Neural Network Based Data Estimation

Master’s Thesis
to confer the academic degree of
Diplom-Ingenieur
in the Master’s Program
Electronics and Information Technology
Statutory Declaration

I hereby declare that the thesis submitted is my own unaided work, that I have not used other than the sources indicated, and that all direct and indirect sources are acknowledged as references.

This printed thesis is identical with the electronic version submitted.

April 2, 2021
Date

[Signature]
Abstract

For data estimation and many other tasks in communication systems model-based methods have been employed for several decades. However, these methods also have some drawbacks. The optimal model-based methods are often extremely computationally expensive, while complexity reduced methods sometimes perform significantly worse than the optimal ones do. Further, available data cannot be incorporated into the estimation process. The incredible success of data-driven machine learning methods in many different application areas has recently led to investigations of the applicability of machine learning methods, mostly neural networks, for data estimation, but also for other tasks in communication engineering. These methods are expected to resolve the issues of model-based methods, but difficulties arise for the machine learning approaches as well.

In this work, neural network based data estimators are investigated for a communication system utilizing the so-called unique word orthogonal frequency division multiplexing (UW-OFDM) transmission scheme through simulations. More precisely, in the simulations three different neural network architectures are employed for data estimation in two communication systems with different system dimensions. For both systems two modulation alphabets and two distinct ways of generating UW-OFDM symbols are considered, and both channel coded and uncoded transmission is regarded. The neural network based estimators are compared with model-based methods, whereby the achieved bit error ratios at specific signal-to-noise-ratio ranges serve as a performance measure. Further, the impact of data pre-processing on the estimation performance of the neural network based data estimators is investigated. Finally, the distribution of their estimates is examined and a brief complexity analysis of the neural networks is conducted.
Acknowledgements

First and foremost, my thanks go to Univ.-Prof. Dr. Mario Huemer for giving me the opportunity to work on this incredibly interesting topic for my Master’s thesis and for a lot of valuable advice. I would like to take this opportunity to express my appreciation for his enthusiasm for his field of research and his great teaching style, which first aroused my interest in this discipline.

I owe my deepest gratitude to Dr. Gergő Bognár for many interesting ideas and fruitful discussions, for sharing some of his vast mathematical knowledge with me and for fixing server problems even on weekends. I especially appreciate his quick responses to all my queries at any time. It was a pleasure to work with you!

Further, I would also like to thank Dr. Oliver Lang for his helpfulness with technical, but also organizational problems and many nice talks.

Additionally, I express my gratitude to Dr. Christian Hofbauer for providing the simulation framework of his dissertation on unique word OFDM, on which I could base my simulation framework.

Finally, I thank my family for their never-ending and unconditional support in all my endeavors.
# Contents

1. **Introduction**  
   1.1. Outline .................................................. 2  
   1.2. Notation ................................................ 2  

2. **Theoretical Background**  
   2.1. Unique Word OFDM ........................................ 5  
   2.1.1. System Model ........................................ 5  
   2.1.2. Transmission Chain .................................. 11  
   2.2. Traditional Estimation Methods ......................... 15  
   2.2.1. Estimation Theory ................................... 15  
   2.2.2. Optimal Estimators for Data Estimation ............. 19  
   2.2.3. Linear Estimators for Data Estimation .............. 27  
   2.2.4. Decision Feedback Equalizer ......................... 31  
   2.2.5. Decision Boundaries of Traditional Data Estimators .. 32  
   2.2.6. Channel Estimation ................................ 35  
   2.3. Neural Networks ......................................... 36  
   2.3.1. Feedforward Neural Networks ......................... 37  
   2.3.2. Training and Testing of Neural Networks ............ 39  
   2.3.3. Deep Unfolding ..................................... 41  
   2.3.4. Self-Attention and Transformers ..................... 42  
   2.3.5. Log-Likelihood Ratios for Neural Network Based Data Estimators 44  

3. **Neural Network Based Data Estimation for UW-OFDM**  
   3.1. Inputs and Outputs of Neural Network based Data Estimators .... 49  
   3.2. DetNet .................................................. 50  
   3.3. FCNN Detector .......................................... 53  
   3.4. Attention Detector ..................................... 55  
   3.5. Training Methodology ................................... 57  
   3.5.1. Data Normalization ................................ 57  
   3.5.2. Pre-Training ........................................ 59  
   3.5.3. SNR for Training .................................... 59  
   3.5.4. Training Set Size and Deep-Fading Training Channels ........ 60  
   3.5.5. General Training Settings ........................... 61  

4. **Results**  
   4.1. Simulation Framework ................................. 63  
   4.2. Perfect Channel Knowledge ........................... 65  
   4.2.1. Systematic UW-OFDM .............................. 65
4.2.2. Non-Systematic UW-OFDM ........................................ 67
4.3. Imperfect Channel Knowledge ....................................... 74
4.4. Impact of Pre-Processing ........................................... 75
4.5. Constellation Diagrams of the Data Symbol Estimates ........... 76
4.6. Complexity Analysis .................................................. 80

5. Conclusion ................................................................. 87

A. Hyperparameter Settings ............................................... 89
1. Introduction

The aim of data estimation, also termed equalization, is to reconstruct the transmitted data, that have been heavily disturbed during transmission. This is essential for communications in modern electronic-based systems (EBS) such as smartphones, sensor networks and connected cars. There are several well-understood model-based methods, in this work also referred to as traditional methods, to accomplish this task, such as the maximum likelihood (ML) estimator, the maximum-a-posteriori (MAP) estimator, the minimum mean square error (MMSE) estimator, the linear minimum mean square error (LMMSE) estimator, or the least squares (LS) estimator, to name just a few. However, optimal model-based methods are often computationally highly demanding or even infeasible. To remedy this obstacle, constraints are introduced that lead to lower computational complexity, but also to suboptimal, sometimes far worse estimation performance. In addition, the performance of model-based methods might severely degrade in case of insufficient model knowledge, and possibly available data cannot be incorporated into the estimation process. In the last years, neural networks (NNs) have become state-of-the-art to solve tasks in many different application areas. However, these data-driven methods are hardly employed in communication engineering, since incorporating domain knowledge into NNs is very difficult. Recently, efforts have been made to achieve this and thus a few approaches for neural network based data estimation have already been published, which show promising results and give hope that some of the issues of model-based data estimators can be resolved by NNs [1–5].

As already mentioned, data transmitted over a so-called channel of a communication system from the transmitter to the receiver device(s), are disturbed such that the received data are usually highly erroneous. Sources of disturbance are e.g. thermal noise and shot noise in the communication devices, but for wireless communications also any kind of obstacles that prevent undisturbed propagation of the transmitted signal (in form of an electromagnetic wave), leading to interference between successively transmitted data. These disturbances can be modeled, the former usually by additive white Gaussian noise (AWGN) and the latter as a multipath channel. The actual model, which is employed by the model-based data estimators to reconstruct the transmitted data, depends also on the communication system used, but many of the models have a similar structure such that the methods for data estimation can be applied to most of them. In this work, neural network based data estimation is investigated for a communication system utilizing the so-called unique word orthogonal frequency division multiplexing (UW-OFDM) transmission scheme [6–11], an alternative to the very popular cyclic prefix (CP) OFDM [12,13], which is used in the majority of current digital communication standards [14,15]. A detailed introduction to UW-OFDM is given in Section 2.1.
In this work, it is assumed that data from example transmissions, where both the transmitted and the received data are known, is available for all communication system settings studied. This is not a serious limitation for real-world applications, as for a given problem known data can be transmitted and the corresponding received values stored. Hence, data estimation can also be treated as a supervised machine learning task, that could be solved e.g. with NNs. The NNs are trained with the available data and should then be able to accurately estimate the transmitted data for different channels, that follow the statistics of the channels with which the NNs were trained. In this thesis, NN based data estimators are investigated and novel approaches are proposed.

1.1. Outline

The outline of this thesis is as follows. In Chapter 2, theoretical aspects of UW-OFDM, model-based estimators, as well as neural networks are given. For all of these very comprehensive topics only the theory that is important for this work is presented, since a general description is far beyond the scope of this thesis. Furthermore, in Chapter 3 the three different NN architectures employed for data estimation in this work are detailed. One of them, namely the DetNet, was proposed in [2] for a general multiple input multiple output (MIMO) communication system. However, without any further modification, the DetNet does not perform well for the UW-OFDM system. The reasons for this issue are investigated and the countermeasures found are described in Chapter 3, too. Moreover, two other NN architectures are proposed in this chapter, which, to the best of my knowledge, are novel methods for data estimation, but are based on well-known NNs. The found results are then presented in Chapter 4, which includes a performance evaluation of the examined NNs, an investigation of the distribution of their estimates and a brief analysis of their computational complexity. Finally, in Chapter 5 a summary of this work, as well as an outlook on possible research on this topic, is given.

1.2. Notation

In this work, bold face lower case letters \( \mathbf{x} \) and bold face upper case letters \( \mathbf{X} \) stand for vectors and matrices, respectively. If a variable is described in both time and frequency domain, they are distinguished by using a tilde for the frequency domain representation, i.e. applying the Fourier transform to \( \mathbf{x} \) yields \( \tilde{\mathbf{x}} \). In case that a variable is used only in time or frequency domain, the tilde is omitted. The \( i \)-th element of a vector is represented by \( x_i \), while for the element in the \( i \)-th row and \( j \)-th column of a matrix \( [\mathbf{X}]_{ij} \) is used. Further, the sets of real and complex numbers are denoted as \( \mathbb{R} \) and \( \mathbb{C} \), the transposition of a matrix or a vector as \( (\cdot)^T \), the conjugate as \( (\cdot)^* \) and the conjugate transposition, also termed Hermitian, as \( (\cdot)^H \). For the identity matrix \( \mathbf{I} \) is written and the real and the imaginary part of a complex scalar value or complex vector are obtained by applying \( \text{Re}\{\cdot\} \) and \( \text{Im}\{\cdot\} \) to the complex quantity, respectively. Moreover,
the probability density function (PDF) of a continuous random variable or vector is denoted as \( p(\cdot) \), the probability mass function (PMF) of a discrete random variable or vector as \( p[\cdot] \), the probability that an event occurs as \( \Pr(\cdot) \) and the expectation operator as \( E_{\cdot} \), whereby the the subscript indicates the random variable over whose PDF or PMF is averaged. In case that the averaging PDF/PMF is clear from context, the subscript is omitted. Finally, an estimate is indicated by a hat, i.e. \( \hat{x} \) is the estimate of the unknown vector \( x \).
2. Theoretical Background

2.1. Unique Word OFDM

In this section, the UW-OFDM signaling scheme is introduced and its linear system model, which plays a central role in the whole thesis, is derived. Finally, the components of the transmission chain are presented. The whole section is a summary of the corresponding parts in [11] and [10], to which is also referred for further details.

2.1.1. System Model

In multi-carrier systems, the data transmission happens in a block-wise manner, i.e. a block of \(N\) data symbols is transmitted simultaneously, whereby the data symbols are placed adjacently in frequency domain using distinct subcarriers. In order to be able to easily separate the symbols on the receiver side again, in OFDM systems the \(N\) subcarriers are chosen to be complex exponentials, defined as

\[
\mathbf{f}_n = \frac{1}{N} \begin{bmatrix}
  e^{j \frac{2\pi}{N} 0 n} \\
  e^{j \frac{2\pi}{N} 1 n} \\
  \vdots \\
  e^{j \frac{2\pi}{N} k n} \\
  \vdots \\
  e^{j \frac{2\pi}{N} (N-1)n}
\end{bmatrix}, \quad n = 0, \ldots, N - 1, \tag{2.1}
\]

which are orthogonal over the symbol interval of \(N\) samples due to the frequency spacing of adjacent subcarriers of \(\Delta f = \frac{2\pi}{N}\).

For data transmission, the orthogonal subcarriers are modulated by a block of data symbols \(\mathbf{x}\) and thus the transmit signal in time domain is

\[
\mathbf{x} = \sum_{n=0}^{N-1} \mathbf{f}_n \tilde{x}_n = \mathbf{F}_N^{-1} \tilde{\mathbf{x}}, \tag{2.2}
\]
2. Theoretical Background

where $F^{-1}_N$ is the inverse discrete Fourier transform (IDFT) matrix

$$F^{-1}_N = \frac{1}{N} \begin{bmatrix}
1 & 1 & \cdots & 1 \\
1 & e^{j\frac{2\pi}{N}} & \cdots & e^{j\frac{2\pi}{N}(N-1)} \\
\vdots & \vdots & \ddots & \vdots \\
1 & e^{j\frac{2\pi}{N}(N-1)} & \cdots & e^{j\frac{2\pi}{N}(N-1)(N-1)}
\end{bmatrix} = [f_0 \ f_1 \ \cdots \ f_{N-1}].$$

The transmit signal is transmitted over a dispersive propagation channel, modeled as multipath channel, which is in turn described by its channel impulse response (CIR). Multipath propagation in general leads to interference between OFDM symbols, which is termed inter-symbol interference (ISI) or inter-block interference (IBI). In order to avoid this, guard intervals of length $N_g$ are introduced, which separate consecutively transmitted OFDM symbols and should be long enough, such that the transient caused by one OFDM symbol is subsided until the following OFDM symbol is transmitted. The most popular choice is to realize the guard interval in form of a cyclic prefix, i.e. to append in front of an OFDM symbol in time domain a copy of its last $N_g$ samples. That is, the guard interval is a cyclic repetition of the data and it can be shown that this leads to a very low complex equalization in frequency domain, which is also the main motivation to use cyclic prefixes as guard intervals. However, the utilization of a CP in the guard interval wastes transmit energy and, as the CP does not provide any further information, also transmit time. This leads to the concept of UW-OFDM, where a deterministic sequence, called the unique word (UW), is employed as a guard interval. With this approach an improved bit error ratio (BER) compared to CP-OFDM is achievable. Further, the deterministic UW can be optimally designed for synchronization purposes or channel estimation. The drawback of the UW-OFDM signaling scheme is, however, that far more advanced and complex equalization methods, on which also the focus of the whole work lies, are needed. In the following, the system model of the UW-OFDM transmission scheme is derived.

First of all, the transmit symbol in time domain is defined as

$$x' = \begin{bmatrix} x_p \\ x_u \end{bmatrix},$$

where $x_p \in \mathbb{C}^{N-N_u}$ is the payload of the transmitted symbol and $x_u \in \mathbb{C}^{N_u}$ is a predefined, deterministic sequence, the unique word. As the UW acts as the guard interval, the length of the UW $N_u = N_g$ has to be chosen according to the expected length of the CIR to prevent ISI. In the UW-OFDM concept presented in [6–11], the transmit symbol in time domain $x'$ is assembled in two steps, namely, the generation of the symbol vector

$$x = \begin{bmatrix} x_p \\ 0 \end{bmatrix}$$

1 Additionally, the transmitted signal is disturbed by noise that is usually assumed to be white Gaussian noise (WGN), which distorts every sample individually. Hence, it does not introduce interference between OFDM symbols and is not considered for now.
as stated in (2.2) as result of an IDFT in a first step and the addition of the UW in a second step:

\[ x' = x + \begin{bmatrix} 0 \\ x_u \end{bmatrix}. \tag{2.6} \]

The structure of \( x \) already highlights the difference between UW-OFDM and CP-OFDM. In CP-OFDM, the OFDM time domain symbol is generated according to (2.2) and \( \tilde{x} = Bd \), whereby \( B \) is a matrix which inserts zero sub-carriers at DC and the band edges to meet spectral requirements, and \( d \) is the data vector with data symbols, typically drawn from a QAM (quadrature amplitude modulation), PSK (phase shift keying), or ASK (amplitude shift keying) alphabet, but does not have to fulfill any further constraints. The CP is appended before transmission in front of every OFDM time domain symbol of length \( N \). However, in UW-OFDM the OFDM time domain symbol must have the structure shown in (2.5), i.e. the UW-OFDM time domain symbol of length \( N \) contains zeros at the last \( N_u \) entries, which clearly introduces constraints on \( \tilde{x} \). The difference of the transmit sequences of the two OFDM signaling schemes is also pointed out in Figure 2.1.

![Figure 2.1.: Structure of transmit sequence in UW-OFDM and CP-OFDM.](image)

As already mentioned, for the UW-OFDM symbol generation the requirement

\[ F_N^{-1} \tilde{x} = \begin{bmatrix} x_p \\ 0 \end{bmatrix} \tag{2.7} \]

has to be fulfilled. Consequently, not the whole UW-OFDM symbol \( \tilde{x} \) can be chosen freely, but at least \( N_u \) redundant subcarriers are needed. In this work, the number of redundant subcarriers \( N_r \) is set equal to the length of the UW \( N_u \). That is, a UW-OFDM symbol can be expressed as

\[ \tilde{x} = BP \begin{bmatrix} d \\ r \end{bmatrix}, \tag{2.8} \]

where \( B \in \{0, 1\}^{N \times (N_d + N_r)} \) optionally inserts, like for CP-OFDM, \( N_z \) zero subcarriers (one at DC and the remaining at the band edges), with \( N = N_d + N_r + N_z \). For now, \( P \in \{0, 1\}^{(N_d + N_r) \times (N_d + N_r)} \) is assumed to be a permutation matrix, which places the data symbols drawn from the modulation alphabet \( S \), contained in \( d \in S^{N_d} \subset \mathbb{C}^{N_d} \), and the redundant values, comprised in \( r \in \mathbb{C}^{N_r} \), on the intended subcarrier positions. This approach produces the so-called systematic generation of UW-OFDM symbols. Another method, which is described later in this section, is termed non-systematic generation.
2. Theoretical Background

There, $P$ is replaced by an invertible full matrix. From (2.7) and (2.8) it can be seen that in UW-OFDM only the $N_d$ data symbols of $d$ can be selected freely from the utilized alphabet $S$, while the remaining $N_r$ values have to be chosen properly to obtain the UW-OFDM time domain symbol $x$. Inserting (2.8) into (2.7) leads to

$$
\begin{align*}
F_N^{-1}BP^M \begin{bmatrix}
d \\
r
\end{bmatrix} &= \begin{bmatrix}
M_{11} & M_{12} \\
M_{21} & M_{22}
\end{bmatrix} \begin{bmatrix}
r
\end{bmatrix} = \begin{bmatrix}
x_p \\
0
\end{bmatrix},
\end{align*}
$$

(2.9)

with the sub-matrices $M_{11} \in \mathbb{C}^{(N_d+N_r) \times N_d}$, $M_{12} \in \mathbb{C}^{(N_d+N_r) \times N_u}$, $M_{21} \in \mathbb{C}^{N_u \times N_d}$ and $M_{22} \in \mathbb{C}^{N_u \times N_u}$. To fulfill (2.9), the relation

$$
M_{21}d + M_{22}r = 0
$$

(2.10)

must hold. Consequently, $r$ has to be chosen according to

$$
r = -M_{22}^{-1}M_{21}^T d = Td.
$$

(2.11)

As $r$ is a linear mapping of the data vector $d$ and thus does not comprise any new information, the term redundant values for the elements of $r$ is justified. However, these redundant values could be exploited on the receiver side to improve the performance of the data estimation. The obtained relation between $d$ and $r$ can be incorporated into (2.8), which yields

$$
\tilde{x} = BP \begin{bmatrix}
d \\
Td
\end{bmatrix} = BP \begin{bmatrix}
I \\
G
\end{bmatrix} d = BGd ,
$$

(2.12)

where $G \in \mathbb{C}^{(N_d+N_r) \times N_d}$ is referred to as the UW-OFDM generator matrix.

The time domain transmit symbol is then created by adding the UW to the UW-OFDM time domain symbol $x$, which can be expressed as

$$
x' = F_N^{-1}BGd + \begin{bmatrix}
0 \\
x_u
\end{bmatrix} = F_N^{-1}(BGd + \tilde{x}_u),
$$

(2.13)

where $\tilde{x}_u = F_N \begin{bmatrix}
0 \\
x_u
\end{bmatrix}$ is the frequency domain influence of the UW on the transmit symbol. Note, that to the transmit signal also some pilot symbols can be added, e.g. for synchronization purposes, at the cost of the number of data symbols per UW-OFDM symbols. For a further elaboration on how to add the pilot symbols properly it is referred to [11].

As mentioned above, after assembling the transmit symbol $x'$ the transmission over a multipath channel, modeled by a tapped delay line, whereby its coefficients are the CIR, follows. In addition to the distortion by the multipath propagation, the transmitted sequence is assumed to be disturbed by circularly symmetric complex white Gaussian
noise (WGN). Therefore, the received signal $y_r[k]$ can generally be written as

$$y_r[k] = x'[k] * h[k] + n[k] = \sum_{i=0}^{\infty} h[i] x'[k-i] + n[k],$$

(2.14)

where "\(*\)" denotes the linear convolution operator, $x'[k]$ the transmitted signal, $h[k]$ the CIR and $n[k]$ the circularly symmetric complex WGN with the variance $\sigma_n^2$. The transmitted signal $x'[k]$ is a concatenation of all the successively transmitted transmit symbols $x'[l][k]$, $k = 0, ..., N - 1$, as shown in Figure 2.1a, where $x'[l][k]$ is the $k$-th entry of the $l$-th transmit symbol $x[l]$. Due to the special structure of the transmit sequence, i.e. one block of payload is surrounded by UWs, it can be shown [10,11] that the effect of the channel on the transmitted sequence can be regarded for each UW-OFDM symbol individually, whereby it is assumed that the length of the CIR is at most $N_g + 1$. However, the linear convolution has to be replaced by a cyclic convolution, which leads to the description of the $l$-th received UW-OFDM symbol as

$$y_{l}[k] = x'[l][k] \circledast h[k] + n[l][k] = \sum_{i=0}^{N-1} x'[l][i] h[(k-i)\mod N] + n[l][k].$$

(2.15)

The latter equation can be expressed easily in vector-matrix notation (neglecting the symbol index $l$) as

$$y_r = H_c x' + n = H_c F_N^{-1} (B G d + \tilde{x}_u) + n,$$

(2.16)

with $n \sim CN(0, \sigma_n^2 I)$ and the cyclic convolution matrix

$$H_c = \begin{bmatrix}
  h_0 & 0 & \cdots & 0 & h_{N_g} & \cdots & h_1 \\
  h_1 & h_0 & \cdots & \vdots & \ddots & \ddots & \vdots \\
  \vdots & h_0 & \ddots & \ddots & \ddots & \ddots & \vdots \\
  h_{N_g} & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
  0 & \cdots & \ddots & \ddots & \ddots & \ddots & \vdots \\
  \vdots & \cdots & \ddots & \ddots & \ddots & \ddots & 0 \\
  0 & \cdots & 0 & h_{N_g} & \cdots & \cdots & h_0 \\
\end{bmatrix} \in \mathbb{C}^{N \times N},$$

where $h_0, ..., h_{N_g}$ are the coefficients of the CIR $h$.

On receiver side, the received vector is transformed to frequency domain by applying the DFT on $y_r$, and the zero subcarriers are excluded by multiplying by $B^T$, leading to the downsized vector

$$y_d = B^T F_N y_r = B^T F_N H_c F_N^{-1} (B G d + \tilde{x}_u) + B^T F_N n,$$

(2.17)

Since any circulant matrix is diagonalized by the DFT matrix, $\tilde{H}_c = F_N H_c F_N^{-1} \in \mathbb{C}^{N \times N}$ is a diagonal matrix, whereby the main diagonal contains the sampled channel frequency
response, i.e. \( \tilde{H}_c = \text{diag} \left( F_N \begin{bmatrix} h \ 0 \end{bmatrix} \right) \). Inserting \( \tilde{H}_c \) into (2.17) yields

\[
y_d = B^T \tilde{H}_c (BGd + \tilde{x}_u) + B^T F_N n .
\] (2.18)

As \( \tilde{H}_c \) is a diagonal matrix (so only the \( i \)-th element of the UW \( \tilde{x}_{ui} \) has an influence on \( y_{di} \)) and the zero subcarriers are removed to obtain \( y_d \), \( \tilde{x}_u \) can be multiplied by \( BB^T \) without altering the result. The multiplication of \( \tilde{x}_u \) by \( BB^T \) actually removes all elements in \( \tilde{x}_u \) at positions of zero subcarriers by multiplying by \( B^T \) and inserts zeros at those positions again. Hence, (2.18) can be reformulated as

\[
y_d = B^T \tilde{H}_c B (Gd + B^T \tilde{x}_u) + B^T F_N n = \tilde{H} Gd + \tilde{H} B^T \tilde{x}_u + w ,
\] (2.19)

where \( \tilde{H} = B^T \tilde{H}_c B \in \mathbb{C}^{(N_d+N_r) \times (N_d+N_r)} \) contains the sampled channel frequency response without the coefficients at positions corresponding to the zero subcarriers. Further, \( w \sim \mathcal{CN} \left( 0, N \sigma_n^2 I \right) \in \mathbb{C}^{N_d+N_r} \) is, like its counterpart in time domain \( n \), circularly symmetric complex WGN, i.e. all its elements \( w_i \) are distributed according to \( \text{Re}\{w_i\} \sim \mathcal{N} \left( 0, \frac{N \sigma_n^2}{2} \right) \) and \( \text{Im}\{w_i\} \sim \mathcal{N} \left( 0, \frac{N \sigma_n^2}{2} \right) \).

Finally, the influence of the UW is removed from \( y_d \) to obtain

\[
y = y_d - \tilde{H} B^T \tilde{x}_u = H d + w ,
\] (2.20)

where \( H = \tilde{H} G \). That is, the UW-OFDM transmission can be described by a linear model, which serves as a starting point for the subsequent data estimation.

As already mentioned, the generator matrix could also have the form

\[
G = A \begin{bmatrix} I \\ T \end{bmatrix} ,
\] (2.21)

where \( A \in \mathbb{R}^{(N_d+N_r) \times (N_d+N_r)} \) is a real valued, non-singular matrix [6,8]. \( A \) can be determined by minimizing a cost function \( J(A) = \text{tr}(C_{ee}) \) at a fixed SNR value, with the error covariance matrix \( C_{ee} \) of the data estimation error \( e = \hat{d} - d \), where \( \hat{d} \) is the estimate of \( d \). Further, the restriction

\[
F_N^{-1} B G d = \begin{bmatrix} x_p \\ 0 \end{bmatrix}
\] (2.22)

still has to be fulfilled, which leads to a constrained optimization problem. This approach leads to the so-called non-systematic generation of UW-OFDM symbols. In [11] and [10] the matrix \( A \) is optimized for linear data estimators, namely the LMMSE or the BLUE estimator, and for AWGN conditions. The generator matrix \( G \) optimized with respect to the LMMSE data estimator is also employed for non-systematic UW-OFDM throughout this work. In non-systematic UW-OFDM, the redundancy is distributed over the whole UW-OFDM symbol. Consequently, no clear distinction between data and redundant subcarriers can be made any longer, as it is the case for systematic UW-OFDM. That is,
2.1. Unique Word OFDM

while in systematic UW-OFDM only the data symbols drawn from the symbol alphabet $S$ are present at the position of data subcarriers, in non-systematic UW-OFDM the data symbol is superimposed by a weighted sum of the remaining data in the UW-OFDM symbol. Purely redundant subcarriers, however, still exist. This method tremendously improves the system performance.

Apart from a different generator matrix $G$ the system model of non-systematic UW-OFDM is equivalent to that of systematic UW-OFDM given in (2.20). The methods presented in this work are applicable for systematic and non-systematic generator matrices unless stated otherwise.

2.1.2. Transmission Chain

In the following the whole UW-OFDM transmission chain with its components is presented briefly. The block diagram of the transmission chain is depicted in Figure 2.2. Further, the signal vectors after each building block, which are also mentioned in the derivation of the system model above, are shown in this figure to establish a connection to the mathematical description of the transmission system.

![Figure 2.2. Transmission chain of the UW-OFDM system.](image)

First of all, the binary transmission data are assumed to be independent and identically distributed (iid) and ‘0’s and ‘1’s are equiprobable. The data transmission can be performed either with or without channel coding, both cases are covered in this work. As the channel coding and the subsequent interleaving is optional, both building blocks are drawn with dashed lines in Figure 2.2.

For the channel coding a convolutional encoder with coding rate $r = 1/2$, constraint length 7 and generator polynomials $(133, 171)_8$ is employed, which is also the code utilized in the IEEE 802.11a standard [16].

In frequency-selective fading channels some subcarriers might be attenuated heavily. The attenuation coefficients of neighboring subcarriers are correlated and can lead to burst errors, which in turn may severely degrade the performance of the channel code. In order to combat this, the encoded bits have to be repositioned such that originally successive bits are separated by more positions than the constraint length of the code.
A block interleaver of block size $L$ is a memory block with $K$ rows and $L/K$ columns. The encoded bits are written to the interleaver row-wise and read out column-wise. This leads to a spread of originally adjacent bits by the interleaving factor $K$, which has to be chosen appropriately. In this work, the block length is always the number of bits in an UW-OFDM symbol, whereas the maximum interleaving distance is smaller than the length of an UW-OFDM symbol.

The binary data, which are either directly the input sequence (uncoded transmission) or come from the interleaver (coded transmission) are mapped to data symbols from the utilized alphabet, which in this work is mainly QPSK (quadrature phase shift keying), and for some simulation cases BPSK (binary phase shift keying). The mapping of two binary bits $b_1b_0$ to the symbols of the QPSK modulation alphabet $S$ is performed according to

$$
11 \rightarrow \rho(1 + j), \quad 01 \rightarrow \rho(1 - j), \quad 10 \rightarrow \rho(-1 + j), \quad 00 \rightarrow \rho(-1 - j),
$$

(2.23)

where $\rho = \frac{1}{\sqrt{2}}$ for a normalized alphabet with unit symbol variance, or $\rho = 1$ otherwise. In this work, always a normalized alphabet is utilized. Furthermore, the bit to symbol mapping for a BPSK alphabet $S$ is defined in this work as

$$
0 \rightarrow -1, \quad 1 \rightarrow 1.
$$

(2.24)

Then, the UW-OFDM symbols are generated from a data vector consisting of $N_d$ data symbols according to (2.12), whereby the structure of the generator matrix $G$ depends on whether systematic or non-systematic UW-OFDM generation is conducted. In this step also optional zero subcarriers at DC and the band edges are inserted into the UW-OFDM symbol.

Afterwards, the UW-OFDM time domain symbol is computed using the IDFT, which is in practice usually implemented by the inverse fast Fourier transform (IFFT) algorithm.

In a last step before transmission, the UW is added to the UW-OFDM time domain symbol. There exist many possibilities for the realization of the UW sequence and a non-zero UW could be employed e.g. for synchronization purposes. In this work, however, the focus lies on data estimation, therefore and for simplicity a sequence of zeros is chosen as a UW.

As already mentioned, the transmit sequence is a concatenation of numerous transmit symbols, which is also visualized in Figure 2.1a. For the practically important simulation scenario, where the CIR is not known, but has to be estimated, a preamble is appended in front of each transmit burst for the purpose of channel estimation. For further information on channel estimation and the exact realization of the preamble it is referred to Section 2.2.6. The channel is estimated for every transmit burst and is assumed to be time-invariant within the duration of the burst. In a real-world scenario the duration of a burst has to be chosen accordingly, such that this assumption is at least approximately valid. For the simulations in this work the length of a burst is chosen to
be 1000 UW-OFDM symbols.

The generated sequence is then transmitted over a multipath channel and is additionally disturbed by AWGN. In this work, the multipath propagation model from [17] is used, which describes the multipath propagation by the discrete linear convolution of the transmit signal by the sampled, discrete CIR of the channel. The CIR of length \(N_h\) is specified by its coefficients \(h_k\), where \(k = 0, \ldots, N_h - 1\) is the sampling index corresponding to the sampling time instants \(\tau_k = kT_s\). \(T_s\) is the sampling time of the UW-OFDM system, which is chosen like in [11] to be \(T_s = 50\) ns. According to [17], the CIR coefficients \(h_k\) follow an exponentially decaying power delay profile \(P(\tau_k)\) and are modeled as a circularly symmetric complex white Gaussian random variable with the probability density function \(h_k \sim \mathcal{CN}(0, \sigma_k^2)\), hence \(\text{Re}\{h_k\}, \text{Im}\{h_k\} \sim \mathcal{N}(0, \sigma_k^2)\). Their variances decay exponentially with

\[
\sigma_k^2 = P(\tau_k) = \sigma_0^2 e^{-k \frac{\tau_RMS}{\tau_k}} ,
\]

where

\[
\sigma_0^2 = 1 - e^{-\frac{\tau_RMS}{\tau_k}}
\]

is chosen such that the condition

\[
\sum_{k=0}^{N_h-1} \sigma_k^2 = 1
\]

is satisfied. The root mean square delay spread of the channel \(\tau_{RMS}\) is a measure for the time spread of a signal due to the transmission over the channel and is the square root of the second central moment of \(P(\tau_k)\):

\[
\tau_{RMS} = \sqrt{\frac{\sum_{k=0}^{N_h-1} P(\tau_k) \tau_k^2 - \left(\sum_{k=0}^{N_h-1} P(\tau_k) \tau_k\right)^2}{\sum_{k=0}^{N_h-1} P(\tau_k)}}.
\]

In this work, \(\tau_{RMS} = 100\) ns is utilized, this choice has also been made in [11]. Due to the exponentially decaying variances of the CIR coefficients the CIR can be assumed to be 0 after \(N_h\) taps. According to [17], \(N_h\) has to be chosen as

\[
N_h = 10 \frac{\tau_{RMS}}{T_s}.
\]

The WGN, more specifically circularly symmetric complex WGN with variance \(\sigma_n^2\), is added on receiver side. The performance of data estimators in this work is compared in terms of the BER they achieve over a specified signal-to-noise-ratio (SNR) range. As an SNR measure serves the ratio \(E_b/N_0\) measured at the receiver input, where \(E_b\) is the average energy per information bit and \(N_0\) is the one-sided power spectral density of the
noise. According to [11], the variance $\sigma_n^2$ of the complex noise is given by

$$
\sigma_n^2 = \frac{NP_s}{2(E_b/N_0)RN_b},
$$

(2.30)

where $N$ is the length of an UW-OFDM symbol in time domain, $R$ the coding rate, $N_b = N_d \cdot N_{bps}$ the number of bits per UW-OFDM symbol, $N_{bps}$ the number of bits per data symbol and $P_s = \frac{E_{y'}}{N}$ the average power per sample, whereby $E_{y'} = E[|Hc|\frac{x}{||x||}]$ denotes the mean symbol energy of an UW-OFDM time domain symbol on receiver side without AWGN.

On receiver side, the received UW-OFDM time domain symbol is transformed to frequency domain by applying the DFT, which is usually again implemented by the fast Fourier transform (FFT) algorithm. Further, the optionally inserted zero subcarriers are removed right after the DFT.

In the next step, the influence of the UW on the received UW-OFDM symbol is removed according to (2.20). The obtained vector is employed for the data estimation.

As already mentioned, for UW-OFDM systems the data estimation, also known as equalization, is - due to the full rectangular matrix $H$ in the system model (2.20) - far more complex than for CP-OFDM, where $H$ is a square diagonal matrix. This also motivates the need of this work, which is focused on this part of the transmission chain. The traditional methods for data estimation are further detailed in Section 2.2, while neural network based data estimators are described in Chapter 3.

The demapping of the estimated data symbols differs for the uncoded and the coded case. In the uncoded case, the data symbol estimates $\hat{d}_k \in \mathbb{C}$, $k = 0, ..., N_d - 1$, are mapped to the symbol $s_j \in \mathbb{S}$, $j = 1, ..., |\mathbb{S}|$, with the smallest Euclidean distance, which is also called hard decision. The performance of the Viterbi algorithm [18], which is employed for the channel decoding, benefits from being provided with reliability information for the estimates. Therefore, the reliability information in form of the so-called log-likelihood ratios is computed for every bit of the binary representation of the data stream, which is also known as soft decision. The log-likelihood ratio for the $j$-th bit of the binary representation of the $i$-th symbol $b_{ji}$ is defined as

$$
L_{ji} = \ln \left( \frac{Pr(b_{ji} = 1|y)}{Pr(b_{ji} = 0|y)} \right),
$$

(2.31)

where $j = 0, ..., |\mathbb{S}| - 1$, $i = 0, ..., N_d - 1$. The final expression of the log-likelihood ratio differs for every estimator and is therefore provided for all traditional estimators that are employed in the coded case in the sections where they are introduced, and for the neural network based data estimators in Section 2.3.5.

Optionally, for coded transmission, the deinterleaving follows the data symbol demapping. The deinterleaver performs the inverse operation of the interleaver. Hence, the demapped data sequence is written to the block deinterleaver, consisting of $K$ rows
and $L/K$ columns, column-wise and read out row-wise, which restores the original data order.

The channel decoding is carried out by a Viterbi decoder [18], which is a standard procedure for convolutional channel codes. As already mentioned, the Viterbi algorithm is provided with soft-information, which tremendously increases its performance.

## 2.2. Traditional Estimation Methods

### 2.2.1. Estimation Theory

In parameter estimation [19], the values of a set of unknown parameters should be determined based on the available data, usually in form of (noisy) measurements. This problem can mathematically be formulated as [19]

$$
\hat{\theta} = g(y_0, \ldots, y_{M-1}),
$$

(2.32)

where $\hat{\theta} \in \mathbb{C}^N$ denotes the estimate of the unknown parameter, $y_0, \ldots, y_{M-1} \in \mathbb{C}$ the available data and $g : \mathbb{C}^M \mapsto \mathbb{C}^N$ the, in general, multidimensional estimator function, which will be specified later on. In order to determine a well working estimator, the relation of the data and the parameter(s) to be estimated has to be modeled accurately. In this work, only the linear model

$$
y = H\theta + w
$$

(2.33)

has to be considered, where $y \in \mathbb{C}^M$, $H \in \mathbb{C}^{M \times N}$, $w \sim \mathcal{CN}(0, \sigma_n^2 I)$ and $\theta \in \mathbb{C}^N$ is the parameter vector to be estimated.

Depending on the assumptions made about $\theta$, estimation can be categorized as follows:

- In **classical** estimation, no so-called prior knowledge, i.e. statistical information about $\theta$ without any available data, is considered. Hence, $\theta$ is treated as deterministic, but unknown. The data vector $y$ is described by a probability density function (PDF) $p(y; \theta)$, parameterized by $\theta$.

- In **Bayesian** estimation, $\theta$ is treated as a random variable and its particular realization, that leads to the observed data, has to be estimated. In this work, prior knowledge is assumed to be known in form of the PDF $p(\theta)$. The relationship between the joint PDF $p(y, \theta)$, the conditional PDF $p(y|\theta)$ and the prior PDF $p(\theta)$ is given by Bayes’ theorem: $p(y, \theta) = p(y|\theta)p(\theta)$.

In the following, important concepts and definitions of both classical and Bayesian estimation required for this work will be presented briefly.
2. Theoretical Background

2.2.1.1. Classical Estimation

In order to be able to determine whether a found estimator is a “good” one, a performance measure has to be introduced. In many cases the mean square error (MSE) cost function is utilized as a performance criterion, which is defined as

$$\text{mse}(\hat{\theta}) = \sum_{i=1}^{N} \text{mse}(\hat{\theta}_i), \quad (2.34)$$

with

$$\text{mse}(\hat{\theta}_i) = E_y[(\hat{\theta}_i - \theta_i)^2], \quad (2.35)$$

where \(\hat{\theta}_i\) and \(\theta_i\) denote the \(i\)-th elements of \(\hat{\theta}\) and \(\theta\), respectively, and the averaging is performed over the PDF of the measurement vector \(y\). By expanding and rearranging the first term, it can be shown that the MSE cost function can be decomposed into

$$\text{mse}(\hat{\theta}_i) = \text{var}(\hat{\theta}_i) + |b(\hat{\theta}_i)|^2, \quad (2.36)$$

where \(\text{var}(\hat{\theta}_i) = E_y[(\hat{\theta}_i - E_y[\hat{\theta}_i])^2]\) is the variance and \(b(\hat{\theta}_i) = E_y[\hat{\theta}_i] - \theta_i\) is the bias of the \(i\)-th element of the estimator. The MSE optimality criterion, however, leads in general to optimal estimators which depend on the unknown parameter and are therefore not realizable [19]. As a consequence, the restriction of unbiasedness, i.e. \(b(\hat{\theta}_i) = E_y[\hat{\theta}_i] - \theta_i = 0\) for all possible values of \(\theta_i\) is introduced for many estimation problems, which leads to the class of unbiased estimators. For this class of estimators a minimization of the MSE collapses to a minimization of the variances \(\text{var}(\hat{\theta}_i)\) of the estimator. The estimator with the minimum variances \(\text{var}(\hat{\theta}_i)\) for all \(i = 1, ..., N\) and all possible values of \(\theta_i\) is termed the minimum variance unbiased (MVU) estimator. In some estimation problems, however, for different values of \(\theta_i\) different estimators show the minimum variance. In these cases the MVU estimator does not exist. Additionally, it is important to emphasize that the MVU estimator, if it exists, is only the optimal estimator in the class of unbiased estimators, it might thus be possible to find a biased estimator which performs better than the MVU estimator in terms of the MSE. Even if the MVU estimator exists, it is generally a hard task to determine it. Furthermore, possibly not the full PDF, but only the first and the second order statistics of the measurements may be available. Faced with these difficulties, often an additional restriction on the estimator is introduced, namely to be linear in the data, that is

$$\hat{\theta} = E_y, \quad (2.37)$$

where \(E\) is the estimator matrix. The resulting best estimator for the parameter estimation problem, which can be obtained with the knowledge of only the first and second moment of the PDF of the measurements, is referred to as the best linear unbiased estimator (BLUE) [19]. The linearity constraint might, in comparison to the MVU estimator, in some cases severely degrade the performance, however, for the linear model considered in this work it can be shown (c.f. [19]) that the BLUE is also the MVU es-
2.2. Traditional Estimation Methods

2.2.1.2. Bayesian Estimation

Analogously to the classical estimation, an error criterion has to be introduced to determine the performance of the estimators. The MSE cost function delivers results which depend, in general, on the realization of the parameter vector to be estimated (which is a random variable in the Bayesian framework) [19]. In order to avoid this, the Bayesian mean square error (BMSE), defined as

\[ \text{Bmse}(\hat{\theta}) = \sum_{i=1}^{N} \text{Bmse}(\hat{\theta}_i) \]  

(2.38)

with

\[ \text{Bmse}(\hat{\theta}_i) = E_{y, \theta}[\{(\theta_i - \hat{\theta}_i)^2\}] \],  

(2.39)

is employed as a performance measure. Although it makes sense to consider different error criterions, which is also done later in this section, for now the focus lies on the quadratic cost function. In contrast to the MSE the BMSE is obtained by averaging not only over the conditional PDF of the measurement vector \( p(y|\theta) \), but also over the PDF of the parameter vector \( p(\theta) \). The estimator which minimizes the BMSE is termed the minimum mean square error (MMSE) estimator, and in contrast to the MVU estimator it always exists. It can be shown (c.f. [19]) that the MMSE estimator is the mean of the posterior distribution, i.e.

\[ \hat{\theta}_{\text{MMSE}} = E_{\theta|y}[\theta|y] = \int \theta p(\theta|y) d\theta = \frac{\int \theta p(y|\theta)p(\theta) d\theta}{p(y)} = \frac{\int \theta p(y|\theta)p(\theta) d\theta}{\int p(y|\theta)p(\theta) d\theta} \]  

(2.40)

where \( \theta \) is assumed to be a continuous random vector, for discrete random vectors the integrals have to be replaced by sums. Due to the multidimensional integrations that have to be carried out, it is in many cases computationally intractable to obtain the MMSE estimate of a given estimation problem. One approach is to restrict the estimator to be linear (or actually affine) in the data, which leads to the linear minimum mean square error (LMMSE) estimator. That is, the LMMSE estimator is of the form

\[ \hat{\theta} = E_{y}y + b \],  

(2.41)

where \( E \) is the estimator matrix and \( b \) is the bias vector, which can be omitted, if \( y \) and \( \theta \) have zero mean. Similar to the BLUE in classical estimation, only first and second order statistics of the joint PDF \( p(y, \theta) \) are needed to determine the LMMSE estimator [19]. The application of the MMSE and the LMMSE estimators in this work is detailed in Section 2.2.2 and Section 2.2.3, respectively.
As already mentioned, it is also reasonable to utilize non-quadratic cost functions. One important cost function is the so-called "hit-or-miss" function. In case of a scalar parameter $\theta$ to be estimated, the performance criterion, also termed Bayes risk, is defined as [19]

$$R(\hat{\theta}) = E_{y,\theta}[C(|\theta - \hat{\theta}|)],$$  \hspace{1cm} (2.42)

with the "hit-or-miss" cost function

$$C(|\theta - \hat{\theta}|) = \begin{cases} 
0 & |\theta - \hat{\theta}| < \delta \\
1 & |\theta - \hat{\theta}| \geq \delta 
\end{cases},$$  \hspace{1cm} (2.43)

where $\delta$ can be made arbitrarily small. Especially for applications where $\theta$ is a discrete random variable this cost function can be interpreted easily: the estimates are solely distinguished to be correct or incorrect, whereby no distance measure is introduced for further gradation. It can be shown (c.f. [19]) that the maximum a posteriori (MAP) estimator, defined as

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(\theta|y),$$  \hspace{1cm} (2.44)

minimizes the Bayes risk for the "hit-or-miss" cost function. Using Bayes’ theorem

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}$$

and considering that $p(y)$ does not depend on $\theta$ the MAP estimator can be rewritten as

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} p(y|\theta)p(\theta).$$  \hspace{1cm} (2.45)

For the special case of a uniform prior PDF $p(\theta)$ (or at least a uniform PDF over the range of $\theta$ where $p(y|\theta)$ is non-zero), the MAP estimator coincides with the Bayesian maximum likelihood (ML) estimator

$$\hat{\theta}_{\text{ML}} = \arg \max_{\theta} p(y|\theta),$$  \hspace{1cm} (2.46)

which maximizes the so-called likelihood function $p(y|\theta)$. For the extension to a parameter vector $\theta$ there exist two variants of the MAP estimator. The first option is to carry out the MAP estimation for each element of the parameter vector $\theta_i$ individually, i.e.

$$\hat{\theta}_i = \arg \max_{\theta_i} p(\theta_i|y), \quad i = 1, \ldots, N.$$  \hspace{1cm} (2.47)

This approach minimizes the Bayes risk for every element of the parameter vector

$$R_i = E_{y,\theta_i}[C(\theta_i - \hat{\theta}_i)],$$  \hspace{1cm} (2.48)

where $C(\theta_i - \hat{\theta}_i)$ is defined as in the scalar case. In order to obtain the required marginal
posterior PDF \( p(\theta_i|y) \) from the joint posterior PDF \( p(\theta|y) \), the remaining \( N-1 \) elements of the parameter vector have to be integrated out:

\[
p(\theta_i|y) = \int \cdots \int p(\theta|y) \, d\theta_1 \cdots d\theta_{i-1} \, d\theta_{i+1} \cdots d\theta_N .
\]  

(2.49)

Hence, the element-wise MAP estimator suffers, like the MMSE estimator, from a high computational complexity. This leads to the second option, where the Bayes risk is chosen to be

\[
R(\hat{\theta}) = E_{y,\theta}[C(||\theta - \hat{\theta}||_2^2)] ,
\]

(2.50)

where

\[
C(||\theta - \hat{\theta}||_2^2) = \begin{cases} 
0 & ||\theta - \hat{\theta}||_2^2 < \delta \\
1 & ||\theta - \hat{\theta}||_2^2 \geq \delta 
\end{cases}
\]

(2.51)

The corresponding optimal estimator, termed vector MAP estimator [19], is given as follows:

\[
\hat{\theta} = \arg \max_{\theta} p(\theta|y) .
\]

(2.52)

Note that no integration is needed to obtain the estimate \( \hat{\theta} \). However, the maximization of \( p(\theta|y) \) is often also a computationally demanding task, which can be seen in Section 2.2.2.1, where the parameter vector is a discrete random variable. It is important to note that the two variants of the MAP estimator deliver, in general, different estimates \( \hat{\theta} \). Both of them are elaborated and compared for the data estimation problem of this work in Section 2.2.2.1.

2.2.2. Optimal Estimators for Data Estimation

2.2.2.1. Maximum a Posteriori Estimation

In a first step, the vector MAP estimator is applied to the model (2.20) to estimate the transmitted data vector \( d \), which consists of \( N_d \) BPSK or QPSK modulated data symbols and is therefore treated as a discrete complex random vector. Hence, the vector MAP estimator can be written as

\[
\hat{d} = \arg \max_{d' \in S^{N_d}} p[d'|y] \\
= \arg \max_{d' \in S^{N_d}} \frac{p(y|d')p[d']}{p(y)} \\
= \arg \max_{d' \in S^{N_d}} p(y|d') ,
\]

(2.53)
where $S^{Nd}$ denotes the set of all possible symbol vectors. Further, the data symbols are assumed to be iid as well as equiprobable, leading to a prior probability mass function (PMF) of a data vector of $p(d) = \frac{1}{|S|}$, where $|S|$ is the alphabet size, i.e. 2 and 4 for BPSK and QPSK, respectively. As a consequence, the constant factor $p(d')$, as well as $p(y)$, which is independent of $d'$, can be omitted in the last step of (2.53) and the vector MAP reduces to the Bayesian vector ML estimator.

Considering the data model, Eq. (2.20), the conditional PDF $p(y|d')$ can be determined to be a circularly symmetric complex Gaussian distribution, i.e. $y|d' \sim CN(Hd', N\sigma_n^2 I)$. Hence, the Bayesian vector ML estimator is

$$\hat{d} = \arg\max_{d' \in S^{Nd}} \frac{1}{(\pi N\sigma_n^2)^{Nd}} e^{-\frac{1}{N\sigma_n^2}||y-Hd'||_2^2}.$$ (2.54)

Finding the maximum of $p(y|d')$ can be simplified by applying the natural logarithm $\ln(.)$ to the likelihood function, which does not have an influence on the position of the maximum since it is a strictly monotonic function. This simplifies the vector ML estimator to

$$\hat{d} = \arg\max_{d' \in S^{Nd}} \ln \left( \frac{1}{(\pi N\sigma_n^2)^{Nd}} e^{-\frac{1}{N\sigma_n^2}||y-Hd'||_2^2} \right)$$

$$= \arg\max_{d' \in S^{Nd}} \ln \left( \frac{1}{(\pi N\sigma_n^2)^{Nd}} \right) - \frac{1}{N\sigma_n^2}||y-Hd'||_2^2$$

$$= \arg\max_{d' \in S^{Nd}} (-||y-Hd'||_2^2)$$

$$= \arg\min_{d' \in S^{Nd}} ||y-Hd'||_2^2,$$ (2.55)

where in the second to last step the terms independent of $d'$ are disregarded. This is the optimal estimator with respect to the data vector error probability, i.e. it minimizes the probability that $\hat{d} \neq d$. The proof of this statement, which is a slightly altered version of that given in [20], is given in the following.

Proof. Let $g : C^N \rightarrow S^{Nd}$ be an arbitrary estimator, which maps the received symbol vector $y$ to one of the possible data vectors $d' \in S^{Nd}$, i.e. $d' = g(y)$. Furthermore, $p(d|y)$ is the posterior PMF of $d|y$ evaluated at $d = d'$, which describes for a given $y$ the probability that the estimated parameter vector $\hat{d}$ is equal to the transmitted vector $d$ and is thus a correct estimate. That is, the probability of a correct estimation is $Pr(d = \hat{d}|y) = Pr(d = g(y)|y) = p(d|y) = p(g(y)|y)$. The error probability can be
expressed as
\[
P_{\text{err}} = \Pr(d \neq g(y)) \\
= 1 - \Pr(d = g(y)) \\
= 1 - \int \Pr(d = g(y)|y)p(y)dy \\
= 1 - \int p(g(y)|y)p(y)dy.
\] (2.56)

The vector MAP estimator \( \hat{d}_{\text{MAP}} = g_{\text{MAP}}(y) = \arg\max_{d^\prime \in S^N} p[d^\prime|y] \) maximizes the posterior PMF by definition, i.e.
\[
p[g_{\text{MAP}}(y)|y] \geq p[g(y)|y] \quad \text{for all possible estimators } g(y).
\]

As \( p(y) \) is non-negative, the integral in (2.56) is maximized by the vector MAP estimator. This, in turn, leads to a minimization of the error probability \( P_{\text{err}} \), which concludes the proof. \( \square \)

In communications, however, the bit error probability \( P_{b,\text{err}} \) and not the error probability of the whole data vector is usually the quantity of interest. Consequently, a special form of the element-wise MAP estimator, specifically a MAP estimation for every bit of every element of \( d \), is regarded. Starting from its definition, the MAP estimator for the \( j \)-th bit \( b_{ji} \) of the \( i \)-th data symbol \( d_i \) of the symbol vector \( d \) can be written as follows:
\[
\hat{b}_{ji} = \arg\max_{b_{ji} \in \{0,1\}} p[b'_{ji}|y] \\
= \arg\max_{b_{ji} \in \{0,1\}} \sum_{d'' \in S_{ji}^{(b_{ji})}} p[d''|y] \\
= \arg\max_{b_{ji} \in \{0,1\}} \sum_{d'' \in S_{ji}^{(b_{ji})}} p(y|d'')p[d'']/p(y) \\
= \arg\max_{b_{ji} \in \{0,1\}} \sum_{d'' \in S_{ji}^{(b_{ji})}} p(y|d''),
\] (2.57)

where \( S_{ji}^{(b_{ji})} \subset S^N \) is the set of data vectors with the bit \( b_{ji} \) fixed to the value \( b'_{ji} \).

Inserting the expression for the likelihood \( p(y|d'') = \mathcal{N}(Hd'', N\sigma_n^2 I) \) leads to
\[
\hat{b}_{ji} = \arg\max_{b_{ji} \in \{0,1\}} \sum_{d'' \in S_{ji}^{(b_{ji})}} \frac{1}{(\pi N\sigma_n^2)^N} e^{-\frac{1}{2\pi\sigma_n^2}||y-Hd''||^2} \\
= \arg\max_{b_{ji} \in \{0,1\}} \sum_{d'' \in S_{ji}^{(b_{ji})}} e^{-\frac{1}{2\pi\sigma_n^2}||y-Hd''||^2}.
\] (2.58)
The bit-wise MAP estimator is the desired optimal estimator regarding the bit error probability $P_{b, err}$. The proof of this statement can be conducted in a similar fashion to the proof of the vector MAP estimator’s optimality regarding the symbol vector error probability, which has already been shown earlier in this section. The bit error probability $P_{b, err} = \Pr(b_{ji} \neq g(y))$ of an arbitrary estimator $g : \mathbb{C}^N \mapsto \{0, 1\}$ results in the same expression as in (2.56), however, in this case the posterior PMF $p_g(y)\mid y$ describes the probability of a correct bit decision for a given $y$. Since the bit-wise MAP estimator

$$\hat{b}_{ji} = \arg \max_{b'_{ji} \in \{0, 1\}} p[b'_{ji} \mid y]$$

(2.59)

maximizes the posterior PMF $p[\hat{b}_{ji} \mid y]$ by definition no other estimator can lead to a smaller bit error probability, which follows by a similar argumentation as for the corresponding statement of the vector MAP estimator.

A comparison of (2.55) and (2.58) reveals an interesting difference, namely, the vector MAP estimator does not depend on the noise variance $N\sigma_n^2$, while the bit-wise MAP estimator does. An illustrative example, which addresses how the noise variance influences the decision boundaries of the bit-wise MAP estimator and to what extent the decision boundaries of the two MAP estimators differ, is presented in Section 2.2.5.

Both estimators, however, suffer from exponential complexity in the length of the data vector $N_d$. In case of the vector MAP estimator, every possible data vector has to be considered to determine which one maximizes the likelihood $p(y\mid d)$. Hence, assuming data symbols are drawn from a BPSK alphabet, $2^8 = 256$ and $2^{32} \approx 4.3 \cdot 10^9$ different data vectors have to be taken into account for the two UW-OFDM systems I ($N_d = 8$) and II ($N_d = 32$) regarded in this work, respectively. Furthermore, for QPSK data symbols, the number of different data vectors grows to $4^8 = 65536$ and $4^{32} \approx 1.8 \cdot 10^{19}$ for system I and system II, respectively. The situation becomes even worse for the bit-wise MAP estimator since for every single bit estimate $\hat{b}_{ji}$ one sum$^2$ has to be computed over those data vectors where the $j$-th bit of the $i$-th data symbol is fixed to ‘0’ or ‘1’, c.f. Eq. (2.58). Hence, all possible data vectors have to be considered and, in addition, the sum of exponential functions has to be calculated. Consequently, it is computationally intractable to evaluate the BER of the optimal estimators for system II, and thus the suboptimal estimators can be compared with the best possible one solely for system I. For further information on system I and II it is referred to Table 4.1.

Finally, the very similar issue of defining optimality for channel decoders is pointed out briefly. The decoding of convolutional codes is, as the transmitted bits are assumed to be equiprobable$^3$, typically accomplished by the maximum likelihood sequence estimator (MLSE), which is usually implemented by the Viterbi algorithm [18]. This approach minimizes the error probability of the whole decoded sequence of bits, however, it does not lead to a minimization of the bit error probability. The latter can

$^2$Since $p[b_{ji} = 1\mid y]$ and $p[b_{ji} = 0\mid y]$ in (2.58) have to sum up to one, only one of these values has to be computed.

$^3$Both of the mentioned algorithms can also be implemented for unequal prior probabilities.
be achieved by applying a bit-wise maximum likelihood estimator, implemented by the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm \[21\]. Similarly to the bit-wise MAP estimator in comparison to the vector MAP estimator, the BCJR algorithm suffers from a higher complexity than the Viterbi algorithm. Additionally, the performance gap regarding the BER between the BCJR and the Viterbi algorithm is in many cases negligible (c.f. e.g. \[22\]). As a consequence, the Viterbi algorithm is the far more popular choice for channel decoding and is also utilized in this work.

### 2.2.2.2. Minimum Mean Square Error Estimator

In order to obtain the MMSE estimator for the given model (2.20), the mean of the posterior PMF, i.e. \( E_{d|y}[d|y] \), has to be computed:

\[
\hat{d} = E_{d|y}[d|y] = \sum_{d' \in S^N_d} d' p[d'|y] \tag{2.60}
\]

Applying Bayes’ theorem leads to:

\[
\hat{d} = \sum_{d' \in S^N_d} \frac{d' p(y|d') p[d']}{p(y)} = \sum_{d' \in S^N_d} \frac{d' p(y|d')}{p(y)} \sum_{d' \in S^N_d} p[d']
\]

\[
= \sum_{d' \in S^N_d} \frac{d' p(y|d')}{p(y)} \sum_{d' \in S^N_d} p[y|d']
\]

\[
= \sum_{d' \in S^N_d} \frac{d' e^{-\frac{1}{N\sigma^2}||y-Hd'||^2}}{\sum_{d' \in S^N_d} e^{-\frac{1}{N\sigma^2}||y-Hd'||^2}} \tag{2.61}
\]

where the prior PMF of the symbol vector \( p[d'] \) is again assumed to be uniform.

In contrast to the vector MAP estimator, the MMSE estimator, which delivers an estimate for the vector parameter \( d \) and is optimal with respect to the performance criterion

\[
B_{mse}(d) = E_{y,d}[||d - \hat{d}||^2] = \sum_{i=1}^{N_d} E_{y,d}[||d_i - \hat{d}_i||^2], \tag{2.62}
\]

also minimizes the element-wise performance criterion

\[
B_{mse}(\hat{d}_i) = E_{y,d_i}[||d_i - \hat{d}_i||^2]. \tag{2.63}
\]
2. Theoretical Background

This can be shown easily, as (2.62) is minimized when all terms of the sum are minimized. The element-wise performance criterion (2.63) can, in case of complex data symbols \(d_i\), further be expanded to

\[
\text{Bmse}(\hat{d}_i) = E_{y,d_i}[|(d_{i,\text{re}} + j d_{i,\text{im}}) - (\hat{d}_{i,\text{re}} + j \hat{d}_{i,\text{im}})|^2] \\
= E_{y,d_i}[(d_{i,\text{re}} - \hat{d}_{i,\text{re}})^2 + (d_{i,\text{im}} - \hat{d}_{i,\text{im}})^2] \\
= E_{y,d_i}[(d_{i,\text{re}} - \hat{d}_{i,\text{re}})^2] + E_{y,d_i}[(d_{i,\text{im}} - \hat{d}_{i,\text{im}})^2] \\
= \text{Bmse}(\hat{d}_{i,\text{re}}) + \text{Bmse}(\hat{d}_{i,\text{im}}),
\]

(2.64)

where \(d_{i,\text{re}} = \text{Re}\{d_i\}\) and \(d_{i,\text{im}} = \text{Im}\{d_i\}\), which is again only minimized if both terms of the sum are minimized. That is, the MMSE estimator minimizes both the BMSE of the real part and the BMSE of the imaginary part of every element of the estimated symbol vector, and the MMSE estimate of the \(i\)-th symbol \(\hat{d}_i\) can be written as

\[
\hat{d}_i = E_{d_i|y}[\text{Re}\{d_i\}|y] + j E_{d_i|y}[\text{Im}\{d_i\}|y].
\]

(2.65)

As already mentioned, in communications the BER is usually of interest, which is minimized by the bit-wise MAP estimator. However, in case of hard decision, i.e. when each estimate is rounded to the nearest possible symbol of the alphabet \(\mathbb{S}\), the MMSE estimator delivers for BPSK and QPSK the same estimates as the bit-wise MAP estimator and thus is also optimal regarding the BER, which is shown in the following. For higher modulation alphabets, the MMSE estimator has to be slightly modified to deliver optimal estimates with respect to the BER, which is detailed later in this section.

Firstly, the data symbols are assumed to be drawn from a BPSK alphabet, i.e.

\[
d_i \in \mathbb{S} = \{-1, 1\}.
\]

Then, the \(i\)-th element \(\hat{d}_i\) in (2.60) can be written as

\[
\hat{d}_i = \sum_{d' \in \mathbb{S}^N} d' p[d'|y] = \sum_{d_i \in \{-1, 1\}} d_i p[d_i|y] = -p[d_i = -1|y] + p[d_i = 1|y].
\]

(2.66)

The result \(\hat{d}_i\) is then rounded to the nearest possible data symbol to obtain the final hard decision estimate. In other words, the final hard decision estimate of the data symbol \(d_i\) is

\[
\hat{d}_i = \begin{cases} 
1 & p[d_i = 1|y] > p[d_i = -1|y] \\
-1 & \text{otherwise}
\end{cases}
\]

which is nothing but the decision criterion of the bit-wise MAP estimator.

The second utilized constellation in this work is QPSK, i.e.

\[
d_i \in \mathbb{S} := \rho \{1 + j, 1 - j, -1 + j, -1 - j\}.
\]
For the following derivation the normalization factor $\rho = \frac{1}{\sqrt{2}}$ is dismissed, as it does not change the estimator’s decision boundaries. Starting again from (2.60), the complex MMSE estimate $\hat{d}_i$ can be written as

$$
\hat{d}_i = \sum_{d' \in \mathbb{S}^{N_d}} d' p[d'|y] = \sum_{d'_i \in \mathbb{S}} (\Re\{d'_i\} + j\Im\{d'_i\}) p[(\Re\{d'_i\} + j\Im\{d'_i\})|y]
$$

$$
= \sum_{d'_{i, re} \in \{-1,1\}} \sum_{d'_{i, im} \in \{-1,1\}} d'_{i, re} p[(d'_{i, re} + j d'_{i, im})|y] + j \sum_{d'_{i, re} \in \{-1,1\}} \sum_{d'_{i, im} \in \{-1,1\}} d'_{i, im} p[(d'_{i, re} + j d'_{i, im})|y]
$$

$$
= \sum_{d'_{i, re} \in \{-1,1\}} d'_{i, re} p[d'_{i, re}|y] + j \sum_{d'_{i, im} \in \{-1,1\}} d'_{i, im} p[j d'_{i, im}|y] \tag{2.67}
$$

which shows that the real and the imaginary part of $\hat{d}_i$ are estimated individually (a hint on this result was already given in (2.64), as the BMSEs of both the real and the imaginary part are minimized independently). Further, the symbol decision for both the real and the imaginary part is

$$
\hat{d}_{i, re/im} = \begin{cases} 
1 & p[d_{i, re/im} = 1|y] > p[d_{i, re/im} = -1|y] \\
-1 & \text{otherwise}
\end{cases}, \tag{2.68}
$$

which again coincides with the bit-wise MAP estimator.

In case of a higher order constellation, e.g. 16-QAM, or 64-QAM, the error criterion has to be reformulated in terms of the estimated bit vector $\hat{b}$ and the transmitted bit vector $b$ to obtain the optimal estimate regarding the BER:

$$
\text{Bmse}(\hat{b}) = E_{y,b}[||b - \hat{b}||^2_2]. \tag{2.69}
$$

Consequently, the MMSE estimator of the bit vector is

$$
\hat{b} = E_{b|y}[b|y] = \sum_{b' \in \{0,1\}^{N_b}} b' p[b'|y], \tag{2.70}
$$

with the number of bits $N_b = N_d \cdot N_{bps}$, where $N_{bps}$ is the number of bits per symbol. Conducting the same steps as in (2.61) leads to

$$
\hat{b} = \frac{\sum_{b' \in \{0,1\}^{N_b}} b' e^{-\frac{1}{N_{bps}^2}||y-Hd(b')||^2_2}}{\sum_{b' \in \{0,1\}^{N_b}} e^{-\frac{1}{N_{bps}^2}||y-Hd(b')||^2_2}} \tag{2.71}
$$

where $d(b)$ is the data symbol vector corresponding to the bit vector $b'$, i.e. blocks of $N_{bps}$ bits in $b'$ are consecutively mapped to data symbols and stacked to the symbol.
vector \( \mathbf{d}(\mathbf{b}') \). The \( N_b \) entries \( \hat{b}_i \) of \( \hat{\mathbf{b}} \) are, in the case of hard decision, rounded to 1 if \( \hat{b}_i > 0.5 \) and to 0 otherwise. For the derivation of this estimator’s optimality with respect to the BER (2.70) has to be expanded for the estimate \( \hat{b}_i, i = 0, ..., N_b - 1 \), analogously to the case of BPSK data symbols, i.e.

\[
\hat{b}_i = \sum_{b' \in \{0,1\}^{N_b}} b'_i p[b'|\mathbf{y}] = \sum_{b'_i \in \{0,1\}} b'_i p[b'_i|\mathbf{y}] = p[b_i = 1|\mathbf{y}]. \tag{2.72}
\]

Since \( p[b_i = 0|\mathbf{y}] + p[b_i = 1|\mathbf{y}] = 1 \) and the decision threshold for the hard decision of \( \hat{b}_i \) is 0.5, the hard decision estimate is

\[
\hat{b}_i = \begin{cases} 
1 & \text{if } p[b_i = 1|\mathbf{y}] > p[b_i = 0|\mathbf{y}] \\
0 & \text{otherwise}
\end{cases}, \tag{2.73}
\]

which is the decision criterion of the bit-wise MAP estimator.

Similar to the (bit-wise) MAP estimator, the MMSE estimator suffers from a \( N_d \) exponential complexity. Hence, it can be computed only for small UW-OFDM systems (like system I).

### 2.2.2.3. Reliability Information

As already mentioned in Section 2.1.2, in case of coded data transmission the estimated symbol vector \( \hat{\mathbf{d}} \) is demapped to a sequence of log-likelihood ratios which provide reliability information and improve the performance of the Viterbi channel decoder. From the optimal estimators only the MMSE estimator is simulated for the coded transmission and these results serve as a benchmark. As obvious from the definition of the LLRs in (2.31), the posterior probabilities \( \Pr(b_{ji} = 1|\mathbf{y}) \) and \( \Pr(b_{ji} = 0|\mathbf{y}) \) have to be determined to obtain the desired soft information. For the simulation of coded transmission in this work solely QPSK is employed as modulation alphabet and thus LLRs are only derived for this specific case in the following. The idea of the derivation is, however, the same for all alphabets.

In the case of QPSK modulation, the real part of the \( i \)-th symbol estimate \( \text{Re}\{\hat{d}_i\} \), c.f. Eq. (2.65), can be written as

\[
\text{Re}\{\hat{d}_i\} = E_{d_i|\mathbf{y}}[\text{Re}\{d_i\}|\mathbf{y}] = \rho(-1)\Pr(\text{Re}\{d_i\} = -1|\mathbf{y}) + \rho\Pr(\text{Re}\{d_i\} = \rho|\mathbf{y})
\]

\[= \rho(-1)\Pr(b_{0i} = 0|\mathbf{y}) + \rho\Pr(b_{0i} = 1|\mathbf{y}), \tag{2.74}\]

where \( \rho \) is \( \frac{1}{\sqrt{2}} \) or 1 for a normalized or unnormalized alphabet, respectively.

Considering, that the posterior probabilities sum up to one, (2.74) can be rewritten as

\[
\text{Re}\{\hat{d}_i\} = \rho(2\Pr(b_{0i} = 1|\mathbf{y}) - 1), \tag{2.75}
\]
or

$$\text{Re}\{\hat{d}_i\} = \rho(1 - 2\Pr(b_{0i} = 0|y)) .$$

(2.76)

Rearranging those two expressions with respect to the posterior probabilities and inserting into the LLR definition (2.31), yields

$$L_{0i} = \ln \left( \frac{\Pr(b_{0i} = 1|y)}{\Pr(b_{0i} = 0|y)} \right) = \ln \left( \frac{\rho + \text{Re}\{\hat{d}_i\}}{\rho - \text{Re}\{\hat{d}_i\}} \right) .$$

(2.77)

The same steps can be conducted to obtain the LLR of the first bit of the $i$-th symbol $b_{1i}$ from the imaginary part of the MMSE estimate $\hat{d}_i$, which results in

$$L_{1i} = \ln \left( \frac{\Pr(b_{1i} = 1|y)}{\Pr(b_{1i} = 0|y)} \right) = \ln \left( \frac{\rho + \text{Im}\{\hat{d}_i\}}{\rho - \text{Im}\{\hat{d}_i\}} \right) .$$

(2.78)

The above given expressions for the LLRs are exact and lead practically to the same results\(^4\) as a direct evaluation of (2.31), which is done in [10]. Both approaches have around the same complexity. With the approach in this work obtaining the LLRs from the MMSE estimates is computationally cheap, but the computation of the MMSE estimates suffers from a high computational complexity.

### 2.2.3. Linear Estimators for Data Estimation

In order to obtain a data estimator which is also computationally feasible for large, practically relevant UW-OFDM systems, the linearity constraint

$$\hat{d} = Ey ,$$

(2.79)

with the estimator matrix $E \in \mathbb{C}^{N_d \times N_d + N_r}$, is introduced. In this work only the data estimation in the Bayesian context, where $d$ is a realization of a random vector, is considered. For the elaboration of classical linear data estimators for UW-OFDM systems it is referred to [7].

In a first step, a complex-valued data symbol constellation with proper statistics [23] is assumed, which requires that the so-called pseudo-covariance matrix vanishes:

$$C_{dd^*} = E[(d - E[d])(d - E[d])^T] = \mathbf{0} .$$

(2.80)

This property holds, if the real and the imaginary parts of the symbols are uncorrelated (which is fulfilled for iid input data) and exhibit the same variance [10]. In case of QPSK or higher order quadratic QAM constellations, (2.80) is fulfilled, however, the BPSK constellation also covered in this work has improper statistics. For the latter case, a linear estimator is presented later in this section.

---

\(^4\)Due to a different calculation the MATLAB results of both approaches slightly differ.
Applying the Bayesian Gauss-Markov Theorem [19] to (2.20) yields the linear minimum mean square error (LMMSE) estimator
\[
\hat{d} = \left( C_{dd}^{-1} + H^H C_{ww}^{-1} H \right)^{-1} H^H C_{ww}^{-1} y, \tag{2.81}
\]
whereby it is presumed that \( C_{dd} \) and \( C_{ww} \) are invertible, which is the case for iid symbols and WGN. Due to the assumptions of independent data symbols and circularly symmetric complex white Gaussian noise, the data vector covariance matrix \( C_{dd} \) and the noise covariance matrix \( C_{ww} \) can be simplified to \( C_{dd} = \sigma_d^2 I \), where \( \sigma_d^2 \) is the data symbol variance, and \( C_{ww} = N \sigma_n^2 I \). By additionally considering the definition of \( H = \hat{H} G \), the LMMSE estimator becomes [24]
\[
\hat{d} = Ey = \left( G^H \hat{H}^H \hat{H} G + \frac{N \sigma_n^2}{\sigma_d^2} I \right)^{-1} G^H \hat{H}^H y. \tag{2.82}
\]

The covariance matrix of the zero mean estimation error \( e = d - \hat{d} \) of the LMMSE estimator is given by [24]
\[
C_{ee} = N \sigma_n^2 \left( G^H \hat{H}^H \hat{H} G + \frac{N \sigma_n^2}{\sigma_d^2} I \right)^{-1}, \tag{2.83}
\]
which can be employed as a reliability measure.

A complexity optimized version of the LMMSE estimator and a sequential LMMSE estimator, as well as the BLUE for data estimation in UW-OFDM systems is detailed in [24].

Now, a data vector \( d \) with improper statistics is considered, which is the case if BPSK is employed as a modulation alphabet. The knowledge of improper data symbol statistics can be incorporated in the estimation, which improves the estimator’s performance. The resulting estimator is termed the widely linear minimum mean square error (WLMMSE) estimator. The theory of widely linear estimation is not in the scope of this work, consequently, only the results, which are taken from [10], are given in the following. The WLMMSE estimator is, in general, of the form
\[
\hat{d} = E_1 y + E_2 y^*, \tag{2.84}
\]
where the estimator matrices are defined as
\[
E_1 = (C_{dy} - C_{dy}^* (C_{yy})^{-1} (C_{yy}^*)^*) P_{yy}^{-1} \tag{2.85}
\]
and
\[
E_2 = (C_{dy}^* - C_{dy} C_{yy}^{-1} C_{yy}^*) (P_{yy}^{-1})^*, \tag{2.86}
\]
with the Schur complement
\[
P_{yy} = C_{yy} - C_{yy}^* (C_{yy})^{-1} C_{yy}^*. \tag{2.87}
\]
2.2. Traditional Estimation Methods

The general results are now refined for a real data vector \( d \), which is the case for BPSK symbols, and the UW-OFDM system model (2.20). Here, the WLMMSE estimator is of the form [10]

\[
\hat{d} = \mathbf{E}_1 \mathbf{y} + \mathbf{E}_i^* y^* = 2 \text{Re} \{ \mathbf{E}_1 \mathbf{y} \}, \tag{2.89}
\]

with \( \mathbf{E}_1 \) as defined in (2.85). Further, the (cross) covariance matrices result in

\[
\begin{align*}
\mathbf{C}_{dy} &= \sigma_d^2 \mathbf{G}^H \tilde{\mathbf{H}}^H, \\
\mathbf{C}_{yy} &= \sigma_d^2 \tilde{\mathbf{H}} \mathbf{G} \mathbf{G}^H \tilde{\mathbf{H}}^T + N \sigma_n^2 \mathbf{I}, \\
\mathbf{C}_{yy^*} &= \sigma_d^2 \mathbf{G} \mathbf{G}^T \tilde{\mathbf{H}}^T.
\end{align*}
\tag{2.90}
\]

Moreover, the estimation error \( e = d - \hat{d} \) has zero mean and its covariance matrix for a real valued data vector is given by

\[
\mathbf{C}_{ee} = \sigma_d^2 \mathbf{I} - 2 \text{Re} \{ \mathbf{E}_1 \mathbf{C}_{dy}^H \} \tag{2.91}
\]

2.2.3.1. Reliability Information

The log-likelihood ratio for the \( j \)-th bit of the \( i \)-th data symbol \( b_{ji} \) is defined in (2.31). However, it can be shown [25] that in case of LMMSE equalization and with the assumption that the conditional LMMSE estimates \( \hat{d}_i | d_i \) are Gaussian distributed (which is approximately fulfilled, see the elaborations later in this section) an alternative definition of the LLR

\[
L_{ji} = \ln \left( \frac{\Pr (b_{ji} = 1 | \hat{d}_i)}{\Pr (b_{ji} = 0 | \hat{d}_i)} \right) \tag{2.92}
\]

is equivalent to the definition given in (2.31). This definition is utilized to derive the LLR expressions of the LMMSE estimates since the LLR computations are then far less complex than they would be if (2.31) were used. As already mentioned, in this work for the coded transmission solely QPSK is utilized as a modulation alphabet. Hence, only the LLRs for the LMMSE estimates are considered in this section. For the evaluation of the LLRs of the WLMMSE estimates it is referred to [26]. In the following only a very brief summary of the derivation of the LLR expressions from [11], where all the details can be found, is given. For equiprobable symbols the expression for the LLR (2.92) can
2. Theoretical Background

be reformulated to

\[ L_{ji} = \ln \left( \frac{\sum_{s \in S_j^{(1)}} p\left( \hat{d}_i | d_i = s \right)}{\sum_{s \in S_j^{(0)}} p\left( \hat{d}_i | d_i = s \right)} \right), \tag{2.93} \]

where \( S_j^{(\alpha)} \subset S \) is the subset of data symbols for which the bit at the \( j \)-th position takes on the value \( \alpha \) and \( p(\hat{d}_i | d_i = s) \) is the conditional PDF of the estimated symbol \( \hat{d}_i \) conditioned on the actually transmitted symbol \( d_i = s \). Considering the form of a linear estimator and the model (2.20), the \( i \)-th estimated symbol can be expressed as

\[ \hat{d}_i = e_i^H y = e_i^H (h_i d_i + \tilde{H}_i \tilde{d}_i + w) = e_i^H h_i d_i + e_i^H \tilde{H}_i \tilde{d}_i + e_i^H w, \tag{2.94} \]

where \( e_i^H \) denotes the \( i \)-th row of the estimator matrix, \( \tilde{H}_i \) the matrix that results from \( H \) by removing the \( i \)-th column \( h_i \), \( d_i \) the data vector \( d \) without the symbol \( d_i \) and \( w \) circularly symmetric complex WGN. The term \( e_i^H \tilde{H}_i \tilde{d}_i \) represents the interference from all other symbols in the data vector on the estimate of \( d_i \), which is also termed as inter-parameter interference (IPI). With central limit theorem arguments it follows that for large data vectors the IPI term can be approximated as Gaussian noise and thus the conditional probability \( p(\hat{d}_i | d_i = s) \) can be well approximated by the Gaussian distribution

\[ p\left( \hat{d}_i | d_i = s \right) = \frac{1}{\pi \text{var}(\hat{d}_i | d_i = s)} e^{-\frac{1}{\text{var}(\hat{d}_i | d_i = s)} |\hat{d}_i - E[\hat{d}_i | d_i = s]|^2}, \tag{2.95} \]

where the conditional mean \( E_{y|d_i} [\hat{d}_i | d_i = s] \) and the conditional variance \( \text{var}(\hat{d}_i | d_i = s) \) can be determined as [11]

\[ E_{y|d_i} [\hat{d}_i | d_i = s] = e_i^H h_i d_i =: \alpha_i d_i, \]

\[ \text{var}(\hat{d}_i | d_i = s) = e_i^H (\tilde{H}_i C_{\tilde{d}_i \tilde{d}_i} \tilde{H}_i^H + C_{ww}) e_i =: \sigma_i^2. \tag{2.96} \]

Specifically for the LMMSE estimator and QPSK constellation, the LLRs for the 0-th and the first bit are [11]

\[ L_{0i} = \frac{4 \text{Re}\{\hat{d}_i\} \alpha_i \rho}{\sigma_i^2}, \quad L_{1i} = \frac{4 \text{Im}\{\hat{d}_i\} \alpha_i \rho}{\sigma_i^2}, \tag{2.97} \]

with \( \rho = \frac{1}{\sqrt{2}} \) for a normalized alphabet and \( \rho = 1 \) for an unnormalized alphabet, \( \alpha_i \) and \( \sigma_i^2 \) as defined in (2.96) and a bit-to-symbol mapping \( b_1 b_0 \mapsto d_i \) as given in (2.23).
2.2.4. Decision Feedback Equalizer

Besides the optimal, computationally highly complex data estimators and the low complex, but in general also worse performing linear data estimators, a number of suboptimal nonlinear equalizers exist, which provide a good trade-off between computational complexity and performance. Popular methods are e.g. noise interpolation, sphere decoding, or decision feedback equalization, to name just a few. In this work, only the latter one is employed and thus elaborated in the following, for the former two it is referred to [10].

Decision feedback equalization (DFE) is an iterative method which estimates one data symbol of the symbol vector in every iteration. The data symbol estimates $\hat{d}_i'$ are determined by a linear estimator. In this work the linear estimator which minimizes the BMSE is utilized, i.e. the WLMMSE estimator and the LMMSE estimator in the case of a BPSK and QPSK constellation, respectively. In the following, only the QPSK case using the LMMSE estimator is described, for BPSK constellation, the equations for the LMMSE estimate and its error covariance matrix have to be replaced by the counterparts of the WLMMSE estimator. The iterative estimation of the DFE starts by estimating the data symbol at position $i$ of the data vector

$$\hat{d}_i' = e_i^H y,$$  \hspace{1cm} (2.98)

where $e_i^H$ is the $i$-th row of the estimator matrix $E$ in (2.82). To determine which of the $N_d$ data symbols is estimated, the error covariance matrix $C_{ee}$ is calculated according to (2.83). Then, $i$ is chosen such that the $i$-th element on the main diagonal of the error covariance matrix $[C_{ee}]_{ii}$ is the smallest, i.e.

$$i = \arg \min_{j=1,\ldots,N_d} [C_{ee}]_{jj}.$$

According to [10], this is a well working approach for the decision which data symbol should be estimated in an iteration step of the DFE. With $\hat{d}_i'$, the data symbol $\hat{d}_i \in \mathbb{S}$ which is closest in terms of the Euclidean distance to $\hat{d}_i'$ is decided as the data symbol estimate. This operation, which is also denoted as slicing, represents the nonlinear part of the DFE. For the further steps of the estimation process, the model (2.20) has to be rewritten as

$$y = \bar{H}_i \bar{d}_i + h_i d_i + w,$$  \hspace{1cm} (2.99)

where $\bar{H}_i$ is the matrix $H$ without the $i$-th column $h_i$ and $\bar{d}_i$ is the data vector without the $i$-th data symbol $d_i$. This shows the impact of the data symbol $d_i$ on the received vector $y$. As an estimate of the $i$-th data symbol is available, which is assumed to be correct, the influence of this data symbol on $y$ can be eliminated. That is, the received vector is updated to

$$\bar{y} = y - h_i \hat{d}_i,$$  \hspace{1cm} (2.100)

which concludes the first iteration. The second one is started by estimating the next data symbol, however, $y$ and $H$ are replaced by $\bar{y}$ and $\bar{H}_i$, respectively. Moreover, only
2. Theoretical Background

$N_d - 1$ symbols are to be estimated and the dimension of $C_{ee}$ is also reduced by one. All further iterations follow the same procedure, until estimates for all data symbols, stacked in the correct order of the estimated data vector $\hat{d}$, are computed. In case of channel coding and soft-decision, some reliability information has to be provided to the Viterbi decoder. Due to the slicing operation, however, all elements of $\hat{d}$ can only attain values of the symbol alphabet. Therefore, the vector $\hat{d}'$ consisting of the estimated data symbols $\hat{d}'_i$ before the respective slicing-operation are used for further processing, which is detailed in the next paragraph.

2.2.4.1. Reliability Information

Due to the non-linear iterative equalization optimal results for coded transmission would be obtained by incorporating the channel decoding into the feedback loop of the DFE [10]. In this work, however, data estimation and channel decoding are always realized separately, and for the computation of the LLRs of the DFE a method which is similar to the approach proposed in [10], is chosen, which leads to satisfying results. Like in Section 2.2.3.1, the focus lies on the calculation of the LLRs in case of proper data vectors. The LLRs corresponding to the estimate $\hat{d}_i$ of the $i$-th data symbol are determined after the LMMSE data estimation $\hat{d}'_i$ in every iteration step of the DFE. Consequently, the calculation of the LLRs corresponding to $\hat{d}_i$ is very similar to that shown in Section 2.2.3.1, and for QPSK the expression of the LLRs in (2.97) can be adopted. However, it has to be considered that after every iteration of the equalization procedure the appropriate column in $H$ is removed and so $\bar{H}_i$ in (2.96) is the matrix resulting from $H$ of the current iteration step by deleting the $i$-th column. Further, also the data vector that contains the remaining data symbols to be estimated is reduced by one entry after every iteration and hence the number of rows and columns of $C_{\bar{d}_i}$ decreases by one from iteration to iteration.

2.2.5. Decision Boundaries of Traditional Data Estimators

In this section, a small example illustrating the decision boundaries of the traditional estimators is given. It is assumed that a system

$$y = Hd + w$$

is given, where

$$H = \begin{bmatrix} 0.9 & 0.6 \\ -0.3 & 0.5 \end{bmatrix}.$$ 

$d_1, d_2$ are drawn from a BPSK alphabet and $w \sim N(0, \sigma^2 I)$. This system has no connection to the UW-OFDM systems investigated in this work and should just visualize the differences between the decision boundaries of the estimators in case of hard decision.
For the given system a block of two bits $b_1b_0$ can be transmitted, whereby the mapping to the symbol vector $\mathbf{d}$ is defined as

$$
00 \mapsto \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad 01 \mapsto \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad 10 \mapsto \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad 11 \mapsto \begin{bmatrix} 1 \\ 1 \end{bmatrix}.
$$

In the following, the decision boundaries of the estimators are shown for different SNRs. Since

$$
(E_b/N_0)_{\text{lin}} = 10^{(E_b/N_0)_{\text{dB}}}/10 = \frac{E[||\mathbf{Hd}||^2]}{E[||\mathbf{w}||^2]} = \frac{\text{tr}(\mathbf{H}^T\mathbf{H})}{\sigma^2},
$$

(2.101)

$\sigma^2$ is chosen for a given value $E_b/N_0$dB as

$$
\sigma^2 = \frac{\text{tr}(\mathbf{H}^T\mathbf{H})}{10(E_b/N_0)_{\text{dB}}/10}.
$$

The vector ML estimator is, as already mentioned, the optimal estimator with respect to the error probability of the symbol vector $\mathbf{d}$. Its decision boundaries are independent of the SNR, which is also shown in Figure 2.3.

![Figure 2.3](image)

Figure 2.3.: Decision boundaries of the vector ML estimator for different SNRs.

The MMSE estimator, in turn, is the optimal estimator with respect to the bit error probability, and its decision boundaries coincide with those of the bit-wise MAP/ML detector. As Figure 2.4 shows, the noise variance influences the decision boundaries of the MMSE estimator, and for $\sigma^2 \to 0$ the MMSE estimator and consequently also the bit-wise ML estimator tend towards the vector ML estimator. This leads to the conclusion, that for low SNRs the MMSE estimator outperforms the vector ML estimator regarding the BER, but the performance difference vanishes for high SNRs.

5The UW-OFDM simulations, however, showed that the BER performance differences between the MMSE estimator and the vector ML estimator are also quite small for low SNRs.
2. Theoretical Background

![Figure 2.4.](image)

(a) $\frac{E_b}{N_0} = 5 \text{ dB}$  
(b) $\frac{E_b}{N_0} = 10 \text{ dB}$  
(c) $\frac{E_b}{N_0} = 15 \text{ dB}$

Figure 2.4.: Decision boundaries of the MMSE estimator for different SNRs.

The decision boundaries of the low-complexity LMMSE estimator can clearly only be represented by straight lines. The results for the decision boundaries shown in Figure 2.5 are quite different to those of the optimal estimators, which already indicates, that the linearity constraint may lead to a considerable performance degradation in case of hard decision.

![Figure 2.5.](image)

(a) $\frac{E_b}{N_0} = 5 \text{ dB}$  
(b) $\frac{E_b}{N_0} = 10 \text{ dB}$  
(c) $\frac{E_b}{N_0} = 15 \text{ dB}$

Figure 2.5.: Decision boundaries of the LMMSE estimator for different SNRs.

Finally, the decision boundaries of the DF equalizer are depicted in Figure 2.6. Here, the iteratively performed linear estimations are clearly visible, and the decision boundaries are far closer to those of the optimal estimators than it is the case for the LMMSE estimator.
2.2. Traditional Estimation Methods

![Decision boundaries of the DF estimator for different SNRs.](image)

(a) $\frac{E_b}{N_0} = 5$ dB  
(b) $\frac{E_b}{N_0} = 10$ dB  
(c) $\frac{E_b}{N_0} = 15$ dB

Figure 2.6.: Decision boundaries of the DF estimator for different SNRs.

2.2.6. Channel Estimation

In most of the investigations on different data estimators perfect channel knowledge is assumed. However, in real-world applications the CIR of the multipath channel can also only be estimated. Due to estimation errors of the CIR the performance of the data estimators degrades. It is of great practical interest to examine how the BER performance of neural network based data estimators changes in comparison to traditional data estimators and thus simulations with imperfect channel knowledge are carried out in Section 4.3.

It is assumed that no prior channel statistics are known, consequently the CIR is estimated in the classical context, specifically by the BLUE. For the purpose of channel estimation a known preamble, which is defined in [16] and also utilized in the IEEE 802.11a standard, is transmitted over the multipath channel and unavoidably further disturbed by AWGN. The preamble includes two identical pilot CP-OFDM time domain symbols $\mathbf{x}_p \in \mathbb{C}^N$, which are preceded by a guard interval that is chosen long enough to eliminate inter-OFDM-symbol interference. The pilot symbols are defined in frequency domain and consist of BPSK data symbols, i.e. $\tilde{\mathbf{x}}_p \in \{-1, 1\}^{N_d + N_r}$. The two received time domain CP-OFDM symbols $\mathbf{y}_p^{(1,2)} \in \mathbb{C}^N$ are averaged to reduce the influence of the AWGN:

$$\bar{\mathbf{y}}_p = \frac{1}{2} (\mathbf{y}_p^{(1)} + \mathbf{y}_p^{(2)}) .$$  (2.102)

Further, the corresponding vector in frequency domain excluding the zero subcarriers is

$$\tilde{\mathbf{y}}_p = \mathbf{B}^T \mathbf{F}_N \bar{\mathbf{y}}_p .$$  (2.103)

Now, the relation between the averaged received symbol in frequency domain $\tilde{\mathbf{y}}_p$ and the
known pilot symbol $\hat{x}_p$ can be modeled as follows [27]:

$$\hat{y}_p = D_p B^T \hat{h} + w,$$

(2.104)

with $D_p = \text{diag}(\hat{x}_p)$ and $w \sim \mathcal{CN}(0, \frac{1}{2} N \sigma_n^2 I)$, whereby $\sigma_n^2$ is the time domain noise variance and the factor $\frac{1}{2}$ comes from the averaging of the received vectors $y_p^{(1)}$ and $y_p^{(2)}$. Moreover, $D_p B^T \hat{h}$ results in an element-wise multiplication of $\hat{x}_p$ and the appropriate elements of the channel frequency response $\hat{h}$ (which is ensured by $B^T \hat{h}$), which corresponds to a cyclic convolution of the CP-OFDM pilot symbol and the CIR in time domain. As the connection between the channel frequency response $\hat{h}$ and the channel impulse response $h$ of length $l_h$ is

$$\hat{h} = F_N \begin{bmatrix} h \\ 0 \end{bmatrix} = M_1 h,$$

(2.105)

where $M_1$ consists of the first $l_h$ columns of $F_N$, the model (2.104) can be rewritten as

$$\hat{y}_p = D_p B^T M_1 h + w.$$

(2.106)

With the linear model (2.106) the BLUE, which in this case is also the MVU estimator, of the channel impulse response becomes [27]

$$\hat{h} = (M_1^H B B^T M_1)^{-1} M_1^H B D_p^{-1} \hat{y}_p$$

(2.107)

With (2.105) and an exclusion of the elements in $\hat{h}$ on the positions of zero subcarriers, an estimate of the channel frequency response

$$\hat{h} = B^T M_1 (M_1^H B B^T M_1)^{-1} M_1^H B D_p^{-1} \hat{y}_p$$

(2.108)

can be obtained. Finally, the estimated channel matrix results in $\hat{H} = \text{diag}(\hat{h})$.

2.3. Neural Networks

The main purpose of this thesis is to find and analyze data estimators which are based on neural networks. The aim is to approximate the estimator function of the optimal estimator $d_{\text{opt}} = g_{\text{opt}}(y, H)$ by the neural network’s function $\hat{d} = g(y, H; W)$ as good as possible, where $W$ represents the network’s parameters which are learned during training. This should be feasible, since, according to the universal approximation theorem [28], a feed forward network with a single hidden layer and sufficiently many neurons exists that can, under weak conditions, approximate any function arbitrarily accurately. However, the theorem does neither give the number of needed neurons (which can be incredibly high and, consequently, the learning problem becomes intractable), nor how to obtain the parameters $W$ of the neural network. Therefore, a number of challenges arise, like how to find a neural network that is expressive enough for the given prob-
lem, how to choose the network architecture such that the number of needed neurons is low, and how to train it such that the neural network generalizes well, to name just a few. In this section, a brief introduction of important definitions and concepts of neural networks, which are required for this work, is given.

2.3. Neural Networks

2.3.1. Feedforward Neural Networks

A feedforward neural network (FFN), or multilayer perceptron (MLP), consists of connected neurons that are organized in layers, which in turn are further distinguished into an input layer, hidden layers\(^6\) and an output layer (c.f. Figure 2.7). Assuming that the layers are numbered in ascending order from the input layer to the output layer, then in an FFN only connections between neurons from a lower layer to a higher layer exist i.e. there is no feedback connection. In the following a fully-connected feedforward neural network (FCNN) is assumed, i.e. there are only connections between neurons from consecutive layers and all neurons from layer \(i\) are connected to all neurons belonging to layer \(i + 1\). In case that feedback connections are present, the neural network is referred to as recurrent neural network (RNN), which is not further elaborated.

The neurons of the FCNN map their inputs \(s^{(k)}_i\) by a nonlinear function, the so-called activation function \(\varphi^{(k)}(.)\) to outputs, also termed activations, \(a^{(k)}_i\):\(^{(2.109)}\)

\[
a^{(k)}_i = \varphi^{(k)}\left(s^{(k)}_i\right),
\]

\(^6\)In case of many hidden layers, the FFN is usually referred to as deep neural network (DNN).
where the subscript $i$ and the superscript $(k)$ denote the affiliation to the $i$-th neuron in the $k$-th layer. Furthermore, the input $s_i^{(k)}$ of the $i$-th neuron in the $k$-th layer is a weighted sum of all activations of the neurons that are connected to the input of this neuron (which are in case of an FCNN all neurons from the lower layer). That is,

$$s_i^{(k)} = \sum_j w_{ij}^{(k)} a_j^{(k-1)} + b_i^{(k)}, \quad (2.110)$$

where $w_{ij}^{(k)}$ denotes a weighing factor and $b_i^{(k)}$ an additional bias. In vector-matrix notation the layer-wise activations can be formulated as

$$a^{(k)} = \varphi^{(k)} (W^{(k)}a^{(k-1)} + b^{(k)}), \quad (2.111)$$

where the activation function $\varphi^{(k)}(.)$ is applied element-wise and the activation vector of the input layer $a^{(0)}$ is the input vector $z$, i.e. $a^{(0)} = z$. Assuming that the activation function of all layers but the output layer are the same, the FCNN with $L$ layers can be described as

$$\hat{x} = \sigma (W^{(L)}a^{(L-1)} + b^{(L)})$$

$$= \sigma (W^{(L)}\varphi (W^{(L-1)}a^{(L-2)} + b^{(L-1)}) + b^{(L)})$$

$$= \sigma (W^{(L)}\varphi (W^{(L-1)}\varphi (... \varphi (W^{(1)}z + b^{(1)}) ... ) + b^{(L-1)}) + b^{(L)}) , \quad (2.112)$$

where $\hat{x}$ denotes the output vector of the network, $\sigma(.)$ the activation function of the output layer and $\varphi(.)$ the activation function applied in all other layers.

**Activation Functions** There exist numerous activation functions for hidden neurons, like e.g. the sigmoid function $\varphi(s) = \frac{1}{1+e^{-s}}$ or the hyperbolic tangent $\varphi(s) = \tanh(s)$. However, these activation functions introduce the so-called vanishing gradient problem [29]. A currently very popular activation function, which alleviates the vanishing gradient problem is the rectified linear unit (ReLU), defined as $\varphi(s) = \max\{0, s\}$. The ReLU activation function is used in most cases throughout this work. The activation function for the neurons in the output layer heavily depends on the task the neural network should accomplish and on the corresponding utilized loss function (a further elaboration on loss functions can be found in Section 2.3.2). For binary classification problems, typically the sigmoid function and for categorical classification the softmax function $\sigma(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{K} e^{s_j}}$, where every $s_i$ represents one of the $K$ possible classes, is used as output activation function, as then the outputs of the neural nets can be interpreted as probabilities. For regression tasks often the linear function $\sigma(s) = s$ is used as activation function of the output layer.
2.3. Neural Networks

2.3.2. Training and Testing of Neural Networks

The aim of training a neural network is to find a setting of network parameters and so-called hyperparameters, such that the expected loss for unseen data, also termed generalization error, or risk, is minimized. In case of the FCNN, its parameters are the weights and biases, while the hyperparameters are the number of layers, the number of neurons per layer and the learning rate, which also play an important role for the network’s performance. As already mentioned, during training an optimal setting for the parameters and hyperparameters should be found, which can be formulated as an optimization problem with respect to a defined loss function, detailed later in this section. This is, however, a very complex problem with a huge number of parameters to optimize, for which analytic solutions cannot be determined. Further, also the underlying data generating distribution is usually unknown. Consequently, the optimization and evaluation of the NNs is done in a data-driven way, where it is assumed that representative data is available. In this context it is important to introduce three different datasets. Firstly, the test set is the dataset, which consists of unseen data, i.e. all the data samples in the test set have never been used to optimize the neural network. The test set is utilized to determine the generalization error and thus to compare different neural network architectures, or, specifically in this work, the neural network based data estimators to the traditional equalizers. Secondly, the training set is the dataset, which consists of unseen data, i.e. all the data samples in the test set have never been used to optimize the neural network. The training set is utilized to determine the generalization error and thus to compare different neural network architectures, or, specifically in this work, the neural network based data estimators to the traditional equalizers. Secondly, the training set is the dataset, which is used to optimize the parameters of the neural network, i.e. to adjust the parameters in order to minimize the error of the network’s output for the given training samples. For the optimization of the parameters of the neural network, which are continuous variables, gradient descent based methods are usually utilized, the exact training procedure is described in the next paragraphs. The hyperparameters, in turn, are mainly discrete variables (e.g. the number of layers), or affect the training in a unique way (e.g. the learning rate). Hence, their optimization cannot be performed by means of gradient descent based methods, but are found most often with a grid search. That is, the networks are trained with different hyperparameter settings and the best setting is determined by an evaluation of the trained networks on a third dataset, namely, the validation set. Furthermore, the validation set is also utilized to detect overfitting by assessing the error on this dataset during training on a regular basis and hence find the point where the network’s generalization error does not improve anymore and even begins to get worse again. This technique is termed early stopping, for more details c.f. Section 3.5.5. Note, that all three datasets have to represent the underlying data distribution well, however, it is crucial that the datasets are independent to obtain reliable results.

Similarly to the traditional estimators, a measure for the performance of the neural network is needed. First of all, the loss function, which measures the error of a single prediction of the neural network $\hat{x}$, is defined. It turns out (c.f. Section 2.3.5), that it is very beneficial to employ solely the quadratic loss function

$$\ell(x, \hat{x}) = \frac{1}{2} ||x - \hat{x}||^2_2 = \frac{1}{2} ||x - g(z; W)||^2_2$$

in this work, where $x$ is the true target vector, $g(.; W)$ the parameterized function of the neural network and $\hat{x}$ the network’s prediction for a given input $z$. As already
mentioned, the risk, which is the expected loss \[30\]

\[
R(\mathbf{g}(\cdot; \mathbf{W})) = E_{\mathbf{x}, \mathbf{z}}[\ell(\mathbf{x}, \hat{\mathbf{x}})],
\]

\[
\text{(2.114)}
\]

where the averaging PDF is the unknown data generating distribution \(p(\mathbf{z}, \mathbf{x})\), is the quantity that should be minimized. The problem that \(p(\mathbf{z}, \mathbf{x})\) is unknown is circumvented by minimizing the empirical risk \[30\]

\[
R_{\text{emp}}(\mathbf{g}(\cdot; \mathbf{W})) = \frac{1}{K} \sum_{i=1}^{K} \ell(\mathbf{x}^{(i)}, \mathbf{g}(\mathbf{z}^{(i)}; \mathbf{W})) \approx R(\mathbf{g}(\cdot; \mathbf{W}))
\]

\[
\text{(2.115)}
\]

using the data samples \(\mathbf{x}^{(i)}\) and \(\mathbf{z}^{(i)}\) from the training set (or from the test set to estimate the generalization error of the trained neural network) of size \(K\). According to the law of large numbers, \(R_{\text{emp}}(\mathbf{g}(\cdot; \mathbf{W}))\) approximates the risk \(R(\mathbf{g}(\cdot; \mathbf{W}))\) for large \(K\) and a properly chosen dataset.

The standard methods for training a neural network, i.e. minimizing the (empirical) risk, are gradient descent based methods. That is, starting from initially specified parameters \(\mathbf{W}_{\text{init}}\) (mostly they are chosen randomly), the parameters of a neural network \(\mathbf{W}\) are adjusted iteratively, according to

\[
\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} - \eta' \nabla_{\mathbf{W}} R_{\text{emp}}(\mathbf{g}(\cdot; \mathbf{W}_{\text{old}})),
\]

\[
\text{(2.116)}
\]

where \(\eta'\) is the learning rate and \(\nabla_{\mathbf{W}} R_{\text{emp}}(\mathbf{g}(\cdot; \mathbf{W}_{\text{old}}))\) is the gradient of the empirical risk with respect to \(\mathbf{W}\), evaluated at \(\mathbf{W}_{\text{old}}\). As the empirical risk is utilized, this procedure is also termed empirical risk minimization. Inserting (2.115) into (2.116), leads to

\[
\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} - \eta' \nabla_{\mathbf{W}} \frac{1}{K} \sum_{i=1}^{K} \ell(\mathbf{x}^{(i)}, \mathbf{g}(\mathbf{z}^{(i)}; \mathbf{W}_{\text{old}}))
\]

\[
= \mathbf{W}_{\text{old}} - \eta \sum_{i=1}^{K} \nabla_{\mathbf{W}} \ell(\mathbf{x}^{(i)}, \mathbf{g}(\mathbf{z}^{(i)}; \mathbf{W}_{\text{old}})),
\]

\[
\text{(2.117)}
\]

with \(\eta = \frac{\eta'}{K}\). The iterative adaption of the parameters of the neural network lasts until either a parameter setting is found such that the magnitude of the gradient of the empirical risk becomes negligibly small, or a fixed number of iterations is reached. As (2.117) shows, the gradient of the loss \(\nabla_{\mathbf{W}} \ell(\mathbf{x}^{(i)}, \mathbf{g}(\mathbf{z}^{(i)}; \mathbf{W}))\) for all \(K\) training samples has to be computed to determine \(\nabla_{\mathbf{W}} R_{\text{emp}}(\mathbf{g}(\cdot; \mathbf{W}))\), which, in turn, is needed for a single weight update. Due to the usually large training set, this learning method, which is referred to as batch or deterministic learning \[31\], is computationally very expensive. The true gradient of the empirical risk can, however, be approximated by

\[
\nabla_{\mathbf{W}} R_{\text{emp}}(\mathbf{g}(\cdot; \mathbf{W})) = \frac{1}{K} \sum_{i=1}^{K} \nabla_{\mathbf{W}} \ell(\mathbf{x}^{(i)}, \mathbf{g}(\mathbf{z}^{(i)}; \mathbf{W})) \approx \frac{1}{B} \sum_{i=1}^{B} \nabla_{\mathbf{W}} \ell(\mathbf{x}^{(i)}, \mathbf{g}(\mathbf{z}^{(i)}; \mathbf{W}))
\]

\[
\text{(2.118)}
\]

whereby a subset of the training set of size \(B\), termed a mini-batch, is employed to
compute the approximate gradient of the empirical risk. In the extreme case of \( B = 1 \) the training method is called stochastic or online learning, and for batch sizes larger than one mini-batch or mini-batch stochastic learning [31]. It is very common to refer to mini-batch gradient descent as stochastic gradient descent (SGD), which will be done in this work, too. The samples for a mini-batch are drawn randomly from the training set. Afterwards, these samples are not considered for the following mini-batches anymore, until all samples from the training set have been used for a mini-batch SGD update step (sampling without replacement). When all training samples were chosen for an update step, which is also referred to as a completed epoch, another epoch starts and the samples from the whole dataset can be selected for a mini-batch again. As already mentioned, the true gradient of the empirical risk is only approximated for SGD training and the larger the batch size, the more accurate is, in general, the approximation. According to [32] a smaller batch size and, consequently, a noisier gradient can lead to a regularizing effect and thus a better generalization error than with large batch sizes can be achieved. Hence, the batch size is a further hyperparameter that has to be chosen properly.

The gradient descent based parameter updates require the computation of the gradient of the loss \( \nabla_W \ell(x^{(i)}, z^{(i)}; W) \) for a given data sample \( x^{(i)}, z^{(i)} \), which can be implemented efficiently by the back-propagation algorithm [33]. In an FFN the gradient in the \( k \)-th layer can be computed recursively on basis of the gradient in the \((k+1)\)-th layer\(^7\), i.e. the gradient of the loss is “propagated backwards” layer by layer. A detailed elaboration of the back-propagation algorithm is omitted here and can be found e.g. in [31].

2.3.3. Deep Unfolding

Model based methods are, especially in communications, currently a widespread approach for solving engineering tasks. As already explained for the traditional data estimators, model based methods can, however, be computationally highly demanding or provide only suboptimal performance. Further, their performance might severely degrade in case of insufficient problem domain knowledge and, additionally, model based methods also cannot benefit from available data. On the contrary, neural networks, can learn patterns from data, but it is hardly possible to integrate domain knowledge into them. Deep unfolding, which is proposed in [34], is an approach to incorporate model knowledge into deep neural networks. The idea is to start with an iterative algorithm that is derived from a model based approach and “unfold” the iteration steps as layers of a deep neural network. The remaining free parameters of the resulting neural network are then learned with training data. There exist already some neural network architectures for data estimation which are based on deep unfolding, like e.g. OAMP-Net [4], LcgNet [5] and DetNet [2]. The latter is thoroughly investigated in this work for the application in an UW-OFDM system, a detailed description of the architecture of DetNet is given in Section 3.2.

\(^7\)The layers are again numbered in ascending order from the input to the output layer.
2. Theoretical Background

2.3.4. Self-Attention and Transformers

The Transformer architecture [35] is currently a state-of-the-art approach for many sequence-to-sequence tasks like neural machine translation (the translation of texts from one to another language by neural networks). When it was published in 2017, it was the first architecture that did not utilize RNNs or convolutional neural networks (CNNs), but relied solely on the so-called self-attention mechanism, with which dependencies in the input and the output sequences can be captured. As the input sequence does not have to be processed sequentially as it is the case for RNN based architectures, the Transformer architecture is highly parallelizable. A Transformer is realized as encoder-decoder structure, in this work, however, only its encoder part is employed. Consequently, only the encoder of the Transformer is detailed in the following and for further details on the whole architecture it is referred to [35].

The encoder of the Transformer consists of multiple encoder layers which, in turn, are composed of two sublayers, namely a multi-head self-attention layer and a position-wise fully-connected feedforward network with a single hidden layer. Additionally, a residual connection [36] around both layers and a layer normalization [37] after each layer is applied. The structure of one encoder layer is visualized in Figure 2.8, which also highlights the very parallelized structure of an encoder layer.

![Figure 2.8.: Structure of one encoder layer of the Transformer.](image)

Mulit-head self-attention is an extension of the self-attention mechanism, which is elucidated first. For this purpose queries \( q_i \), keys \( k_i \), and values \( v_i \) are introduced, which are the results of a mapping of the elements \( m_i \) of the input sequence \( \{m_1, ..., m_N\} \) by learned parameters \( W_Q, W_K, W_V \), i.e.

\[
q_i = W_Q m_i, \quad k_i = W_K m_i, \quad v_i = W_V m_i, \quad (2.119)
\]

with \( i = 1, ..., N_S \). Note, that the parameters are unique per encoder layer, a layer index of the parameters is here omitted for readability. As all the queries, keys and values originate from elements of the same sequence, the attention mechanism is termed self-attention, while for queries, keys and values from elements of different sequences it is
referred to as co-attention. Every query \( q_i \) is compared with every key \( k_j \) by computing weighing coefficients

\[
\alpha_{ij} = \text{softmax}\left( \frac{q_i^T k_j}{\sqrt{d_k}} \right),
\]

(2.120)

where \( d_k \) is the dimension of the keys. The scaling by \( \frac{1}{\sqrt{d_k}} \) is introduced to counteract small gradients for large \( d_k \) and, consequently, for a better learning behaviour [35]. That is, the weighing coefficients are a measure for the “compatibility” between all elements of the input sequence. The output of a self-attention layer is then a sequence \( \{s_1, ..., s_{N_s}\} \), where

\[
s_i = \sum_{j=1}^{N_s} \alpha_{ij} v_i.
\]

(2.121)

The operations in (2.120) and (2.121) can also be written in matrix form, which results in [35]

\[
S = \text{Attention}(Q, K, V) = \text{softmax}\left( \frac{QK^T}{\sqrt{d_k}} \right) V,
\]

(2.122)

with \( S = [s_1, ..., s_{N_s}]^T \), \( Q = [q_1, ..., q_{N_s}]^T \), \( K = [k_1, ..., k_{N_s}]^T \), \( V = [v_1, ..., v_{N_s}]^T \) and a row-wise applied softmax function.

In case of multi-head self-attention with \( H \) heads, every input element \( m_i \) is projected to \( H \) different queries \( q_{i,h} \), keys \( k_{i,h} \) and values \( v_{i,h} \) by \( H \) different learned parameters \( W_{Q,h}, W_{K,h} \) and \( W_{V,h} \), respectively, where \( h = 1, ..., H \). On each of the \( H \) queries, keys and values self-attention is applied again, resulting in \( H \) outputs \( s_{i,h} \), which are concatenated and projected once again by the learned parameter \( W_O \) to obtain the output of the multi-head self-attention layer. Mathematically, multi-head self attention can be written as [35]

\[
S = [S_1, ..., S_H] W_O,
\]

\[
S_h = \text{Attention}(Q_h, K_h, V_h) = \text{softmax}\left( \frac{Q_h K_h^T}{\sqrt{d_{k,h}}} \right) V_h,
\]

(2.123)

where \( d_{k,h} \) is the dimension of the keys for every attention head, which is defined to be \( d_{k,h} = \frac{d_k}{H} \) in [35]. According to [35], multi-head self-attention layers lead to more expressive models than the single-head self-attention layers do.

As already mentioned, the multi-head self-attention layer is followed by a position-wise fully-connected feedforward network with a single hidden layer which is defined as

\[
s = W_2 \varphi_{\text{ReLU}}(W_1 z + b_1) + b_2,
\]

(2.124)

where the output of the hidden layer typically has a larger dimension than the input \( z \) and the output \( s \), which have the same dimension. The weights \( W_1 \) and \( W_2 \) and
the bias vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ are the same for all positions $i = 1, ..., N$ (the feedforward blocks in Figure 2.8 all have the same parameters), differ, however, from encoder layer to encoder layer.

The self-attention mechanism and the architecture of the encoder layers is utilized for another neural network based data estimator, namely the Attention Detector, which is detailed in Section 3.4.

### 2.3.5. Log-Likelihood Ratios for Neural Network Based Data Estimators

As already mentioned, in case of channel coded transmission it is important to provide a reliability measure of the estimation in form of log-likelihood ratios (LLRs) to improve the performance of the Viterbi algorithm, which is utilized for channel decoding. The LLR of the $j$-th bit of the $i$-th data symbol $b_{ji}$ of a symbol vector is defined in (2.31) and repeated here again:

$$L_{ji} = \ln \left( \frac{\Pr(b_{ji} = 1|y)}{\Pr(b_{ji} = 0|y)} \right).$$  \hspace{1cm} (2.125)

Consequently, it were highly beneficial if an NN based data estimator would provide the posterior probabilities $\Pr(b_{ji} = 1|y)$ and $\Pr(b_{ji} = 0|y)$ for a given received data vector $y$. There are a few approaches to provide posterior probabilities by the NN based data estimators. Firstly, the posterior probabilities for a given received data vector $y$ could be computed as for the bit-wise MAP estimator in Section 2.2.2.1. The NNs could then be trained with the computed posterior probabilities. However, the computation of the posterior probabilities is computationally highly expensive (c.f. the discussion on the complexity of the bit-wise MAP estimator in Section 2.2.2.1) and thus the generation of a training set with a few thousand data samples and corresponding posterior probabilities is not feasible. Secondly, the NNs could be trained to directly approximate the MMSE estimates\(^8\) and compute the LLRs using (2.77) and (2.78). A third approach, which is utilized in this work and is also proposed in [2], is to deliver approximations for the posterior probabilities by the neural network based data estimators. For this purpose, the neural networks are trained to estimate the so-called one-hot representation $d_{oh}$ instead of the data vector $d$. In a first step, let’s assume that the data symbols are drawn from a real-valued constellation, where $\mathcal{S} = \{s_1, ..., s_{|\mathcal{S}|}\}, s_k \in \mathbb{R}$, and the symbols of the alphabet are numbered in ascending order, i.e. $s_{k-1} < s_k$. Then, the one-hot vector corresponding to $s_k$ is defined to be a unit vector of length $|\mathcal{S}|$ with a 1 at position $k$. That is, for BPSK the one-hot mapping is defined as

$$b_i = 0 \rightarrow d_i = -1 \rightarrow d_{oh,i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad b_i = 1 \rightarrow d_i = 1 \rightarrow d_{oh,i} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \hspace{1cm} (2.126)$$

\(^8\)To this end, the NNs have to be trained with the true data vectors $d$ utilizing a quadratic loss function.
Further, the reverse operation can be written as

\[ d_i = f_{oh}(d_{oh,i}) = s^T d_{oh,i}, \]  

(2.127)

where \( s^T = [s_1, \ldots, s_{|S|}] \). The one-hot mapping of the whole data vector \( d \) is obtained by stacking the one-hot representations \( d_{oh,i} \) of the individual data symbols:

\[
\begin{bmatrix}
    d_{oh,0} \\
    \vdots \\
    d_{oh,N_d-1}
\end{bmatrix}
\]  

(2.128)

In case of a complex constellation a complex symbol can be mapped to one single one-hot vector, or the real- and the imaginary parts of the data symbols can be mapped separately to their corresponding one-hot representation. In this work the latter approach has been chosen. Consequently, the complex alphabet \( S \subset \mathbb{C} \) is split into two subsets \( S^{re} \subset \mathbb{R} \) and \( S^{im} \subset \mathbb{R} \), containing the real and the imaginary parts of the complex symbols, respectively. The mapping of both the real and the imaginary part of the data symbols then follows the same procedure as for real-valued alphabets. For QPSK\(^9\) the one-hot mapping is therefore defined as

\[
\begin{align*}
    & b_{0i} = 0, b_{1i} = 0 \rightarrow d_i = -1-j \rightarrow d_{re,oh,i}^{re} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad d_{im,oh,i}^{im} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\
    & b_{0i} = 0, b_{1i} = 1 \rightarrow d_i = -1+j \rightarrow d_{re,oh,i}^{re} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad d_{im,oh,i}^{im} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \\
    & b_{0i} = 1, b_{1i} = 0 \rightarrow d_i = 1-j \rightarrow d_{re,oh,i}^{re} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad d_{im,oh,i}^{im} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\
    & b_{0i} = 1, b_{1i} = 1 \rightarrow d_i = 1+j \rightarrow d_{re,oh,i}^{re} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad d_{im,oh,i}^{im} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.
\end{align*}
\]  

(2.129)

The one-hot representation of the data vector \( d \) is then obtained by

\[
\begin{bmatrix}
    d_{re,oh,1} \\
    \vdots \\
    d_{re,oh,N_d} \\
    d_{im,oh,1} \\
    \vdots \\
    d_{im,oh,N_d}
\end{bmatrix}
\]  

(2.130)

The combination of one-hot encoded data symbols and the utilization of a quadratic loss function leads to the advantageous property, that a neural network approximately provides posterior probabilities, which is shown in the following. As described in Section 2.3.2, a properly trained neural network minimizes the empirical risk \( R_{emp}(g(\cdot; W)) \). In case of an appropriately chosen training set the empirical risk is in turn an approxima-

\(^9\)The normalization factor \( \frac{1}{\sqrt{2}} \) for unit data variance is omitted here.
2. Theoretical Background

Inserting the quadratic loss function and replacing the variables in (2.131) by the corresponding variables of the target application, namely the one-hot representations of the data vectors \( d_{oh} \) for the target data and the received data vector \( y \) for the input data, yields

\[
\hat{d}_{oh} \approx \arg \min_{d_{oh}} E_{d,y}[\|d_{oh} - d'_{oh}\|^2] = E_{d,y}[d_{oh}|y],
\]

whereby the last step is well known from the derivation of the MMSE estimator \([19]\). The posterior expectation can also be expressed element-wise, i.e. the neural network’s estimate for the \( i \)-th one-hot vector is

\[
\hat{d}^{re}_{oh,i} \approx \begin{bmatrix} E_{d,y}[d^{re}_{oh,i,1}|y] \\ \vdots \\ E_{d,y}[d^{re}_{oh,i,|S^{re}|}|y] \end{bmatrix}, \quad \hat{d}^{im}_{oh,i} \approx \begin{bmatrix} E_{d,y}[d^{im}_{oh,i,1}|y] \\ \vdots \\ E_{d,y}[d^{im}_{oh,i,|S^{im}|}|y] \end{bmatrix},
\]

where a complex-valued alphabet is assumed, for a real-valued alphabet the one-hot vector for the imaginary part can be omitted.

Considering the definition of the elements of a one-hot vector

\[
d^{re}_{oh,ik} = \begin{cases} 1 & \text{Re}\{d_i\} = s^{re}_k \\ 0 & \text{Re}\{d_i\} \neq s^{re}_k \end{cases}, \quad d^{im}_{oh,ik} = \begin{cases} 1 & \text{Im}\{d_i\} = s^{im}_k \\ 0 & \text{Im}\{d_i\} \neq s^{im}_k \end{cases},
\]

where \( s^{re}_k \in S^{re} \) and \( s^{im}_k \in S^{im} \), leads to

\[
E_{d,y}[d^{re}_{oh,ik}|y] = 1 \cdot \Pr(d^{re}_{oh,ik} = 1|y) + 0 \cdot \Pr(d^{re}_{oh,ik} = 0|y)
= \Pr(d^{re}_{oh,ik} = 1|y)
= \Pr(\text{Re}\{d_i\} = s^{re}_k|y)
\]

and the same applies for the one-hot vector of the imaginary part. Inserting (2.134) into (2.133) results in

\[
\hat{d}^{re}_{oh,i} \approx \begin{bmatrix} \Pr(\text{Re}\{d_i\} = s^{re}_1|y) \\ \vdots \\ \Pr(\text{Re}\{d_i\} = s^{re}_{|S^{re}|}|y) \end{bmatrix}, \quad \hat{d}^{im}_{oh,i} \approx \begin{bmatrix} \Pr(\text{Im}\{d_i\} = s^{im}_1|y) \\ \vdots \\ \Pr(\text{Im}\{d_i\} = s^{im}_{|S^{im}|}|y) \end{bmatrix}.
\]

Consequently, for BPSK the neural network based data estimates of the \( i \)-th one-hot
2.3. Neural Networks

The vector are

\[ \hat{d}_{oh,i} \approx \begin{bmatrix} \Pr(b_i = 0|y) \\ \Pr(b_i = 1|y) \end{bmatrix} \]

and for QPSK they become

\[ \hat{d}_{re,oh,i} \approx \begin{bmatrix} \Pr(b_{0i} = 0|y) \\ \Pr(b_{0i} = 1|y) \end{bmatrix}, \quad \hat{d}_{im,oh,i} \approx \begin{bmatrix} \Pr(b_{1i} = 0|y) \\ \Pr(b_{1i} = 1|y) \end{bmatrix}. \]

These are exactly the quantities required for the calculation of the log-likelihood ratios (2.125). The extension to higher order constellations is straightforward.

As elaborated above, the \(k\)-th entry \(\hat{d}_{re,oh,ik}\) of an estimated one-hot vector \(\hat{d}_{re,oh,i}\) provides an approximation for the posterior probability of the \(k\)-th symbol from the alphabet \(S_{re}\), i.e. \(\hat{d}_{re,oh,ik} \approx \Pr(d_{re,i} = s_k|y)\). The same applies for the estimates of the one-hot vector of the imaginary part and \(S_{im}\), but in the following only the real part is considered. The MMSE estimates for \(\text{Re}\{\hat{d}_i\} = d_{re,i}\) are computed as

\[ \text{Re}\{\hat{d}_i\} = \mathbb{E}_{d_i|y}[d_{re,i}|y] = \sum_{s_k \in S_{re}} s_k \Pr(d_{re,i} = s_k|y). \tag{2.136} \]

Hence, the demapping of the estimates of the one-hot vectors to symbol estimates in the same manner as in (2.127), namely

\[ \hat{d}_{i} = f_{oh}(\hat{d}_{re,im,i}) = s^T \hat{d}_{re,im,oh,i} = \sum_{s_k \in S_{re/im}} s_k \hat{d}_{re,im,oh,ik}, \tag{2.137} \]

yields approximately the posterior mean of the transmit symbols. Consequently, the demapped estimates of the one-hot vectors \(\hat{d}_{re/im,i}\) provide approximations for the MMSE estimates. This result also reveals another property of the DetNet estimates and the MMSE estimates. Since the posterior probabilities are smaller than one, for QPSK all MMSE data symbol estimates lie in the complex plane within the square connecting the constellation points. This is in theory also true for the DetNet data symbol estimates, but as they deliver only approximately MMSE estimates a few “outliers” might occur. The distributions of the data symbol estimates of the MMSE estimator and the DetNet are very different to the distribution of the LMMSE estimates, which will be visualized in Section 4.5.
3. Neural Network Based Data Estimation for UW-OFDM

In this chapter, the neural network based approaches for data estimation are presented. The three different network architectures all utilize the received vector $y$ and the matrix $H$ in different ways to estimate the data vector $d$. In this work, these neural network architectures are investigated specifically for UW-OFDM systems, their application is, however, not restricted to these systems. They could also be employed for all kinds of multiple input multiple output (MIMO) systems, whereby the system dimensions account e.g. for multiple transmit and receive antennas, multiple users, or frequency subcarriers, just to name a few.

3.1. Inputs and Outputs of Neural Network based Data Estimators

The majority of the state-of-the-art neural network architectures can only handle a real-valued input. In order to reuse existing knowledge and concepts about neural networks, the complex-valued system model (2.20) is mapped to an equivalent real-valued system model as follows:

$$
y_R = H_R d_R + w_R,
$$

where

$$
y_R = \begin{bmatrix} \text{Re}\{y\} \\ \text{Im}\{y\} \end{bmatrix}, \quad d_R = \begin{bmatrix} \text{Re}\{d\} \\ \text{Im}\{d\} \end{bmatrix}, \quad w_R = \begin{bmatrix} \text{Re}\{w\} \\ \text{Im}\{w\} \end{bmatrix} \sim \mathcal{N}\left(0, \frac{1}{2} N\sigma_n^2 I\right),
$$

with the dimensions $y_R \in \mathbb{R}^{2(N_d+N_r)}$, $d_R \in \mathbb{R}^{2N_d}$, $H_R \in \mathbb{R}^{2(N_d+N_r) \times 2N_d}$ and $w_R \in \mathbb{R}^{2(N_d+N_r)}$. In the following complex-valued alphabets are always assumed to be symmetric, which means $S_R = \text{Re}\{S\} = \text{Im}\{S\}$. In case of a real-valued alphabet $S$, $d_R$ and
reduce to

\[ d_R = d, \quad H_R = \begin{bmatrix} \text{Re}\{H\} \\ \text{Im}\{H\} \end{bmatrix}. \]

Unless stated otherwise, in this chapter always the real-valued system (3.1) is utilized and for the purpose of readability the index \( R \) is omitted from now on.

As already mentioned, the neural network based data estimators predict the one-hot encoded data vector \( d_{oh} \) as described in Section 2.3.5, whereas approximations of the posterior probabilities are provided. Consequently, the same architectures, loss functions and training methods can be applied for both coded and uncoded transmission.

### 3.2. DetNet

DetNet is proposed in [2] and is based on the deep unfolding approach (c.f. Section 2.3.3). The idea of the authors is to mimic a projected gradient descent method to solve the optimization problem of the vector ML estimator. As shown in Section 2.2.2.1, the (real-valued) vector ML estimator is defined as

\[
\hat{d} = \arg\min_{d' \in \mathbb{S}^{N'd}} ||y - Hd'||^2_2, \tag{3.2}
\]

where \( N_d' = N_d \) or \( N_d' = 2N_d \) for a real-valued and complex-valued alphabet, respectively. Hence, the \( k \)-th iteration step of a projected gradient descent method of the vector ML estimator’s optimization problem is

\[
\hat{d}_k = \Pi \left( \hat{d}_{k-1} - \delta_k \frac{\partial ||y - Hd||^2_2}{\partial d} \bigg|_{d=\hat{d}_{k-1}} \right)
= \Pi \left( \hat{d}_{k-1} + 2\delta_k H^T y - 2\delta_k H^T H \hat{d}_{k-1} \right), \tag{3.3}
\]

where \( \Pi(\cdot) \) is a non-linear projection to a convex subspace \( \mathcal{D} \), which contains all possible data vectors \( d \), i.e. \( \mathbb{S}^{N_d'} \subset \mathcal{D} \subset \mathbb{R}^{N_d} \).

From (3.3), the structure of a single layer of the DetNet, which is visualized in Figure 3.1, can be inferred. The inputs of the \( k \)-th layer are the variables \( \hat{d}_{k-1} \) and \( v_{k-1} \), whose roles in the DetNet are detailed below. Firstly, the linear part in (3.3) is realized by computing [2]

\[
q_k = \hat{d}_{k-1} - \delta_{k1} H^T y + \delta_{k2} H^T H \hat{d}_{k-1}, \tag{3.4}
\]

where \( \delta_{k1} \) and \( \delta_{k2} \) are learned parameters and \( k \) is the layer index. Secondly, an FCNN with a single hidden layer follows, which replaces the projection operator \( \Pi(\cdot) \). As input of the FCNN \( q_k \) extended by \( v_{k-1} \) is utilized, the reason for this is discussed later in
this section. Consequently, the output of the FCNN can be written as
\[
\begin{bmatrix}
\hat{d}_{oh,k} \\
v_k'
\end{bmatrix} = \begin{bmatrix} W_{2k} \\ W_{3k} \end{bmatrix} \varphi_{\text{ReLU}} \begin{bmatrix} q_k \\ v_{k-1} \end{bmatrix} + b_{1k} + \begin{bmatrix} b_{2k} \\ b_{3k} \end{bmatrix},
\] (3.5)

with the learned parameters \(W_{1k} \in \mathbb{R}^{d_h \times d_q}, \ W_{2k} \in \mathbb{R}^{d_q \times d_h}, \ W_{3k} \in \mathbb{R}^{d_k \times d_h}, \ b_{1k} \in \mathbb{R}^{d_h}, \ b_{2k} \in \mathbb{R}^{d_q} \) and \(b_{3k} \in \mathbb{R}^{d_k} \) and the dimensions \(d_q = N_d' \cdot |\mathcal{S}| \) of \(q_k \) and \(\hat{d}_{oh,k}, \ d_v \) of \(v_{k-1} \) and \(v_k' \), \(d_{qv} = d_q + d_v \) and \(d_h > d_{qv} \) as the dimension of the FCNN’s hidden layer. Both \(d_v \) and \(d_h \) are hyperparameters of the DetNet. The initial values \(\hat{d}_0 \) and \(v_0 \) are chosen to be zero vectors.

Furthermore, weighted residual connections [36] are applied, i.e. the output of a DetNet layer is a weighted sum of its input and the output of its sublayers:
\[
\begin{bmatrix}
\hat{d}_k \\
v_k
\end{bmatrix} = \alpha \begin{bmatrix} \hat{d}_{k-1} \\ v_{k-1} \end{bmatrix} + (1 - \alpha) \begin{bmatrix} \hat{d}_k' \\ v_k' \end{bmatrix},
\] (3.6)

where the hyperparameter \(\alpha \in [0, 1] \) is the weighing factor and \(\hat{d}_k' = f_{oh}(\hat{d}_{oh,k}) \) is the demapped one-hot vector following the element-wise one-hot demapping (2.137). The output of the DetNet is then \(\hat{d}_{oh} = \hat{d}_{oh,L} \), where \(L \) is the number of layers.

For the training of the DetNet the loss function [2]
\[
\ell(d_{oh}, \hat{d}_{oh}) = \sum_{k=1}^{L} \log(k)||d_{oh} - \hat{d}_{oh,k}||^2_2
\] (3.7)
is utilized, i.e. also the interim results for \(\hat{d}_{oh} \) provided by the \(L \) layers are taken into account in the loss, which is inspired by the auxiliary classifiers in GoogLeNet [38]. According to [2], this loss function leads to a better and more robust learning behavior of the DetNet. However, this type of loss function forces all the interim one-hot vectors \(\hat{d}_{oh,k} \) of every layer to be close to the true one-hot encoded data vector. This, in turn, inhibits the neural network to capture very complex relations, which is one of
the main advantages of deep neural networks. To counteract this problem, the authors of [2] employ $\mathbf{v}_k$ to propagate important information through the network, without any constraints introduced by the loss function.

### Preconditioning

The investigations on the DetNet throughout this work revealed, that preprocessing steps can significantly improve its performance. Besides a normalization, which is detailed in 3.5, preconditioning turns out to be beneficial, as the performance of the DetNet suffers from ill-conditioned channel matrices. To this end, (3.2) is altered for the derivation of the structure of a DetNet layer to

$$\hat{d} = \arg \min_{d' \in \mathbb{S}_N} ||\mathbf{y} - \mathbf{H} \mathbf{L}^{-1} \mathbf{L} d'||_2^2,$$

(3.8)

where $\mathbf{L} \in \mathbb{R}^{N_d \times N_d}$ is an invertible matrix. With $\mathbf{d}_{pr} = \mathbf{L} \hat{d}$ the gradient descent step without projection, i.e. the linear part of (3.3), can be expressed as

$$\hat{d}_{pr,k} = \hat{d}_{pr,k-1} - \delta_k \frac{\partial||\mathbf{y} - \mathbf{H} \mathbf{L}^{-1} \mathbf{d}_{pr}||_2^2}{\partial \mathbf{d}_{pr}} |_{\mathbf{d}_{pr} = \hat{d}_{pr,k-1}}$$

$$= \hat{d}_{pr,k-1} + 2\delta_k \mathbf{L}^{-T} \mathbf{H}^T \mathbf{y} - 2\delta_k \mathbf{L}^{-T} \mathbf{H}^T \mathbf{H} \mathbf{L}^{-1} \hat{d}_{pr,k-1},$$

(3.9)

where $\mathbf{L}^{-T} = (\mathbf{L}^T)^{-1} = (\mathbf{L}^{-1})^T$. Thus, the $k$-th projected gradient descent iteration step is

$$\hat{d}_k = \Pi \left( \mathbf{L}^{-1} \hat{d}_{pr,k} \right)$$

$$= \Pi \left( \mathbf{L}^{-1} \left( \hat{d}_{pr,k-1} + 2\delta_k \mathbf{L}^{-T} \mathbf{H}^T \mathbf{y} - 2\delta_k \mathbf{L}^{-T} \mathbf{H}^T \mathbf{H} \mathbf{L}^{-1} \hat{d}_{pr,k-1} \right) \right)$$

$$= \Pi \left( \hat{d}_{k-1} + 2\delta_k (\mathbf{L}^T \mathbf{L})^{-1} \mathbf{H}^T \mathbf{y} - 2\delta_k (\mathbf{L}^T \mathbf{L})^{-1} \mathbf{H}^T \mathbf{H} \hat{d}_{k-1} \right)$$

$$= \Pi \left( \hat{d}_{k-1} + 2\delta_k \mathbf{P}^{-1} \mathbf{H}^T \mathbf{y} - 2\delta_k \mathbf{P}^{-1} \mathbf{H}^T \mathbf{H} \hat{d}_{k-1} \right),$$

(3.10)

with the preconditioning matrix $\mathbf{P} = \mathbf{L}^T \mathbf{L}$. In this work the low complex Jacobi preconditioner $\mathbf{P} = \text{diag}(\mathbf{H}^T \mathbf{H})$ is employed.

That is, the layer structure of the DetNet stays the same, however, for the computation of $\mathbf{q}_k$ in (3.4) $\mathbf{H}^T \mathbf{y}$ and $\mathbf{H}^T \mathbf{H}$ have to be replaced by $\mathbf{P}^{-1} \mathbf{H}^T \mathbf{y}$ and $\mathbf{P}^{-1} \mathbf{H}^T \mathbf{H}$, respectively. As both terms are constant for a given $\mathbf{H}$ and $\mathbf{y}$, they have to be computed only once and not in every layer. Hence, the increase in complexity due to the preconditioning is negligible.
3.3. FCNN Detector

As already mentioned, even an FCNN with a single hidden layer and sufficiently many neurons can, according to the universal approximation theorem [28], approximate any function arbitrarily accurately. Consequently, it is also possible to approximate the MMSE estimator, although the needed network architecture to accomplish this task is of course unknown. Efforts have already been made, e.g. in [2], to utilize an FCNN for data estimation. In case of a fixed channel, i.e. a single channel is employed for both training and testing the FCNN, the authors of [2] managed to successfully estimate $d$ with an FCNN and $y$ as input vector. However, the fixed channel is practically irrelevant for wireless communication systems. For varying channels the authors in [2] made the attempt to use an FCNN, where $H$, reshaped column-wise to a vector, and concatenated with $y$ serves as an input. However, they reported that this structure did not provide satisfying results.

In this work, an FCNN with an input which differs from the two input variants described in the last paragraph is employed for data estimation. To motivate the choice of the input, two observations have to be elucidated. Firstly, an inspection of the MMSE estimator (2.61) reveals, that this estimator is composed of a sum of exponential functions, with exponents

$$ ||y - Hd'||_2^2 = y^T y - 2d'^T H^T y + d'^T H^T Hd', \quad d' \in S^N_d. $$

That is, the MMSE estimator links the data vector $d$ with the known data $H$ and $y$ solely via the terms $H^T H$ and $H^T y$. Secondly, $H^T y$ provides a sufficient statistic for $d$, i.e. $H^T y$ contains all relevant information that is available from $y$ to estimate $d$, which is shown in the following.

Proof. Let $T(y) = H^T y$ be a statistic of $y$ and $p(y|d)$ the conditional PDF of the received UW-OFDM time domain symbol. Then, according to the Fisher-Neyman Factorization theorem [39], $T(y)$ is a sufficient statistic of $d$, if and only if $p(y|d)$ can be factored as

$$ p(y|d) = g(T(y)|d) h(y) \quad \forall y, d, $$

(3.11)

where $g(.)$ is a function depending on $y$ solely through $T(y)$ and on $d$, while $h(.)$ is a function depending only on $y$. The conditional PDF for the real-valued model (3.1) is

$$ p(y|d) = \frac{1}{(2\pi \frac{1}{2} N \sigma_n^2)^{2N}} e^{-\frac{1}{2N\sigma_n^2}||y-Hd||_2^2} $$

$$ = \frac{1}{(\pi N \sigma_n^2)^N} e^{-\frac{1}{2N\sigma_n^2} (y^T y - 2d^T H^T y + d^T H^T H d)} , $$

53
and can be factored as
\[
p(y|d) = \frac{1}{(\pi N\sigma_n^2)^N} \exp\left(-\frac{1}{N\sigma_n^2}(d^T\mathbf{H}^T\mathbf{H}d - 2d^T\mathbf{T}(y))\right) \exp\left(-\frac{1}{N\sigma_n^2}y^Ty\right),
\]
which concludes the proof.

That is, a multiplication of \( y \) by \( \mathbf{H}^T \), which modifies (3.1) to
\[
\mathbf{H}^T\mathbf{y} = \mathbf{H}^T\mathbf{H}d + \mathbf{H}^T\mathbf{w},
\]
does not lead to a loss of relevant information for the estimation of \( d \), but reduces the system dimension from \( 2(N_d + N_r) \) to \( N'_d \), which might also be beneficial for the complexity of the employed FCNN, as a lower dimensional input usually also leads to a more compact network. The relevant terms in this compressed system are again \( \mathbf{H}^T\mathbf{y} \) and \( \mathbf{H}^T\mathbf{H} \).

The two given arguments lead to the conclusion that \( \mathbf{H}^T\mathbf{H} \), reshaped column-wise to a vector, concatenated with \( \mathbf{H}^T\mathbf{y} \) is a better choice for the input than the variants described at the beginning of this section, which will be justified by the results shown in Chapter 4.

In order to obtain approximately posterior probabilities of the transmitted data bits, c.f. Section 2.3.5, the FCNN is trained to estimate the one-hot representation of the data vector \( \mathbf{d}_{oh} \), whereby the quadratic loss function
\[
\ell(d_{oh}, \hat{d}_{oh}) = ||d_{oh} - \hat{d}_{oh}||_2^2
\]
is employed as an error measure.

Finally, the ReLU activation function is utilized for the neurons of the hidden layers in combination with a linear output activation function, which is a standard approach for neural networks with a quadratic loss function.

**Preconditioning**

The modified system model (3.13) can of course also be multiplied by a nonsingular matrix \( \mathbf{P}^{-1} \in \mathbb{R}^{N_d \times N'_d} \) leading to
\[
\mathbf{P}^{-1}\mathbf{H}^T\mathbf{y} = \mathbf{P}^{-1}\mathbf{H}^T\mathbf{H}d + \mathbf{P}^{-1}\mathbf{H}^T\mathbf{w},
\]
3.4. Attention Detector

where $P^{-1}$ is chosen such that the condition number of the equation system is reduced. As preconditioning matrix the Jacobi preconditioner

$$ P = \text{diag}(H^T H) \quad (3.16) $$

is chosen, as it is the case for the DetNet. Hence, $H^T H$, reshaped as a vector, and $H^T y$ are replaced by $P^{-1} H^T H$ and $P^{-1} H^T y$ as inputs, respectively. The simulations showed that the effect of preconditioning on the performance of the FCNN is smaller than for the DetNet, however, a slight performance gain can still be achieved with negligible increase in complexity.

3.4. Attention Detector

Since $H^T y$ provides a sufficient statistic for $d$, which is shown in Section 3.3, it makes sense to consider the “compressed” model (3.13) instead of the model (3.1), repeated here once again for the sake of clarity:

$$ H^T y = H^T H d + H^T w. \quad (3.17) $$

As visualized in Figure 3.2, for a systematic UW-OFDM system, most of the non-zero entries of the matrix $H^T H$ are located on its diagonal, but there exist also non-diagonal non-zero entries. That is, a row of $H^T H$ is correlated with a few other rows, while the correlation with the remaining rows is negligible. This observation motivates the utilization of the self-attention mechanism, described in Section 2.3.4, in order to exploit those correlations to achieve a better estimation result. More specifically, instead of using $H^T H$ and $H^T y$ directly as input of an FCNN, as it is the case in Section 3.3 (apart from pre-processing), an abstract representation of these two quantities is computed by an encoder, which is very similar to the encoder of the Transformer detailed in Section 2.3.4. Afterwards, the encoder output is forwarded to a shallow FCNN for the final estimation of $d_{oh}$. 
The encoder consists of $N_{\text{enc}}$ encoder layers and each of them has the same structure as an encoder layer of the Transformer shown in Figure 2.8, however, the multi-head self-attention layer is replaced by an ordinary self-attention layer. As input sequence of the encoder serve the rows of

$$M = [m_1 \ldots m_{N_d}]^T = [H^T y, H^T H].$$

As the rows of the equation system (3.17) are interchangeable, the vectors $m_1$ to $m_{N_d}$ do not need a positional encoding, as it is the case for the elements of the input sequence of the Transformer [35].

The $N_d$ output vectors of the encoder $s_1, \ldots, s_{N_d}$ are concatenated to the input vector $s$ of a shallow FCNN. The output of the FCNN then provides the estimated one-hot encoded data vector $\hat{d}_{\text{oh}}$, which in turn contains, as the quadratic loss function (3.14) is employed, approximately the posterior probabilities of the data bits. As with the FCNN in Section 3.3, a ReLU activation function and a linear activation function are employed for the hidden layers and the output layer, respectively.

The generator matrix $G$ for non-systematic UW-OFDM is created in a way such that the columns are orthonormal, and since the taps of the CIRs are drawn from a normal distribution, which in turn means

$$E[\hat{H}^H \hat{H}] \propto I,$$

the columns of $H$ are orthonormal on the mean. Consequently, the matrix $H^T H$ is a diagonal matrix on the mean. As visualized in Figure 3.3, the off-diagonal entries of $H^T H$ are not only zero on the mean, but also have rather small absolute values for each realization of a channel. Consequently, for non-systematic UW-OFDM only very little performance gain in comparison to the FCNN Detector can be expected, as there exist hardly any correlations between the rows of $H^T H$ that can be exploited by the self-attention mechanism. However, there exist a number of MIMO systems in communications where the columns of $H$ are highly correlated, e.g. multi-element antenna communication systems, which are modeled by

\footnote{Here, the original definition of $\hat{H}$ from (2.19) is used and thus $\hat{H}$ is a complex matrix.}

Figure 3.2.: Absolute values in $H^T H$ of a systematic UW-OFDM system.
the one-ring model described in [40]. For data estimation in these systems the attention mechanism may still be worth to be considered.

Finally, also the input of the Attention Detector is preconditioned in the same way as for the FCNN Detector, detailed in Section 3.3. The performance gain is approximately the same as for the FCNN Detector and thus also the Attention Detector benefits from preconditioning less than the DetNet does.

3.5. Training Methodology

3.5.1. Data Normalization

A very important issue for NNs is the normalization of the input data, as their training behavior and thus their performance usually profits from normalized data. This means, the values of the data samples should follow a distribution with a fixed mean and variance (often the data is normalized to zero mean and unit variance). The input of the NN based data estimators consist of $H$ and $y$, whereby their relation is given by the system model (3.1). That is, $y$ consists of the component $Hd$, which is the data vector distorted by the channel, and the noise component $w$. As already mentioned in Section 2.1.2, the noise variance and thus its power is chosen according to a defined $E_b/N_0$ value, which is a measure for the SNR. For now, a slightly different SNR measure is defined, namely

$$\text{SNR} = \frac{E_d[||Hd||^2]}{E_w[||w||^2]} = \frac{E_d[\text{tr}(HdH^T)]}{N^2\sigma_d^2} = \frac{\frac{1}{N}\sigma_d^2\text{tr}(HH^T)}{N\sigma_n^2} = \frac{\frac{1}{N}\sigma_d^2\text{tr}(H^TH)}{N\sigma_n^2},$$

(3.18)

where $\sigma_d^2$ is the variance of the data symbols of the real-valued symbol alphabet, and the last step follows from the invariance of the trace operator under cyclic permutations of the multiplied matrices. All of the following results are also valid, when $E_b/N_0$ is
utilized as an SNR measure, since $E_b/N_0$ differs from the SNR definition (3.18) just by a constant factor.

In the majority of the current publications on neural network based data estimators, e.g. [2], [4], or [5], the elements of $\mathbf{H} \in \mathbb{R}^{2(N_d+N_r) \times N_d'}$ are assumed to be drawn iid from a standard normal distribution, i.e. $[\mathbf{H}]_{ij} \sim \mathcal{N}(0,1)$. This leads to matrices $\mathbf{H}^T\mathbf{H} \in \mathbb{R}^{N_d' \times N_d'}$, where

$$E\left[[\mathbf{H}^T\mathbf{H}]_{ij}\right] = \begin{cases} 2(N_d + N_r) & i = j \\ 0 & i \neq j \end{cases}$$

and

$$\text{var}\left((\mathbf{H}^T\mathbf{H})_{ij}\right) = \begin{cases} 4(N_d + N_r) & i = j \\ 2(N_d + N_r) & i \neq j \end{cases}.$$ 

Hence, the numerator in (3.18) has on the mean (averaged over all possible channel realizations) the value

$$E\left[\frac{1}{N}\sigma_d^2 \text{tr}(\mathbf{H}^T\mathbf{H})\right] = \frac{1}{N} \sigma_d^2 \sum_{i=1}^{N_d'} E[[\mathbf{H}^T\mathbf{H}]_{ii}] = 2 \frac{N_d + N_r}{N} \sigma_d^2 N_d',$$  

and a variance of

$$\text{var}\left(\frac{1}{N}\sigma_d^2 \text{tr}(\mathbf{H}^T\mathbf{H})\right) = \left(\frac{\sigma_d^2}{N}\right)^2 \sum_{i=1}^{N_d'} \text{var}(\mathbf{H}^T\mathbf{H})_{ii} = 4 \sigma_d^4 (N_d + N_r) N_d' \frac{N_d'}{N^2}.$$ 

Especially for larger systems, the numerator of (3.18) will therefore always be close to $2 \frac{N_d + N_r}{N} \sigma_d^2 N_d'$. This, in turn, means that the variance of the $2N$ elements of the noise vector $\mathbf{w}$ is

$$\text{var}(w_i) = \frac{N \sigma_n^2}{2} = \frac{\frac{1}{N} \sigma_d^2 \text{tr}(\mathbf{H}^T\mathbf{H})}{2\text{SNR}} \approx \frac{N_d + N_r}{\text{SNR} \cdot N} \sigma_d^2 N_d'$$

and thus is approximately constant for all realizations of $\mathbf{H}$. Consequently, the values of the noise component of $\mathbf{y}$ are always in the same range for a fixed SNR. That is, due to the choice of $\mathbf{H}$ the data estimation problem is implicitly also normalized. In case of small systems also for this choice of $\mathbf{H}$ a normalization might be advantageous, but the above mentioned publications omit any normalization.

For UW-OFDM systems, however, the matrix $\mathbf{H}$ has a different structure and thus the variance of the noise that has to be chosen to achieve a desired $E_b/N_0$ at the receiver is dependent on the channel realization, which necessitates normalization. A first approach is to multiply (3.1) by the normalization factor $\sqrt{\frac{2}{N\sigma_n}}$, which leads to unit variance of $w_i$, independent of the current realization of $\mathbf{H}$. However, this normalization method leads
3.5. Training Methodology

to numerical problems as $\sigma_n$ tends towards zero for high SNR values. Therefore,

$$\sqrt{\frac{N}{\text{tr}(H^TH)}} = \sqrt{\frac{N}{||H||_F}}$$

is utilized as a normalization factor, where $||H||_F$ is the Frobenius norm of $H$. With this normalization the numerator of (3.18) becomes

$$E_d \left| \frac{\sqrt{N}}{||H||_F} y \right|^2 = \frac{N}{||H||_F^2} E_d [\text{tr}(Hd^TH^T)] = \sigma_d^2,$$

which means, that the noise variance is, as desired, independent of the current channel realization and for a fixed SNR always the same.

The data normalization is implemented by multiplying both $y$ and $H$ by $\frac{\sqrt{N}}{||H||_F}$ before any further processing is performed to obtain the inputs of the the NN based data estimators.

3.5.2. Pre-Training

The weights and biases in the NNs are initialized randomly and their values are in the usual way drawn from the uniform distribution $\mathcal{U}(-\frac{1}{\sqrt{d_{in}}}, \frac{1}{\sqrt{d_{in}}})$, with $d_{in}$ as the dimension of the input feature vector. For training, data samples are taken randomly from the training set, which is generated by a number of transmissions at different SNRs, whereby the value for $E_b/N_0$ lies between a lower and an upper limit, c.f. Section 3.5.3. Hence, unfavorable combinations of initial weight and bias values, as well as training samples that have been generated at low SNRs, i.e. the received data symbols are highly noisy, could occur at the first few NN parameter updates randomly. This in turn could lead the SGD update steps in a wrong direction, could make the optimization either converge slowly or even could even lead to divergence, since gradient descent methods are in general very sensitive to initialization and wrong first steps. In order to avoid this scenario, the NNs are pre-trained with noiseless data samples. The pre-training dataset consists of 2000 different channel realizations, with a sequence of 500 UW-OFDM symbols transmitted over each channel. Furthermore, the NNs are pre-trained for seven epochs. The parameter values of the pre-trained NN serve then as initialization for the training of the NNs with the noisy data samples. Besides a more reliable training the pre-training also leads to a faster convergence of the training procedure.

3.5.3. SNR for Training

As already mentioned, the training data is generated with different values for $E_b/N_0$ within a specified lower and upper limit. The investigations throughout this work showed
that it is unfortunately not possible to train the regarded NN based data estimators over a wide SNR range, as this leads to an unsatisfactory performance. Intuitively this issue can also be explained, as also all traditional estimators, but the vector ML estimator, change their decision boundaries depending on the SNR, and the investigated NNs do not incorporate information about the SNR. Efforts to include the value of $E_b/N_0$ in the decision process of the NNs did not succeed. Hence, for the choice of the upper and the lower limit of $E_b/N_0$ values for the training set the SNR range of the test set has to be taken into account. Since the simulations in Chapter 4 are conducted for different SNR ranges, the training SNR range for each simulation is specified individually in the associated description. The $E_b/N_0$ values for the training set are chosen randomly in the specified SNR range. However, it is important to emphasize that the $E_b/N_0$ values of the training set are distributed uniformly on the linear scale between their specified upper and lower limits. A uniform distribution on the logarithmic scale, which is usually utilized to specify SNR ranges, would lead to an unbalanced training set, as the data samples with high $E_b/N_0$ values were underrepresented in this case. This underrepresentation in turn results in a poor performance of the neural networks at high SNRs.

### 3.5.4. Training Set Size and Deep-Fading Training Channels

Although all CIRs follow the channel model described in Section 2.1.2 there exist more and less demanding channels for data estimation. While the frequency responses of the former have deep fading holes, such that some subcarriers are heavily attenuated, the latter are rather flat in frequency domain. Two exemplary channel frequency responses for system II are plotted against the subcarrier indices in Figure 3.4, which makes the difference between a flat fading channel and a deep fading channel clearly visible. For the deep fading channel a few subcarriers, especially subcarrier 34, are heavily attenuated and thus it is very hard to recover the data symbols modulated onto these subcarriers. Further, it can be shown that the eigenvalues of $H$ are directly related to the frequency response of the channel. That is, deep fading channels result also in ill-conditioned $H$ matrices, which, as already mentioned, degrade the NN performance. In order to focus especially on the deep fading channels during the training of the NN based data estimators, the training set is composed of a subset of randomly generated channels, and a subset that consists solely of deep fading channels. Every channel of the latter subset is created by generating a few thousand channels according to the channel model and picking the worst channel. It is, however, important to choose the size of the subset of ill-conditioned channels not too large, as this would unbalance the training set too much and then lead again to a worse performance of the NNs on the test set. The subset sizes for a good training behavior of the NNs are found empirically. In this work, for simulations with system I the training sets consist of 27000 randomly generated channels and 3000 deep fading channels, and for those with system II of 20000 randomly generated channels and 20000 deep fading channels. The proportion of deep fading channels in

\[\text{Similarly, training with ill-condition channels is e.g. also suggested in [41] to enhance the performance of NNs.}\]
the training sets for system I and II is chosen to obtain NNs that generalize best. The reason why for system II a higher relative number of deep fading channels in the training sets is beneficial might be that the training and test SNRs are higher than for system I and thus the AWGN provides a weaker regularization effect for the NNs trained for system II, which is in turn improved by a higher proportion of deep fading channels.

![Figure 3.4.: Frequency responses of exemplary channels.](image)

### 3.5.5. General Training Settings

In this section a few general settings for the training of the NN based data estimators, which apply for all three of them, are summarized.

Firstly, the NNs are trained by the SGD based method called Adam Optimizer [42], which is a very robust optimization method and is employed for the training of many state-of-the-art neural networks. The parameters of the optimizer are chosen as $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, which are the very commonly used default parameter settings, and $\epsilon = 10^{-6}$. For the influence of these parameters on the training process it is referred to [42].

As mentioned earlier, the learning rate is one of the hyperparameters of the NN based data estimators and is thus chosen individually for every simulation case. However, the learning rate is not constant throughout the whole training process, but is decreased to allow fine-tuning of the parameters by smaller optimization steps. This improves convergence of the optimization process, as any small oscillations of the parameter values during the SGD optimization are damped due to the decaying learning rate. In this work the following learning rate schedule is employed. For the first 15% of the total optimizer steps $N_{total}$ the learning rate is left constant. Afterwards, the learning rate is exponentially decayed, whereby the learning rate update follows

$$lr_i = lr_0 \cdot 0.98^i, \quad i = 1, ..., 150,$$

where $i$ is the counter for the learning rate update, $lr_i$ is the learning rate after the $i$-th update step and $lr_0$ is the initial learning rate. The number of optimizer update
3. Neural Network Based Data Estimation for UW-OFDM

steps after which a learning rate update step is performed is determined by dividing the remaining 85% of the total optimizer update steps into 150 equidistant intervals. That is, the final learning rate update $lr_{150} = lr_0 \cdot 0.98^{150} = 0.0483 \cdot lr_0$ is conducted before the final optimizer step. This strategy leads to a good learning behavior of all three NNs.

Furthermore, early stopping is utilized as a regularization technique to avoid overfitting and thus achieve a better generalization error. That is, during training the performance of the NN on the validation set is checked regularly, more specifically, every $\lfloor N_{\text{total}}/40 \rfloor$ optimizer update steps. In case that the error made on the validation set is smaller than at any check before, the parameter setting of the NN is stored. When the training is finished, the parameter setting corresponding to the lowest error on the validation set throughout the whole training procedure is chosen as the final parameter setting of the NN. This method avoids overfitting and thus helps to find well generalizing NNs.

Finally, the choice of the batch size, i.e. the size of a mini-batch which is utilized for an single optimizer update, is described. The batch size has also an impact on the training behavior and the performance of NNs (c.f. Section 2.3.2). In this work the batch size is chosen to be 1024, which has proven to be a good choice to obtain well performing NN based data estimators on the one hand and to achieve a high utilization of the GPUs on the other hand, resulting in shorter training times than with smaller batch sizes.
4. Results

The three different neural network architectures detailed in Chapter 3 are evaluated for both the systematic and non-systematic UW-OFDM transmission scheme, for two different system dimensions, for coded and uncoded data transmission, and for the modulation alphabets BPSK and QPSK. Due to the multitude of possible combinations of simulation settings and the computationally expensive and thus time-consuming hyperparameter optimizations of the NN based data estimators only selected simulation cases are presented in this chapter, while those simulation cases for which no further insights are expected are omitted. To assess the accuracy of the NN based data estimators, the resulting BERs are plotted against the corresponding $E_b/N_0$ values and compared with the BERs traditional data estimators achieve. Mostly the LMMSE estimator and the DF equalizer are employed to represent the traditional data estimators, in selected cases, the MMSE estimates are also computed to gain insight into the performance gap to the best possible estimation performance. In most simulation cases in this chapter, perfect channel knowledge is assumed. The impact of imperfect channel knowledge on the data estimation performance is investigated in Section 4.3. Further, in Section 4.5, the distributions of the estimated values of the different estimation methods are visualized in constellation diagrams. Finally, in Section 4.6, the complexities of the different data estimators are compared.

Unless stated otherwise, preconditioning as well as a normalization of the input data as described in Chapter 3 is conducted for all NN based data estimators in all simulations and is in the following not explicitly mentioned anymore.

4.1. Simulation Framework

The whole UW-OFDM transmission chain and thus the data generation is conducted in MATLAB, based on a framework which was utilized to produce the simulation results in [11]. Furthermore, the traditional data estimators are implemented and evaluated in MATLAB. For the implementation and evaluation of the NN based data estimators, the deep learning framework PyTorch 1.8.0 [43] is employed, since this framework provides support to compute on CUDA [44] capable Nvidia graphics processing units (GPUs), which in turn accelerates the training and testing of the NNs many times over. The training and test data are generated in MATLAB and forwarded to the Python framework, to train and evaluate the NNs. In the case of coded transmission, the sequence of log-likelihood ratios provided by the NN based data estimators is decoded by a Viterbi
decoder implemented in C and compiled as MATLAB executable. To this end, also the MATLAB Engine API for Python [45] is employed.

As already mentioned, in this work the data estimators are evaluated for two UW-OFDM systems with different system dimensions. For the small sized system I the MMSE estimator can be evaluated and thus a benchmark for the achievable BERs is available. However, the number of \( N_d = 8 \) data subcarriers for this system setup is too small to be relevant for real-world communication systems. Hence, the data estimators are also investigated for the practically relevant system II, where \( N_d = 32 \). Note, that system I and system II correspond to the system setting ‘ML’ in [10] and setting ‘F’ in [11], respectively. The settings of both systems are summarized in Table 4.1.

Table 4.1.: Settings of investigated UW-OFDM systems

<table>
<thead>
<tr>
<th></th>
<th>System I</th>
<th>System II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nr. data subcarriers</td>
<td>( N_d )</td>
<td>8</td>
</tr>
<tr>
<td>Nr. redundant subcarriers</td>
<td>( N_r )</td>
<td>4</td>
</tr>
<tr>
<td>Nr. zero subcarriers</td>
<td>( N_z )</td>
<td>0</td>
</tr>
<tr>
<td>Nr. pilot subcarriers</td>
<td>( N_p )</td>
<td>0</td>
</tr>
<tr>
<td>Length of the UW</td>
<td>( N_u )</td>
<td>4</td>
</tr>
<tr>
<td>DFT size</td>
<td>( N )</td>
<td>12</td>
</tr>
<tr>
<td>redundant subcarrier indices</td>
<td>( I_r )</td>
<td>{1, 4, 7, 10}</td>
</tr>
<tr>
<td>zero subcarrier indices</td>
<td>( I_z )</td>
<td>{}</td>
</tr>
<tr>
<td>pilot subcarrier indices</td>
<td>( I_p )</td>
<td>{}</td>
</tr>
<tr>
<td>interleaving factor</td>
<td>( K )</td>
<td>8</td>
</tr>
</tbody>
</table>

The simulations are run on a server equipped with an AMD Ryzen Threadripper PRO 3975WX, 32-core @ 3.49 GHz CPU, 256 GB main memory and an Nvidia Quadro RTX 5000 with 16 GB memory.

To give a rough idea how time intensive the training of the networks was, approximate training times are given in the following. The optimization of the networks trained for the small system I takes between 6 and 8 hours for each hyperparameter setting, depending on which of the NN based data estimators is trained. However, the optimization of the DetNet trained for system II takes around 24 hours for each hyperparameter configuration. As already all available computational resources are exhausted and the whole training procedure is performed on GPUs, these training times currently cannot be reduced. However, also the authors of [2] mention that the training of the DetNet is a very time consuming task. Their training procedure for a system which is a bit smaller than system II took them 3 days on an Intel i7-6700 processor for a single hyperparameter setting, confirming that the training on GPUs is a far better and faster approach.
For all simulation settings the hyperparameters of the DetNet, the Attention Detector and the FCNN Detector are optimized by means of grid search and the found locally optimal hyperparameter settings are given in Table A.1, A.2 and A.3, respectively. Considering the training times from the last paragraph for a single hyperparameter configuration reveals that the hyperparameter optimization is a very time-consuming task. Thus, a reasonable selection of simulation cases to be examined is crucial in order to be able to thoroughly investigate the impact of the different combinations of the communication system’s settings on the behavior of the NN based data estimators on the one hand and to avoid an exploding simulation time on the other hand.

4.2. Perfect Channel Knowledge

In this section, the CIR is assumed to be perfectly known to the receiver. Further, the CIR is constant for one transmitted data burst and changes from burst to burst according to the channel model described in Section 2.1.2, independently from its previous realizations.

4.2.1. Systematic UW-OFDM

The first investigation of the NN based data estimators is conducted for system I with the systematic UW-OFDM transmission scheme, uncoded transmission and a QPSK modulation alphabet.

![Figure 4.1.: Systematic UW-OFDM system I, uncoded case, QPSK.](image-url)
As shown in Figure 4.1, all three NN based data estimators show a similar behavior, but the DetNet is the best performing one. Further, the Attention Detector outperforms the FCNN Detector over the whole $E_b/N_0$ range. This justifies the idea to utilize the attention mechanism in order to exploit correlations of the rows of $H^T H$ and increase the performance compared with the FCNN. However, the Attention Detector is outperformed by the DetNet by 0.3 dB at a BER of $10^{-4}$. The better overall performance of the DetNet results mainly from the fact that the DetNet benefits more from preconditioning than the Attention Detector does. Before preconditioning was applied, the Attention Detector was the best performing NN based data estimator for this simulation case. More details on the impact of preconditioning on the performance of the DetNet can be found in Section 4.4. Nevertheless, this is a promising result and suggests that the Attention Detector could be applied for data estimation in MIMO communication systems, like multi-element antenna communication systems, where the columns of the channel matrix are far more correlated than it is the case for systematic UW-OFDM systems. As already mentioned in Section 3.4, according to [2] and [3], many NN based data estimators and also the DetNet suffer from channel matrices with highly correlated columns. The Attention Detector, in turn, should actually benefit from high correlations. Hence, the Attention Detector could be applied for data estimation in these communication systems in future work.

The results, shown in Figure 4.1, also reveal an unfavorable property of the NN based data estimators, namely a flattening out of their BER curves at lower BERs and thus higher values for $E_b/N_0$. While the BER curves of the traditional data estimators decrease even steeper at higher SNRs, this is not the case for the NNs. This disadvantageous effect can be mitigated amongst others by preconditioning, but unfortunately could not be eliminated. As evident from the tables in Appendix A, the upper limit of the training $E_b/N_0$ range exceeds the upper limit of the test $E_b/N_0$ range by 1 dB to 2 dB, which is a further remedy to reduce the flattening out to a minimum. Also the increased amount of deep fading channels in the training set helps to reduce this problem. Further approaches unfortunately did not show the desired effect. However, this behavior can also be observed for the scenario of general MIMO channels, for which the DetNet originally was proposed, when the SNR range becomes wider. Thus, this behavior seems to be a general problem of NN based data estimators and not only specifically for the UW-OFDM system. More investigations on this problem are left for future work.

The comparison to the traditional data estimators shows that the DFE and the DetNet feature a quite similar performance, the DetNet can even outperform the DFE between $E_b/N_0 = 10$ dB and $E_b/N_0 = 15$ dB. At higher SNRs the DFE, however, performs better than the DetNet. The LMMSE estimator, in turn, is clearly outperformed by the DetNet, at a BER of $10^{-4}$ their BER curves are separated by 2.3 dB.
4.2. Perfect Channel Knowledge

4.2.2. Non-Systematic UW-OFDM

The investigations in [10] and [11] revealed that at a fixed SNR with non-systematic UW-OFDM far lower BERs can be attained than with systematic UW-OFDM. Hence, from now on the focus lies on non-systematic UW-OFDM systems, as these results are then also relevant for comparisons to the state-of-the-art CP-OFDM systems. Note, that in this work a comparison of UW-OFDM and CP-OFDM is not conducted, but can be found in [11].

4.2.2.1. Uncoded Case

The simulation results for system I with uncoded data transmission and QPSK modulation alphabet are shown in Figure 4.2. As already elaborated in Section 3.4, in non-systematic UW-OFDM the correlations of the rows of $H^T H$ that can be exploited by the Attention Detector are weak. However, it still performs slightly better than the FCNN Detector, especially at higher SNRs. As shown in Figure 4.2, the DetNet is again the best performing of the investigated NN based data estimators for non-systematic UW-OFDM. The DetNet slightly outperforms the DFE in the $E_b/N_0$ range from 8 dB to around 14 dB and is in this range also very close to the, with respect to the BER, optimal MMSE estimator. Further, at a BER of $10^{-4}$ the DetNet achieves a gain of 3.4 dB over the LMMSE estimator, which is even higher at lower BERs. It also has to be considered that the generator matrix in non-systematic UW-OFDM is optimized in order to achieve the best possible LMMSE estimation performance, from which of course also the DFE benefits. Thus, it is even more remarkable that the DetNet can perform approximately as good as the DFE and can clearly outperform the LMMSE estimator. The Attention Detector and the FCNN Detector, in turn, gain 2.9 dB and 2.6 dB in performance over the LMMSE estimator at a BER of $10^{-4}$, respectively.
Although the FCNN cannot achieve the performance of the DetNet, it is still shown with this result that an FCNN can also handle the varying channel case when $H^T H$, reshaped column-wise to a vector and concatenated with $H^T y$ is chosen as input instead of $H$ and $y$. This also highlights that pre-processing (in this case a multiplication by $H^T$) is very crucial for NNs and could make the difference whether or not they work properly.

However, it can again be observed that the BER curves of all three NN based data estimators do not decrease in the same manner as the traditional estimators do, since they flatten out at lower BERs and thus at the higher $E_b/N_0$ values of the investigated $E_b/N_0$ range. As already mentioned in Section 4.2.1, some remedies have been found to mitigate this problem, but they do not solve it completely. In this context there also has to be mentioned that a too wide SNR range for the training sets of the NNs leads to a severe degradation in estimation performance. Thus, the training SNR range has to be adapted to the test SNR range. Approaches to incorporate the noise variance into the estimation process of the NNs did not succeed, but could be the key for NN based data estimators that can be trained for a wide SNR range. This could be part of future research.

The above simulation results, shown in Figure 4.2, indicate that the DetNet performs best for non-systematic UW-OFDM. Hence, the focus for the remaining simulations in Section 4.2.2, which address system II, coded transmission and the BPSK modulation alphabet, lies on the DetNet to rank the performance of NN based data estimators compared with traditional methods, but to significantly reduce the time needed for hyperparameter optimization.
4.2. Perfect Channel Knowledge

For system II, the relative differences in performance between the DFE, the LMMSE estimator and the DetNet do not change substantially compared with those for system I. As shown in Figure 4.3, the DetNet can slightly outperform the DFE at lower SNRs, while from $E_b/N_0 = 22$dB on the DFE is the better performing estimator. At a BER of $10^{-4}$ the DetNet achieves a substantial performance gain of 3.3 dB over the LMMSE estimator.

![Figure 4.3.: Non-systematic UW-OFDM system II, uncoded case, QPSK.](image)

As elaborated in Section 2.2.3, a real-valued data vector, which is the case for a BPSK modulation alphabet, in a complex-valued environment, i.e. the CIR and the WGN are both complex-valued, requires special treatment for linear estimation methods. The incorporation of this knowledge by replacing the LMMSE by the WLMMSE estimator leads to a considerable performance gain. Since a linear estimation step is carried out in each iteration of the DFE, it of course also benefits from utilizing the WLMMSE estimation in every iteration step. The knowledge about real valued data vectors is implicitly provided to NN based data estimators, as they are trained exclusively with them. Further, the optimal hyperparameter settings have to be found for this case, too. However, the fact of improper data vectors does not have to be incorporated explicitly into the NN architecture. Thus, it is interesting to investigate how the NN based estimators, more specifically the DetNet, perform for data vectors with improper statistics compared with the WLMMSE DFE and the WLMMSE estimator, which explicitly incorporate this additional knowledge.
As the results in Figure 4.4 show, the DetNet does not lose performance in comparison to the DFE which utilizes a WLMMSE estimation in each iteration. Hence, the NN based data estimators can handle both modulation alphabets that result in data vectors with proper and modulation alphabets that lead to data vectors with improper statistics.

4.2.2.2. Coded Case

In this section, the practically very important case of channel coded data transmission is investigated. The results for coded transmission for system I, where it is also computationally tractable to simulate the MMSE estimator, and system II are shown in Figure 4.5 and 4.6, respectively. The majority of the hyperparameter settings of the DetNet for the coded simulation cases coincide with those for the corresponding uncoded simulation cases, however, the learning rate and the training SNR ranges have to be chosen differently (c.f. Appendix A).
Firstly, only a very small performance difference between the traditional estimators can be observed from Figure 4.5 and 4.6. While in the uncoded case and system I the MMSE estimator achieves at a BER of $10^{-4}$ a gain of 3.5 dB over the LMMSE, and even more at lower BERs (c.f. Section 4.2.2.1), in the coded case the gain is only 0.3 dB at a BER of $10^{-4}$. Due to this observation, a high performance gain of the DetNet
against the LMMSE estimator cannot be expected in the coded case. Surprisingly, for coded transmission the DetNet performs even worse than the LMMSE estimator. While the DetNet performs considerably worse than the LMMSE estimator for system I, the performance loss against the LMMSE estimator and the DF equalizer for system II below a BER of $10^{-2}$ is small and nearly constant around 0.3 dB. That is, the performance loss of the DetNet in comparison to the LMMSE estimator for system II is less than for system I, but a performance gain cannot be achieved at all.

These surprising results need to be further investigated. Due to the large increase in performance by channel coding, the coded transmissions are simulated at low SNRs, where the estimators’ performances are quite similar, i.e. the gaps between the BER curves are also small for uncoded transmission. This is depicted in Figure 4.7 and 4.8 for system I and system II, respectively. Further, for the coded case, the information loss in the LLRs of the LMMSE estimates defined in (2.92) due to the suboptimal linear estimation seems to be very small in comparison to the exact LLRs of the MMSE estimates defined in (2.77) and (2.78), which is remarkable. Although the DetNet outperforms the LMMSE estimator for all the simulations with uncoded transmission, it cannot achieve the performance of the LMMSE estimator in the coded case. Note, that the LLRs of the DetNet are computed from approximate posterior probabilities. As will be presented in Section 4.5, the corresponding symbol estimates of the DetNet are not distributed in exactly the same way as those of the MMSE estimator. That is, at low SNRs (c.f. Figure 4.11 and 4.14) the (QPSK) estimates of the DetNet are still nearly exclusively located on the edges of the square connecting the constellations points in the complex plane, while a few estimates of the MMSE estimator are located also inside this square, which represent the data symbol estimates where the estimates for both bits of the symbol are rather unsure. This leads to the conclusion that the approximate posterior probabilities provided by the DetNet are not as reliable as desired and thus also the performance of the Viterbi channel decoder suffers, since it highly benefits from (reliable) soft information.

Further investigations and options to make the approximate posterior probabilities of the DetNet more reliable or to find different approaches for channel coding that work well for NN based data estimators could be part of future work.
Figure 4.7.: Non-systematic UW-OFDM system I, uncoded case, QPSK, at low SNRs.

Figure 4.8.: Non-systematic UW-OFDM system II, uncoded case, QPSK, at low SNRs.
4. Results

4.3. Imperfect Channel Knowledge

In all the simulations of the previous sections channel knowledge in terms of a CIR is assumed to be perfectly known. However, in practice the true CIR is unknown and has to be estimated as well. Hence, the impact of channel estimation errors on the performance of the data estimation methods is of high practical relevance. In this section, the impact of imperfect channel knowledge on the traditional data estimators and the NN based data estimators is examined and compared. The simulation setting for this investigation is equivalent to that of the simulation of system I in Section 4.2.2.1, namely the non-systematic UW-OFDM transmission scheme with uncoded transmission and QPSK as modulation alphabet. Thus, also the hyperparameters of the NN based data estimators are the same as in Section 4.2.2.1. However, it is important to note that they have been trained with the estimated CIRs for the evaluation of their performance with imperfect channel knowledge, as training with the true CIRs and testing with the estimated ones would degrade their performance on the test set.

In Figure 4.9 the BER curves for all data estimators in case of perfect channel knowledge (the solid lines) and imperfect channel knowledge (the dashed lines) are plotted. The results reveal that an imperfect CIR impacts the traditional equalizers and the

![Figure 4.9](image-url)
NN based data estimators in approximately the same scale. More specifically, the performance of all the data estimators degrades by around 0.7 dB over the whole $E_b/N_0$ range investigated due to imperfect channel knowledge. Hence, it can be concluded that imperfect channel knowledge influences the performance of NN based data estimators neither more nor less than that of traditional data estimators.

4.4. Impact of Pre-Processing

To give an impression how the pre-processing of the data, more specifically the data normalization and the preconditioning, influences the performance of the DetNet, the BER curves for the DetNet using its original implementation, the DetNet with data normalization and the DetNet with both data normalization and preconditioning are compared.

![BER curves comparison](image)

**Figure 4.10.: Influence of pre-processing on the performance of the DetNet.**

In Figure 4.10 the results of the simulation for system I with non-systematic UW-OFDM, uncoded transmission and the QPSK modulation alphabet are shown. It is clearly observable, that the DetNet without normalization and without preconditioning performs poor, even worse than the LMMSE estimator. Further the flattening out of the BER curve occurs already at a BER of around $3 \cdot 10^{-4}$. The data normalization leads to a huge
4. Results

performance gain, but the DetNet is still 0.7 dB behind the performance of the DFE at a BER of $10^{-4}$ and 1.4 dB at a BER of $10^{-5}$. Preconditioning brings another performance gain, such that the DetNet shows around the same performance as the DFE, although it flattens out slightly below a BER of $10^{-5}$.

4.5. Constellation Diagrams of the Data Symbol Estimates

Besides the BER curves as performance measure also the distributions of the estimates of the data estimators are of interest. These distributions can be visualized by plotting the data symbol estimates in the complex plane, also termed constellation diagram or I/Q (in-phase/quadrature-phase) diagram.

It is desirable that, for a fixed transmitted data symbol, the distribution of the symbol estimates is centered around a point close the constellation point of the transmitted symbol and has a low variance. The distributions of the estimates of the MMSE estimator, the DetNet\(^1\) and the LMMSE estimator, given that the transmitted data symbol is $\frac{1}{\sqrt{2}}(1+j)$, are visualized in Figure 4.11, 4.12 and 4.13 at $E_b/N_0 = 4$ dB, 8 dB and 14 dB, respectively. Since the DFE performs an LMMSE estimation in every iteration, the estimates for a single symbol are distributed in a similar way as those of the LMMSE and thus a visualization of the distribution of the estimates of the DFE is omitted. The histograms in all the following plots are computed from the estimates of over 9 million data symbols transmitted over different channels, to obtain statistically significant results. In the I/Q diagrams a subset of 5000 samples is depicted. Furthermore, the constellation points of the normalized QPSK alphabet are marked by red crosses.

\(^1\)Since for all three NN based data estimators one-hot encoding in combination with a quadratic loss function is employed, the distributions of their estimates show approximately the same behavior and thus only the results for the DetNet are plotted.
4.5. Constellation Diagrams of the Data Symbol Estimates

Figure 4.11.: Distribution of the conditional data symbol estimates at $E_b/N_0 = 4$ dB.

Figure 4.12.: Distribution of the conditional data symbol estimates at $E_b/N_0 = 8$ dB.

Figure 4.13.: Distribution of the conditional data symbol estimates at $E_b/N_0 = 14$ dB.

From Figure 4.11, 4.12 and 4.13 the fundamentally different distribution of the LMMSE estimates in comparison to the MMSE and DetNet estimates becomes obvious.
4. Results

As already described in Section 2.2.3.1, the conditional LMMSE estimates are (approximately) Gaussian distributed, which is now justified by the given constellation diagrams. Further, the component-wise conditional biasedness of the LMMSE estimator, which is taken into account by the scaling factor \( \alpha_k \) in the computation of the LLRs, is also clearly evident in Figure 4.11, 4.12 and 4.13, since the centers of the point clouds are located apart from the constellation point \( \frac{1}{\sqrt{2}}(1 + j) \).

The PDF of the MMSE estimator, however, is not Gaussian at all. For the simple case of an AWGN channel without multipath propagation its PDF \( p(\hat{d}_i|d_i) \) can be computed analytically, which reveals that the PDF is concentrated at the constellation points and decreases extremely steeply, such that it is close to zero elsewhere within the square connecting the constellation points. Further, the PDF is identically zero outside this square, which is obvious since the MMSE estimates \( \text{Re}\{\hat{d}_i\} \) and \( \text{Im}\{\hat{d}_i\} \) are the posterior means \( E_{d_i|y}[\text{Re}\{d_i\}|y] \) and \( E_{d_i|y}[\text{Re}\{d_i\}|y] \), respectively. This implies that the estimates must lie between the constellation points, which has already been described in more detail in Section 2.3.5. For the multipath channel, an analytic expression of \( p(\hat{d}_i|d_i) \) cannot be easily computed, however, the simulation results indicate that it has very similar behavior, which is visible in the histograms of Figure 4.11, 4.12 and 4.13. As shown in Figure 4.11, a few estimates of the MMSE estimator can also occur which are far apart from the constellation point, although the histograms indicate that still the vast majority of estimates is located close to the constellation point. At increasing SNRs the estimates become distributed nearly solely in horizontal or vertical direction, as plotted in Figure 4.12. Finally, at \( E_b/N_0 = 14 \) dB, no more “outliers” of the MMSE estimates are visible in Figure 4.13.

As already mentioned in Section 2.3.5, the data symbol estimates of the DetNet are approximately distributed like the MMSE estimates. Due to the imperfect training of the DetNet, also values slightly outside the square connecting the constellation points may occur. Remarkably, also at lower \( E_b/N_0 \) values the DetNet estimates are concentrated even more on the lines connecting the constellation points than it is the case for the MMSE estimates (c.f. Figure 4.11). This observation emphasizes that the behavior of the NN based data estimators is still somewhat different to that of the MMSE estimator. As shown in Figure 4.11 and 4.12, especially at low SNRs, estimates far from the constellation point \( \frac{1}{\sqrt{2}}(1 + j) \) can still occur, the number of these “outliers” is, however, very small as indicated by the histograms.

Finally, in Figure 4.14, 4.15 and 4.16 the constellation diagrams of the MMSE, DetNet and LMMSE estimates at \( E_b/N_0 = 4 \) dB, 8 dB and 14 dB, respectively, are also shown for a uniform prior data symbol distribution to give an impression on how the constellation diagrams look like for this general case.
4.5. Constellation Diagrams of the Data Symbol Estimates

Figure 4.14.: Distribution of the data symbol estimates for uniform prior data symbol probabilities at $E_b/N_0 = 4$ dB.

Figure 4.15.: Distribution of the data symbol estimates for uniform prior data symbol probabilities at $E_b/N_0 = 8$ dB.

Figure 4.16.: Distribution of the data symbol estimates for uniform prior data symbol probabilities at $E_b/N_0 = 14$ dB.
4. Results

4.6. Complexity Analysis

Finally, a very brief overview of the inference complexity of the three NN based data estimators as well as the LMMSE estimator and DFE is presented to give an impression how many computational operations are required to estimate an UW-OFDM data vector. It is important to consider that NNs additionally need to be trained, which is usually computationally expensive. However, for the training huge computational resources can be provided, while in the inference phase of the trained networks, e.g. for an application on a mobile device, comparably limited computational resources are available. Hence, only the inference complexity is compared. In this analysis only the number of scalar multiplications is investigated, as this is the computationally most intensive scalar operation. For the variable names it is referred to the tables of hyperparameter settings in Appendix A, or Section 3.2, 3.3 and 3.4 for the DetNet, the FCNN Detector and the Attention Detector, respectively.

The determination of the number of multiplications required by the FCNN Detector $M_{\text{FCNN}}$ is straightforward and can be expressed as

$$M_{\text{FCNN}} = \left( N_d' + N_d^2 \right) d_h + \frac{d_h^2}{d} L + N_d' |S_{re}| + (2(N_d + N_r)N_d' + 2N_d'(N_d + N_r)), \quad (4.1)$$

where the last term in brackets accounts for the computation of the inputs $H^T H$ and $H^T y$. The complexity of the FCNN Detector is dominated by the term $d_h^2 L$, as the number of hidden neurons per layer is chosen to be larger than the input and the output dimension.

For the DetNet, in a first step, the number of scalar multiplications in a single layer is determined. The number of multiplications to compute $H^T H d_k$ for a given $H^T H$ is $N_d'^2$. Further, the multiplications of the learned parameters $\delta_k$ and $\delta_k^2$ with their corresponding vectors take another $N_d'$ scalar multiplications for both of them. For the FCNN with a single hidden layer $d_h d_qv$ scalar multiplications in the input layer and $(N_d'|S_{re}| + d_v) d_h$ in the hidden layer have to be performed. The one-hot demapping and the residual connections require another $N_d'|S_{re}|$ and $(N_d' + d_v)$ scalar multiplications, respectively. Hence, in every DetNet layer

$$M_{\text{DetNet},i} = N_d'^2 + 2N_d' + d_h d_qv + (N_d'|S_{re}| + d_v) d_h + N_d'|S_{re}| + (N_d' + d_v) \quad (4.2)$$

scalar multiplications are performed, which leads to an overall complexity of

$$M_{\text{DetNet}} = LM_{\text{DetNet},i} - N_d'|S_{re}| + (2(N_d + N_r)N_d'^2 + 2N_d'(N_d + N_r)). \quad (4.3)$$

The subtracted term considers that in the last layer no one-hot decoding is conducted anymore and the last term in brackets accounts for the computation of the inputs. The complexity of the DetNet can roughly be assessed by computing $L(N_d'^2 + d_h d_qv + d_v d_h + N_d'|S_{re}| d_h) + 2N_d N_d'^2$, since these are the dominating terms for larger dimension UW-
4.6. Complexity Analysis

OFDM systems. The dimensions $d_h$, $d_v$ and $d_{qv}$ of the DetNet of course scale with the system dimensions.

To determine the complexity of the Attention Detector, in a first step, a single encoder layer is investigated. The mappings of the inputs to the queries $Q$, keys $K$ and values $V$ require $(N_d' + 1)^2 N_d'$ multiplications each. Further, for the computation of all self-attention scores $\alpha_{ij}$ another $3(N_d' + 1)N_d'$ multiplications are needed. To obtain the output of the self-attention layer the values have to be weighted by the attention scores. In this weighing process $(N_d' + 1)N_d'$ scalar multiplications are conducted. Moreover, in the single layer FCNN $2d_{h,enc}(N_d' + 1)N_d'$ multiplications and in each of the layer normalizations $(N_d' + 1)N_d'$ multiplications are performed. The residual connections introduce another $(N_d' + 1)N_d'$ scalar multiplications. Hence, the number of scalar multiplications conducted in one encoder layer is

$$M_{\text{Att,enc},i} = 3(N_d' + 1)^2 N_d' + 8(N_d' + 1)N_d' + 2d_{h,enc}(N_d' + 1)N_d'. \quad (4.4)$$

On top of the encoder layers a shallow FCNN is utilized for which

$$M_{\text{Att,fcnn}} = (N_d' + 1)N_d'd_{h,fcnn} + L_{f\text{cnn}}d_{h,fcnn}^2 + d_{h,fcnn}(N_d'|\mathbb{S}_{re|}) \quad (4.5)$$

multiplications are required. Therefore, the Attention Detector has a complexity of

$$M_{\text{Att}} = L_{\text{enc}}M_{\text{Att,enc},i} + M_{\text{Att,fcnn}} + (2(N_d + N_r)N_d'^2 + 2N_d'(N_d + N_r)) \quad (4.6)$$

scalar multiplications. The complexity of the Attention Detector is dominated by the terms $L_{\text{enc}}(3N_d'^3 + 14N_d'^2 + 2d_{h,enc}N_d'^2) + L_{f\text{cnn}}d_{h,fcnn}^2$.

To determine the complexity of the LMMSE estimator, the number of multiplications needed to compute a matrix inverse has to be determined. In this work, the same approach as in [10] is chosen, where a Cholesky decomposition is utilized to compute the matrix inverse. According to [10], where in turn is referred to [46], the number of complex multiplications (divisions are also counted as multiplications) needed to solve

$$AX = B \quad (4.7)$$

for $X$, where $X \in \mathbb{C}^{R \times C}$, $A \in \mathbb{C}^{R \times R}$ and $B \in \mathbb{C}^{R \times C}$, is $\frac{1}{6}R^3 + R^2C + RC$. Inserting the identity matrix for $B$, yields the desired inverse of $A$.

In to following, optimizations to reduce the complexity of the computations are not considered, as this was not the case for the NNs, too. In a first step, the number of complex scalar multiplications needed to obtain the estimator matrix, which has to be done once for every data transmission burst, is regarded. To compute $H^dH$ in (2.82), where $H = \tilde{H}G$, $(N_d + N_r)N_d^2$ complex scalar multiplications are needed. Further, for the inverse in (2.82), which is an $N_d \times N_d$ matrix, $\frac{7}{6}N_d^3 + N_d^2$ complex multiplications and for the matrix product of the inverse and $H^d$ another $(N_d + N_r)N_d^2$ complex multiplications
are required. Hence,
\[
M_{\text{LMMSE,E},\text{cpx}} = 2(N_d + N_r)N_d^2 + \frac{7}{6}N_d^3 + N_d^2
\]
\[
= \frac{19}{6}N_d^3 + 2N_d^2N_r + N_d^2
\]  
(4.8)
complex multiplications are required to compute the LMMSE estimator matrix. As the results are compared with the NNs’ complexities, the number of real scalar multiplications are needed. As in [10], it is assumed, that four real multiplications account for one complex multiplication, although a complex multiplication can also be realized by three real multiplications, albeit at the cost of more additions. Therefore, the number of real multiplications needed to obtain the estimator matrix is
\[
M_{\text{LMMSE,E,real}} = 4M_{\text{LMMSE,E,}\text{cpx}} = \frac{38}{3}N_d^3 + 8N_d^2N_r + 4N_d^2.
\]  
(4.9)
For the equalizing operation for one UW-OFDM symbol, i.e. when the LMMSE estimator matrix is already computed,
\[
M_{\text{LMMSE,Ey,}\text{cpx}} = (N_d + N_r)N_d
\]  
(4.10)
complex multiplications or
\[
M_{\text{LMMSE,Ey,real}} = 4M_{\text{LMMSE,Ey,}\text{cpx}} = 4(N_d + N_r)N_d
\]  
(4.11)
real multiplications are required.

The complexity of the LMMSE estimator is dominated by \(\frac{38}{3}N_d^3\) for the computation of the estimator matrix (which has to be done only once per data burst).

For the DFE, in a first step the operations which have to be conducted once every transmission burst are regarded. Firstly, the matrix \(\mathbf{H}^H\mathbf{H}\) has to be computed once, where \((N_d+N_r)N_d^2\) complex multiplications are needed. Further an estimator row-vector \(\mathbf{e}_i^H\) has to be computed for every iteration step. Since the system dimensions are reduced by one after every iteration step, the complexity of determining \(\mathbf{e}_i^H\) for an iteration step of the DFE is described in the following for a system matrix \(\mathbf{H}\) with \((N_d+N_r)\) rows and \(C\) columns and a data vector of length \(C\). In order to determine which element of \(\mathbf{d}\) is estimated in an iteration step, the diagonal elements of the error covariance matrix \(\mathbf{C}_{ee}\) have to be computed. For a given \(\mathbf{H}^H\mathbf{H}\), the inverse term in (2.83) requires \(\frac{7}{6}C^3 + C^2\) complex multiplications. The multiplication of the diagonal elements of the inverse term by the scaling factor \(N\sigma_n^2\) requires \(C\) further multiplications. To compute the estimator row-vector \(\mathbf{e}_i^H\), the inverse term from the covariance matrix can of course be utilized again. Hence, only the \(i\)-th row of the inverse term has to be multiplied by \(\mathbf{H}^H\), for
which \((N_d + N_r)C\) complex multiplications are required. Consequently,

\[
M_{\text{DFE,E,cpx}} = (N_d + N_r)N_d^2 + (N_d + N_r) + \sum_{C=2}^{N_d} \frac{7}{6} C^3 + C^2 + C + (N_d + N_r)C
\]

\[
= \frac{7}{24} N_d^4 + \frac{29}{12} N_d^3 + \frac{43}{24} N_d^2 + \frac{3}{2} N_d^2 N_r + \frac{2}{3} N_d + \frac{1}{2} N_d N_r - \frac{19}{6}
\]

(4.12)

complex multiplications or

\[
M_{\text{DFE,E,real}} = 4M_{\text{DFE,E,cpx}}
\]

\[
= \frac{7}{6} N_d^4 + \frac{29}{3} N_d^3 + \frac{43}{6} N_d^2 + 6N_d^2 N_r + \frac{8}{3} N_d + 2N_d N_r - \frac{38}{3}
\]

(4.13)

real multiplications are required to compute the \(N_d\) estimator rows \(e_i^H\).

To determine the estimate \(\hat{d}_i\), the estimator row \(e_i^H\) has to be multiplied by \(y\), which requires \(N_d + N_r\) complex multiplications. Further, for the removal of the influence of the estimated symbol \(\hat{d}_i\) on \(y\) another \(N_d + N_r\) multiplications have to be taken into account after every iteration step. Hence,

\[
M_{\text{DFE,Ey,cpx}} = 2N_d(N_d + N_r)
\]

(4.14)

complex multiplications or

\[
M_{\text{DFE,Ey,real}} = 8N_d^2 + 8N_d N_r
\]

(4.15)

real multiplications are needed to obtain \(\hat{d}\).

The complexity of the DF estimator is dominated by \(\frac{7}{6} N_d^4\) for the computation of the estimator row-vectors (which has to be done only once per data burst).

For the purpose of comparing the complexity of the three different NN based data estimators as well as the LMMSE estimator and the DFE, a numerical example is presented in the following. As exemplary case system I with the non-systematic UW-OFDM transmission scheme, uncoded transmission and a QPSK alphabet is chosen. Hence, \(N_d = 8\), \(N_r = 4\), \(N'_d = 16\) and \(|S_{re}| = 2\). For the hyperparameter settings of the NN based data estimators it is referred to Appendix A. The results are shown in Table 4.2, which highlight that the DetNet is by far the lowest complex of the investigated NNs. This emphasizes that the deep unfolding approach, by which the structure of the DetNet is inferred, is a very promising method to incorporate model knowledge into NNs, as far less complex and better performing networks can be obtained, compared with e.g. an FCNN. Note however, that it is not in the scope of this work to find very low complex NNs and thus the inference complexity of the investigated architectures has not been considered in any step of network design or optimization. The comparison with the complexity of the traditional estimators reveals, that there is a large gap between the complexities of the NN based data estimators and the traditional data estimators (c.f. Table 4.3 for the traditional data estimators’ complexities). Hence, finding methods to
4. Results

reduce the complexity of the NN based data estimators should be part of future research. Here, NNs that can cope with complex inputs could be helpful, as then the complex-to-real mapping of the system can be omitted. However, standard training methods for NNs can then not be applied anymore.

Table 4.2.: Number of multiplications of the NN based data estimators for system I with the non-systematic UW-OFDM transmission scheme, uncoded transmission and QPSK alphabet

<table>
<thead>
<tr>
<th>Nr. multiplications rounded to hundreds</th>
</tr>
</thead>
<tbody>
<tr>
<td>DetNet</td>
</tr>
<tr>
<td>Attention Detector</td>
</tr>
<tr>
<td>FCNN Detector</td>
</tr>
</tbody>
</table>

Table 4.3.: Number of multiplications of the traditional data estimators for system I with the non-systematic UW-OFDM transmission scheme, uncoded transmission and QPSK alphabet

<table>
<thead>
<tr>
<th>Nr. multiplications rounded to hundreds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimator determination</td>
</tr>
<tr>
<td>LMMSE estimator</td>
</tr>
<tr>
<td>DFE</td>
</tr>
</tbody>
</table>

The same analysis is conducted, for the large system II and QPSK alphabet, where \( N_d = 32 \), \( N_r = 12 \), \( N'_d = 64 \) and \( |S_{re}| = 2 \). As shown in Table 4.4 and 4.5, the traditional methods again require less multiplications, however, the relative gap between the DFE and the DetNet decreases considerably for system II.

Table 4.4.: Number of multiplications of the NN based data estimators for system II with the non-systematic UW-OFDM transmission scheme, uncoded transmission and QPSK alphabet

<table>
<thead>
<tr>
<th>Nr. multiplications rounded to hundreds</th>
</tr>
</thead>
<tbody>
<tr>
<td>DetNet</td>
</tr>
</tbody>
</table>

84
Table 4.5.: Number of multiplications of the traditional data estimators for system II with the non-systematic UW-OFDM transmission scheme, uncoded transmission and QPSK alphabet

<table>
<thead>
<tr>
<th>Estimator determination</th>
<th>Nr. multiplications rounded to hundreds</th>
<th>Equalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMMSE estimator</td>
<td>451900</td>
<td>5700</td>
</tr>
<tr>
<td>DFE</td>
<td>1622000</td>
<td>11300</td>
</tr>
</tbody>
</table>
5. Conclusion

In this work, the application of neural networks on data estimation in communication systems, more specifically in UW-OFDM systems, was investigated. The aim was to mimic optimal model-based data estimators by neural networks to achieve a close to optimal performance with a far lower computational complexity than that of the optimal estimators. To this end, three different neural network architectures were examined.

Firstly, the DetNet, that was proposed in [2] for the data estimation in a general MIMO system, was applied to the UW-OFDM system. Due to the special structure of the matrix $H$ in comparison to the case of a general MIMO system, the original DetNet architecture without any data normalization did not lead to satisfying results. A thorough investigation of the reasons for this revealed that a proper data normalization on the one hand and preconditioning on the other hand leads to a huge performance increase such that the LMMSE estimator can be significantly outperformed in the uncoded case.

Secondly, it was shown in this work that a simple FCNN can also deal with varying channels, although a contrary statement was made e.g. in [2]. This could be achieved by utilizing $H^T H$ and $H^T y$ instead of $H$ and $y$ as inputs, which is motivated by sufficient statistics and an analysis of the terms that have to be computed for the optimal MMSE estimator. Although the performance in terms of the BER of the DetNet could not be achieved by the FCNN Detector, the LMMSE estimator could still be clearly outperformed. Hence, the insight that pre-processing of the input can make or break the functionality of a neural network for data estimation is at least an interesting theoretical finding.

Thirdly, the fact that the rows of the matrix $H^T H$ for systematic UW-OFDM are correlated lead to the decision to utilize the attention mechanism for data estimation in order to improve the estimation results compared to the simple FCNN. The comparison of the resulting BERs of the Attention Detector and the FCNN Detector justified this decision. However, in non-systematic UW-OFDM the columns of $H$ are nearly orthogonal and thus the attention mechanism can hardly benefit from correlations of the rows of $H^T H$. From these investigations it can be concluded that the Attention Detector could be applied for the data estimation in many MIMO communication systems where the columns of the channel matrix are correlated. According to [2] and [3], existing neural network based data estimators like the DetNet suffer from these correlations, while the Attention Detector benefits contrariwise. An examination of possible application cases of the attention mechanism in communication systems could be part of future work.

Although the performance results of the NN based data estimators are promising, there
are still many open questions and plenty of room for improvement. As observed from the BER curves of the NN based data estimators, their performance flattens out at low BERs and thus high SNRs while model-based methods do not show this behavior. In this context there has also to be mentioned that the neural networks can only be trained for a limited range of SNRs, since their performance degrades significantly in case of a training over a too wide test SNR range. These two issues definitely have to be examined in more detail and the key to solve them could be to successfully incorporate knowledge on the variance of the AWGN into the networks, which could not be achieved in this work. Furthermore, the reliability of the LLRs provided by the NNs has to be improved or the channel decoder has to be adapted or incorporated into the training of the NNs to guarantee a good performance also for coded transmission. Finally, also the complexity of the NNs has to be reduced to be practically applicable. For this purpose one approach could be to work with complex-valued NNs, for which, however, currently hardly any standard training methods can be applied.

To sum up, machine learning and in particular neural networks are also promising for applications in the very model-based communications discipline and in the specific problem of data estimation, but there are still many issues that have to be solved in this research field.
A. Hyperparameter Settings

The hyperparameter settings of the NN based data estimators are found in all simulation cases via an extensive grid search, which is a state-of-the-art method for finding good hyperparameters\(^1\) of neural networks. The hyperparameters of the DetNet, the Attention Detector and the FCNN Detector for the system settings for which they are simulated are given in Table A.1, A.2 and A.3, respectively. It turns out that the performance of the NN based data estimators also depends on the SNR range in which they are trained. Consequently, besides the typical hyperparameters learning rate, number of layers, number of neurons per layer, etc., the training SNR ranges are given in the following tables, too. The comparison with the SNR ranges in which the NNs are tested reveals that the upper limit of the \(\frac{E_b}{N_0}\) values in the training set is most of the times around 1 dB to 2 dB higher than their upper limit in the test set, which is beneficial for the performance of the NNs at higher values of the SNR range they are tested for.

\(^1\)The hyperparameters can only be claimed to be optimal on the grid of investigated hyperparameter combinations.
<table>
<thead>
<tr>
<th>Simulation setting</th>
<th>Test $E_b/N_0$ range $(E_b/N_0)_{dB, test}$</th>
<th>Learning rate $lr$</th>
<th>Nr. DetNet layers $L$</th>
<th>Nr. neurons / hidden layer $d_h$</th>
<th>Nr. memory neurons $d_w$</th>
<th>Residual factor $\alpha$</th>
<th>Training $E_b/N_0$ range $(E_b/N_0)_{dB, train}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>system I, non-systematic, uncoded, QPSK</td>
<td>[8 dB, 16 dB]</td>
<td>$1.9 \cdot 10^{-3}$</td>
<td>10</td>
<td>80</td>
<td>32</td>
<td>0.1</td>
<td>[9 dB, 18 dB]</td>
</tr>
<tr>
<td>system I, non-systematic, uncoded, QPSK</td>
<td>[0 dB, 8 dB]</td>
<td>$1.9 \cdot 10^{-3}$</td>
<td>10</td>
<td>80</td>
<td>32</td>
<td>0.1</td>
<td>[1 dB, 9 dB]</td>
</tr>
<tr>
<td>system I, non-systematic, coded, QPSK</td>
<td>[0 dB, 8 dB]</td>
<td>$1.9 \cdot 10^{-3}$</td>
<td>10</td>
<td>80</td>
<td>32</td>
<td>0.1</td>
<td>[1 dB, 9 dB]</td>
</tr>
<tr>
<td>system I, systematic, uncoded, QPSK</td>
<td>[10 dB, 18 dB]</td>
<td>$1.9 \cdot 10^{-3}$</td>
<td>10</td>
<td>80</td>
<td>32</td>
<td>0.1</td>
<td>[10 dB, 19.5 dB]</td>
</tr>
<tr>
<td>system I, non-systematic, uncoded, BPSK</td>
<td>[8 dB, 16 dB]</td>
<td>$8 \cdot 10^{-4}$</td>
<td>8</td>
<td>80</td>
<td>16</td>
<td>0.1</td>
<td>[9 dB, 18 dB]</td>
</tr>
<tr>
<td>system II, non-systematic, uncoded, QPSK</td>
<td>[16 dB, 26 dB]</td>
<td>$4.6 \cdot 10^{-4}$</td>
<td>30</td>
<td>250</td>
<td>80</td>
<td>0.9</td>
<td>[18 dB, 27.5 dB]</td>
</tr>
<tr>
<td>system II, non-systematic, uncoded, QPSK</td>
<td>[4 dB, 12 dB]</td>
<td>$6 \cdot 10^{-4}$</td>
<td>30</td>
<td>250</td>
<td>80</td>
<td>0.85</td>
<td>[3 dB, 12 dB]</td>
</tr>
<tr>
<td>system II, non-systematic, coded, QPSK</td>
<td>[0 dB, 12 dB]</td>
<td>$6 \cdot 10^{-4}$</td>
<td>30</td>
<td>250</td>
<td>80</td>
<td>0.85</td>
<td>[1 dB, 10 dB]</td>
</tr>
</tbody>
</table>
### Table A.2.: Hyperparameter settings of the Attention Detector

<table>
<thead>
<tr>
<th>Simulation setting</th>
<th>Test $E_b/N_0$ range $(E_b/N_0)<em>dB</em>{test}$</th>
<th>Learning rate $lr$</th>
<th>Nr. encoder layers $L_{enc}$</th>
<th>Nr. neurons / hidden encoder lay. $d_{h,enc}$</th>
<th>Nr. FCNN layers $L_{fcnn}$</th>
<th>Nr. neurons / hidden FCNN lay. $d_{h,fcnn}$</th>
<th>Dropout $D$</th>
<th>Training $E_b/N_0$ range $(E_b/N_0)<em>dB</em>{train}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>system I, non-systematic, uncoded, QPSK</td>
<td>[8 dB, 16 dB]</td>
<td>1.8 · 10$^{-3}$</td>
<td>8</td>
<td>80</td>
<td>2</td>
<td>150</td>
<td>0.1</td>
<td>[9 dB, 18 dB]</td>
</tr>
<tr>
<td>system I, systematic, uncoded, QPSK</td>
<td>[10 dB, 18 dB]</td>
<td>1.2 · 10$^{-3}$</td>
<td>8</td>
<td>80</td>
<td>2</td>
<td>150</td>
<td>0.1</td>
<td>[10 dB, 19.5 dB]</td>
</tr>
</tbody>
</table>

### Table A.3.: Hyperparameter settings of the FCNN Detector

<table>
<thead>
<tr>
<th>Simulation setting</th>
<th>Test $E_b/N_0$ range $(E_b/N_0)<em>dB</em>{test}$</th>
<th>Learning rate $lr$</th>
<th>Nr. hidden layers $L_{h}$</th>
<th>Nr. neuron / hidden layer $d_{h}$</th>
<th>Training $E_b/N_0$ range $(E_b/N_0)<em>dB</em>{train}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>system I, non-systematic, uncoded, QPSK</td>
<td>[8 dB, 16 dB]</td>
<td>4 · 10$^{-4}$</td>
<td>10</td>
<td>300</td>
<td>[9 dB, 18 dB]</td>
</tr>
<tr>
<td>system I, systematic, uncoded, QPSK</td>
<td>[10 dB, 18 dB]</td>
<td>4 · 10$^{-4}$</td>
<td>10</td>
<td>300</td>
<td>[10 dB, 19.5 dB]</td>
</tr>
</tbody>
</table>
Bibliography


