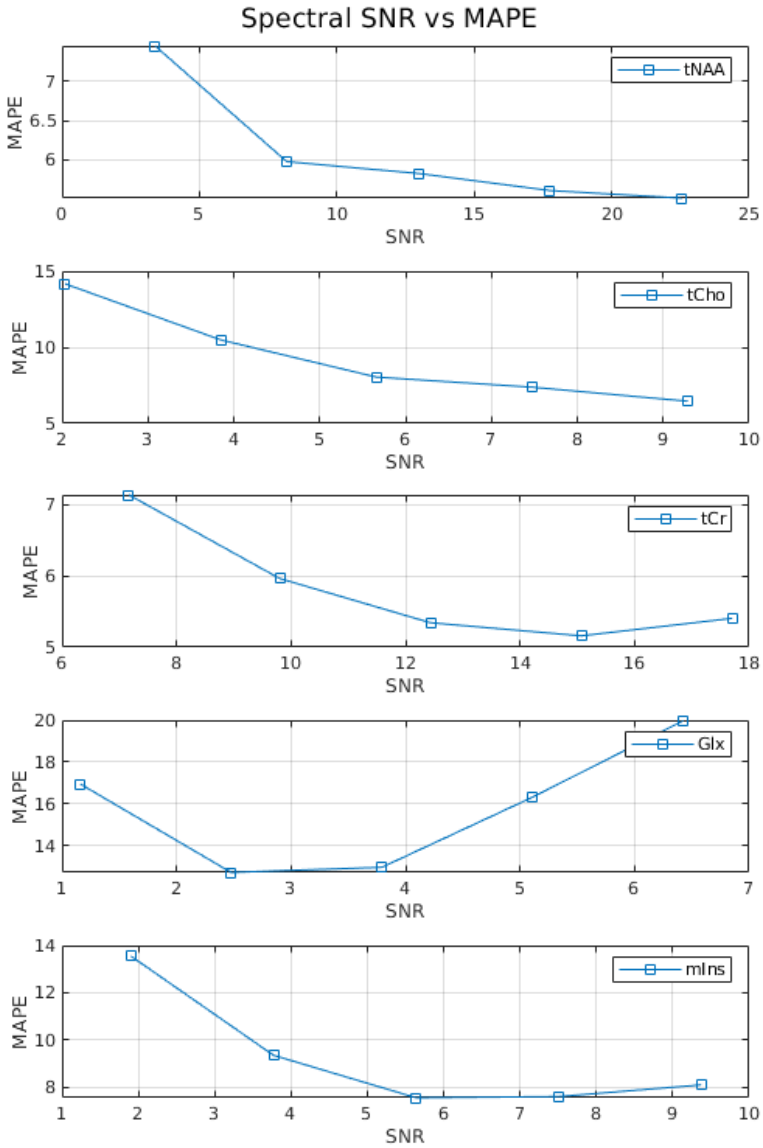

Figure 3.21: MAPE of tNAA, tCho, tCr, Glx and m-Ins in relation to the respective SNR.

Furthermore, for tNAA, tCho, tCr, Glx, and m-Ins, MAPEs were plotted against their individual SNRs in five intervals from −2· SD to +2· SD (Figure 3.21). Except for Glx, for all the other four metabolites, the MAPEs decreased with increasing SNRs. For Glx, the MAPE was lowest at the mean SNR and decreased with increasing distance from the mean SNR.

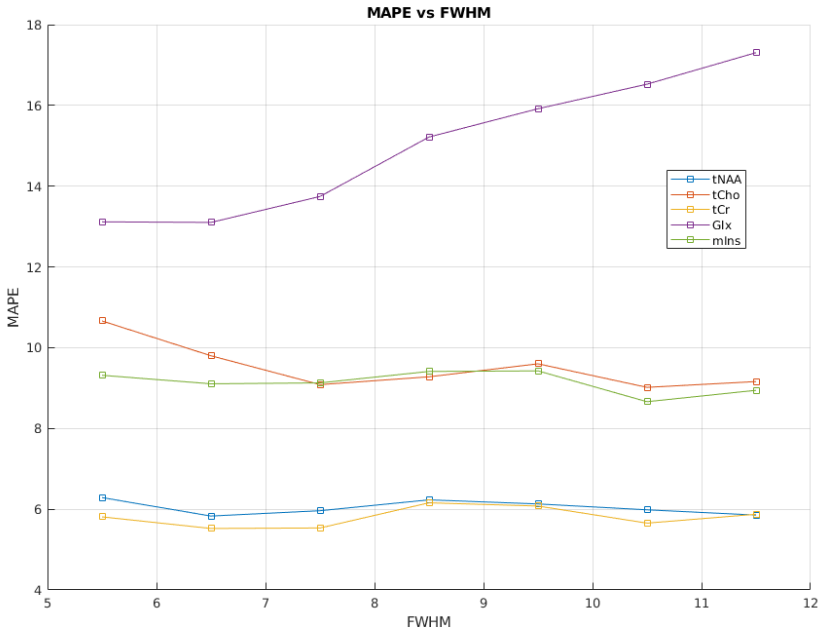

Figure 3.22: MAPE of tNAA, tCho, tCr, Glx and m-Ins in relation to the FWHM.

Also, to illustrate the relation between the MAPEs and the FWHM, Figure 3.22 shows the individual MAPEs corresponding to seven equal intervals from FWHM < 6 to FWHM > 11. Similar to the SNR, the MAPEs of all metabolites except Glx decreased with increasing FWHM. The MAPE of Glx increased significantly with increasing FWHM.

The ratios of NAAG/NAA, PC/GPC, Cr/PCr, and Gln/Glu had MAPEs $> 20\%$. In particular, PC/GPC had a MAPE of $460\% \pm 5200\%$. All four relative metabolite concentrations with respect to NAA, except Glx/tNAA ($14\% \pm 9.50\%$) had a MAPE $< 10\%$. MAPEs were $5.80\% \pm 4.90\%$ for tCr/tNAA, $9.50\% \pm 8.20\%$ for tCho/tNAA, and $9.70\% \pm 8.10\%$ for m-Ins/tNAA. The corresponding MedAPEs were lower than the MAPEs. All mean true and predicted ratios, the corresponding MAPEs and SD, as well as the MedAPEs, are shown in Table 3.2 and illustrated as boxplots in Figure 3.23.

Relative Metabolite Concentration Ratios				
Ratio	Ground Truth	Prediction	MAPE (in %)	MedAPE (in %)
tCho/tNAA	0.34 ± 0.12	0.34 ± 0.12	9.50 ± 8.20	7.48
tCr/tNAA	0.89 ± 0.29	0.89 ± 0.29	5.80 ± 4.90	4.70
Glx/tNAA	0.76 ± 0.21	0.68 ± 0.14	14.00 ± 9.50	12.66
m-Ins/tNAA	0.51 ± 0.18	0.50 ± 0.17	9.70 ± 8.10	7.79
NAAG/NAA	0.22 ± 0.10	0.20 ± 0.07	37.00 ± 36.00	27.45
PC/GPC	2.50 ± 1.50	10.00 ± 99.00	460.00 ± 5200.00	56.60
Cr/PCr	1.80 ± 0.53	1.60 ± 0.38	26.00 ± 20.00	22.24
Gln/Glu	0.51 ± 0.15	0.41 ± 0.13	30.00 ± 23.00	25.06

Table 3.2: Mean relative metabolite concentration ratios and SD of ground truth and prediction, the respective MAPEs, and the MedAPEs.

The relative metabolite concentration ranges (containing about 95% of the spectra) were 0.34 ± 0.24 in the simulated and 0.34 ± 0.24 in the predicted spectra for tCho/tNAA, 0.89 ± 0.58 in the simulated and 0.89 ± 0.58 in the predicted spectra for tCr/tNAA, 0.76 ± 0.42 in the simulated and 0.68 ± 0.28 in the predicted spectra for Glx/tNAA and 0.51 ± 0.36 in the simulated and 0.50 ± 0.34 in the predicted spectra for m-Ins/tNAA.

To visualize the relation between the true and predicted ratios, scatter plots were created for each of the ratios, which clearly showed the strong positive correlation between the true and predicted relative metabolite concentrations with respect to tNAA (Figure 3.24). For tCho/tNAA, tCr/tNAA and m-Ins/tNAA, a Pearson correlation coefficient of $Rho > 0.9$ showed a strong correlation between the ground truth and the estimated ratios. Also Glx/tNAA with a $Rho = 0.8019$ showed a correla-

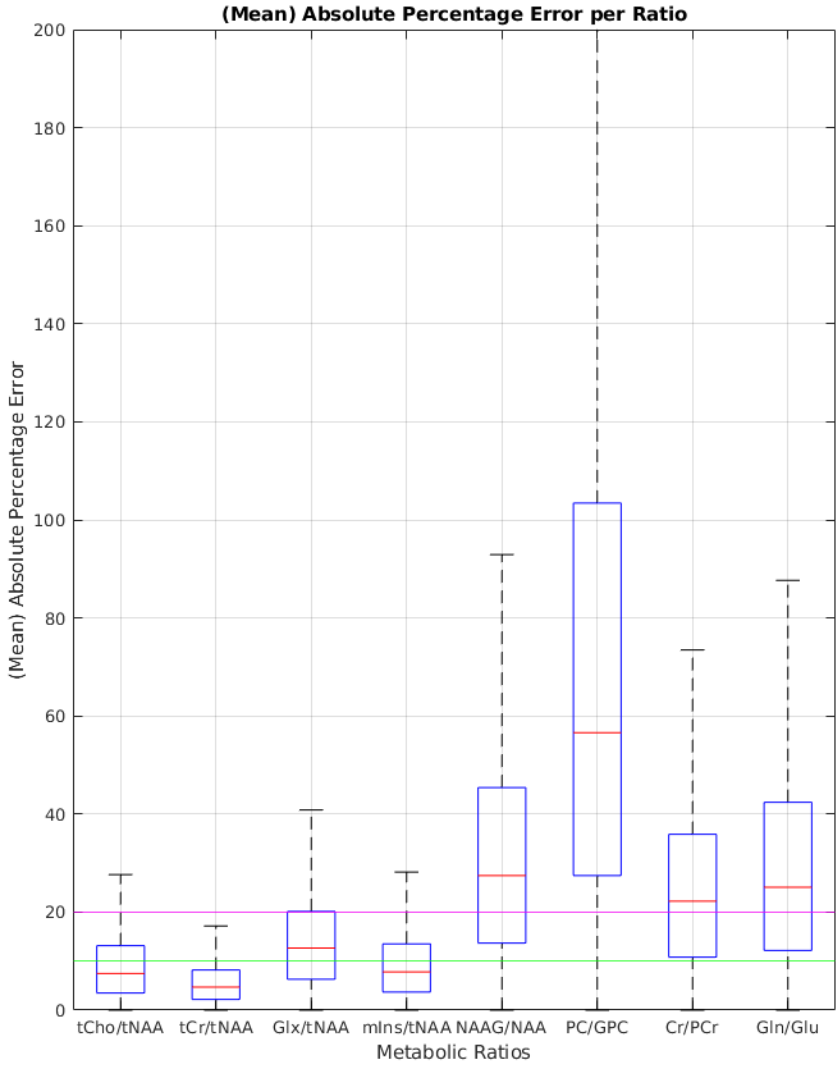

Figure 3.23: (M)APEs per ratio illustrated as boxplots. The blue boxes show the interquartile range, the red lines inside the boxes represent the corresponding medians. The dashed whiskers show the range of ±2.7 SD, which covers about 99.7% of the data. The big green and the big red line highlight 10% and 20%, respectively.

tion between ground truth and estimation. Although NAAG/NAA still showed a positive correlation ($Rho = 0.5526$), all four ratios, including NAAG/NAA, PC/GPC, Cr/PCr, and Gln/Glu, appeared to have a low correlation or no correlation at all corresponding to the respective MAPEs. For all ratios except tCho/tNAA, the p-values were < 0.05. For tCho/tNAA, the p-value was 0.2467.

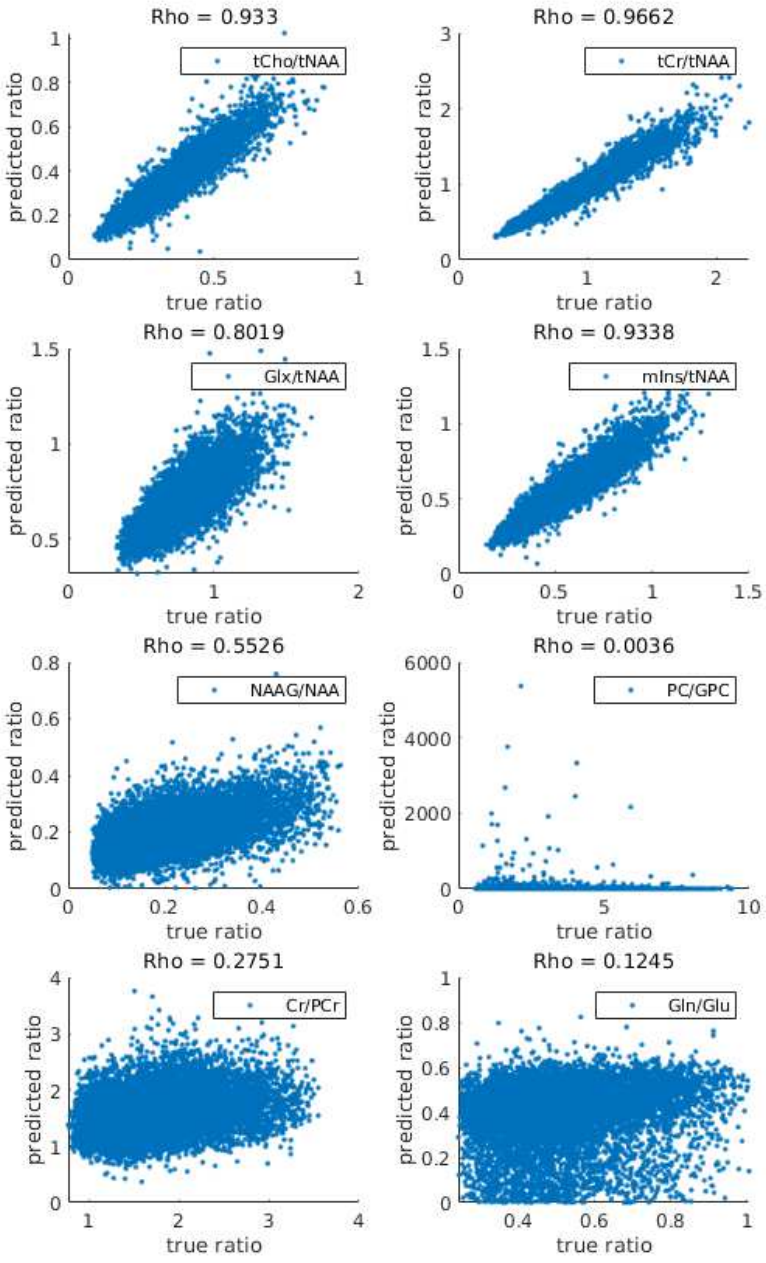

Figure 3.24: Scatter plots of all ratios with the corresponding correlation coefficient Rho.

4 Discussion

In a routine clinical setting, brain metabolite spectra are inevitably degraded due to many factors: i.a. line broadening, low SNR, overlapping metabolic signals, spectral baselines and artefacts originating from lipids or water. Currendtly used software for metabolite quantifications are based on nonlinear least squares fitting algorithms, which are not suitable for all MR metabolic signals [2] [24]. Furthermore, these algorithms are extremely time-consuming; consequently, MRSI is still mainly used in research, but rarely in clinical protocols [9].

In the few past years, several studies have proven a potentially high applicability of deep learning in the context of proton MRSI of the brain. Kyathanahally et al. reported detection and removal of spectroscopic artifacts via deep learning [25]. Das et al. quantified major metabolic peaks using machine learning in simulated and *in vivo* spectra [26]. Hatami and colleagues reported deep learning-based metabolite quantification in the time domain [27].

In particular, the studies of Gurbani et al. and Lee et al. were a motivation for this thesis: Gurbani et al. managed to improve spectral quality and to remove the spectral baseline by a CNN trained with unsupervised learning. These authors focused only on the three singlet peaks of NAA, Cho, and Cr [20]. Lee et al. proved the quantification of all representative MR-visible brain metabolites using a CNN, but optimized with supervised learning, and first-order phase distortion was not included [28].

4.1 Concept and Performance of Superfit

Inspired by Lee et al. and Gurbani et al., and also motivated by the great potential of MRSI applications in clinical protocols, Superfit was developed, which is a CNN that consists of two serialized autoencoders that can disassemble simulated spectra into spectral baseline, noise, and

the metabolite components, including NAA, NAAG, GPC, PC, Cr, PCr, m-Ins, Gln, and Glu. The simulated spectra were degraded by noise and baseline, line broadening, frequency shift, zero- and first-order phase distortions, and overlapping metabolic signals.

After training, Superfit performed on the test set with a RMSE of 0.0197522 ± 0.0088865, which was close to the optimal generalization RMSE of 0.0193547. Loading, fitting, and reconstructing the 10,335 test spectra took less than 40s. For 3D MRSI volumes of $32 \times 32 \times 16$ containing over 16,000 spectra, for instance, spectral fitting could be performed by Superfit in less than a minute.

However, Superfit was limited. Low SNR, but especially extreme line broadening and first-order distortions made it difficult, and in some cases impossible, to fit the spectrum. In Figure 3.9 such a case is illustrated.

4.2 Metabolite Quantification by Superfit

In this study, the concentrations of NAAG, GPC, PC, and Gln were quantified with a MAPE $> 20\%$. Quantification of the corresponding concentrations is, therefore, difficult and not that reliable. The weakness of Superfit in quantifying those metabolites may be explained by their overlapping peaks. NAAG with its low amplitudes overlaps with NAA due to line broadening and has a naturally small amplitude. Gln overlaps with Glu, which has higher amplitudes compared to Gln [1]. Another consequence of the low amplitudes of NAAG and Gln is a low SNR, which also leads to higher MAPEs.

In general, GPC and PC are not separatible, because the acquired spectral resolution (especially *in vivo*) is too low to differentiate between both metabolites and, thus, their individual concentrations, as well as their ratio GPC/PC, are negligible. The same applies for PCr, Cr, and Cr/PCr, respectively. Instead, the total concentrations of tCho and tCr are examined.

NAA and m-Ins were quantified with a MAPE $< 10\%$. While m-Ins has a unique position and does not overlap with neighboring peaks, NAA naturally has a high amplitude [1]. Thus, both metabolites could be quantified very well compared to Glu, with a MAPE between 10% and 20%.

Examining total metabolic concentrations, Superfit quantified tNAA,

tCho, and tCr, with a MAPE $< 10\%$, which enabled the robust prediction of their concentrations. Also, Glx still had a MAPE $< 20\%$.

Only m-Ins and tCho had p-values > 0.05. However, considering the Bland-Altman plots, true and predicted concentrations of NAA, m-Ins, tNAA, tCho, and tCr correlate and, considering the respective MAPEs, the predicted concentrations were significantly accurate.

For all metabolites, extreme outliers influenced the MAPEs and, thus, the corresponding median APEs were calculated, which, in general, were lower than the MAPEs.

An interesting observation was the increasing MAPE of Glx with increasing FWHM. While the MAPEs of tNAA, tCho, tCr, and m-Ins decreased with increasing FWHM (with their high amplitudes still detectable at higher FWHM); the smaller peaks of the Gln and Glu multiplets might be too flat to be separated from the spectral baseline.

With regard to the relative concentration ratios, tCr/tNAA, tCho/tNAA, and m-Ins had MAPEs $< 10\%$, correlating to the corresponding MAPEs above. Glx had a MAPE $< 20\%$. For all four total metabolite concentrations, a mean MAPE of $8.99\% \pm 7.41\%$ was recorded. Significantly high MAPEs were recorded for NAAG/NAA and Gln/Glu, showing how Superfit struggled with the separation of overlapping peaks.

4.3 Conclusion

In summary, Superfit fit a broad range of spectra degraded by many factors and managed to disassemble them into their metabolic components. Although struggling with the separation of overlapping peaks, Superfit could sufficiently quantify the total metabolic concentrations of the major metabolites visible in brain MR: tNAA; tCho; tCr; m-Ins; and Glx. While Gurbani et al. dealt only with singlets, Superfit's approach using basis sets enabled the management of j-coupled metabolites as well [20]. In contrast to the CNN implemented by Lee et al., Superfit was trained by unsupervised learning [28]. Thus, Superfit can be trained without a dependence on ground truth.

The concept of Superfit could be expanded, modified and improved to quantify the metabolite concentrations more accurately. For example, during the training, prior knowledge could be used to detect outliers and penalize the loss; thus, the quantification results might more accurate. In

addition to that, the number of metabolites that are investigated could be expanded by including , e.g., lactate or GABA.

Furthermore, the concept and performance of Superfit could be examined under more realistic conditions by using only *in vivo* data and comparing them directly to the results of, e.g., LCModel.

As in other previous studies, Superfit also proved that sub-minute metabolite quantification in brain MRSI can be achieved using deep learning, a critical step toward implementing MRSI in the daily clinical routine.

Bibliography

[1] R.A. de Graaf. *In Vivo NMR Spectroscopy: Principles and Techniques*. Wiley, 2007. ISBN: 9780470026700. URL: https://books.google.at/books?id=ShBbuAAACAAJ.

[2] Stephen W. Provencher. "Estimation of metabolite concentrations from localized in vivo proton NMR spectra". In: *Magnetic Resonance in Medicine* 30.6 (1993), pp. 672–679. DOI: 10.1002/mrm.1910300604. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.1910300604. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.1910300604.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[4] Ullrich Mueller-Lisse and Michael Scherr. "1H magnetic resonance spectroscopy of the prostate". In: *Der Radiologe* 43 (July 2003), pp. 481–8.

[5] Kim M Cecil. "MR spectroscopy of metabolic disorders". In: *Neuroimaging Clinics* 16.1 (2006), pp. 87–116.

[6] Marinette Graaf. "In vivo magnetic resonance spectroscopy: Basic methodology and clinical applications". In: *European biophysics journal : EBJ* 39 (Sept. 2009), pp. 527–40. DOI: 10.1007/s00249-009-0517-y.

[7] Wolfgang Bogner et al. "In vivo quantification of intracerebral GABA by single-voxel (1)H-MRS-How reproducible are the results?" In: *European journal of radiology* 73 (Mar. 2009), pp. 526–31. DOI: 10.1016/j.ejrad.2009.01.014.

[8] Gilbert Hangel. *Accelerated High Resolution 3d MRSI in the brain at 7T*. Doctoral Thesis. Medical University of Vienna, 2015.

[9] Alex A. Bhogal et al. "1H–MRS processing parameters affect metabolite quantification: The urgent need for uniform and transparent standardization". In: *NMR in Biomedicine* 30.11 (2017). e3804 NBM-17-0072.R2, e3804. DOI: 10.1002/nbm.3804. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nbm.3804. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nbm.3804.

[10] Shih-Chung Lo et al. "Computer-Assisted Diagnosis of Lung Nodule Detection using Artificial Convolution Neural Network". In: *Proceedings of SPIE - The International Society for Optical Engineering* 1898 (Feb. 1993), pp. 859–869.

[11] Heang-Ping Chan et al. "Computer-aided detection of mammographic microcalcifications: Pattern recognition with an artificial neural network". In: *Medical physics* 22 (Nov. 1995), pp. 1555–67. DOI: 10.1118/1.597428.

[12] Berkman Sahiner et al. "Classification of mass and normal breast tissue: A convolution neural network classifier with spatial domain and texture images". In: *Medical Imaging, IEEE Transactions on* 15 (Nov. 1996), pp. 598–610. DOI: 10.1109/42.538937.

[13] Warren Mcculloch and Walter Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147.

[14] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6 (1958), pp. 386–408.

[15] Michael Nielsen. *Neural Networks and Deep Learning*. 2015. URL: https://http://neuralnetworksanddeeplearning.com/.

[16] Xavier Glorot and Y. Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research - Proceedings Track* 9 (Jan. 2010), pp. 249–256.

[17] MATLAB. *version 8.1.0.073 (R2013a)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[18] D. Stefan et al. “Quantitation of magnetic resonance spectroscopy signals: The jMRUI software package”. English. In: *Measurement, Science and Technology* 20.10 (2009). ISSN: 0957-0233. DOI: 10.1088/0957-0233/20/10/104035.

[19] Guido Van Rossum and Fred L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[20] Saumya S. Gurbani et al. “Incorporation of a spectral model in a convolutional neural network for accelerated spectral fitting”. In: *Magnetic Resonance in Medicine* 81.5 (2019), pp. 3346–3357. DOI: 10.1002/mrm.27641. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.27641. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.27641.

[21] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992. DOI: 10.1137/1.9781611970104. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611970104. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611970104.

[22] Martin Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: http://tensorflow.org/.

[23] J. Martin Bland and DouglasG. Altman. “STATISTICAL METHODS FOR ASSESSING AGREEMENT BETWEEN TWO METHODS OF CLINICAL MEASUREMENT”. In: *The Lancet* 327.8476 (1986). Originally published as Volume 1, Issue 8476, pp. 307–310. ISSN: 0140-6736. DOI: https://doi.org/10.1016/S0140-6736(86)90837-8. URL: http://www.sciencedirect.com/science/article/pii/S0140673686908378.

[24] Hélène Ratiney et al. “Time-domain quantitation of 1H short echo-time signals: Background accommodation”. In: *Magma (New York, N.Y.)* 16 (June 2004), pp. 284–96. DOI: 10.1007/s10334-004-0037-9.

[25] Sreenath P Kyathanahally, André Döring, and Roland Kreis. “Deep learning approaches for detection and removal of ghosting artifacts in MR spectroscopy: Detection and Removal of Ghosting Artifacts in MRS Using Deep Learning”. In: *Magnetic Resonance in Medicine* 80 (Feb. 2018). DOI: 10.1002/mrm.27096.

[26] Dhritiman Das et al. "Quantification of Metabolites in Magnetic Resonance Spectroscopic Imaging Using Machine Learning". In: Sept. 2017, pp. 462–470. ISBN: 978-3-319-66178-0. DOI: 10.1007/978-3-319-66179-7_53.

[27] Nima Hatami, Michaël Sdika, and Hélène Ratiney. "Magnetic Resonance Spectroscopy Quantification using Deep Learning". In: June 2018.

[28] Hyeong Hun Lee and Hyeonjin Kim. "Intact metabolite spectrum mining by deep learning in proton magnetic resonance spectroscopy of the brain". In: *Magnetic Resonance in Medicine* 82.1 (2019), pp. 33–48. DOI: 10.1002/mrm.27727. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrm.27727. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.27727.