Indian Institute of Space Science and Technology, Thiruvanthapuram
Faculty of Optical Engineering
Department of Physics

Application of Deep Learning in Optical Fiber Communication

Master of Technology Thesis

Under the supervision of
Dr. C.S.Narayanmurthy, IIST,Trivandrum
&
Dr. Debashri Ghosh, CSIR-CGCRI, Kolkata

SANKHA SUBHRA MUKHERJEE
SC19M084

A thesis submitted to IIST,Thiruvanthapuram in partial fullfilment of the requirements of the degree of Master of Technology

I, Sankha Subhra Mukherjee, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Conventional communication systems consist of several signal processing blocks, each performing an individual task at the transmitter or receiver, e.g. coding, modulation, or equalization. However, there is a lack of optimal, computationally feasible algorithms for nonlinear fiber communications as most techniques are based upon classical communication theory, assuming a linear or perturbed by a small nonlinearity channel. Consequently, the optimal end-to-end system performance cannot be achieved using transceivers with sub-optimum modules. Carefully chosen approximations are required to exploit the data transmission potential of optical fibers.

In this thesis, novel transceiver designs tailored to the nonlinear dispersive fiber channel using the universal function approximator properties of artificial neural networks (ANNs) are proposed and experimentally verified. The fiber-optic system is implemented as an end-to-end ANN to allow transceiver optimization over all channel constraints in a single deep learning process. While the work concentrates on highly nonlinear short-reach intensity modulation/direct detection (IM/DD) fiber links, the developed concepts are general and applicable to different models and systems.

Found in many data center, metro and access networks, the IM/DD links are severely impaired by the dispersion-induced inter-symbol interference and squarelaw photodetection, rendering the communication channel nonlinear with memory. First, a transceiver based on a simple feedforward ANN (FFNN) is investigated and a training method for robustness to link variations is proposed. An improved recurrent ANN-based design is developed next, addressing the FFNN limitations in handling the channel memory. The systems' performance is verified in first-infield experiments, showing substantial increase in transmission distances and data rates compared to classical signal processing schemes. A novel algorithm for endto-end optimization using experimentally-collected data and generative adversarial networks is also developed, tailoring the transceiver to the specific properties of the transmission link. The research is a key milestone towards end-to-end optimized data transmission over nonlinear fiber systems.

Impact Statement

Optical fiber networks form the major part of the current internet infrastructure and carry most of the generated digital data traffic. However, the design of conventional fiber links is based upon classical communication theory, which was developed for linear systems. Although convenient to engineer, systems using such techniques are not able to fully exploit the potential for data transmission provided by the nonlinear dispersive communication channel, imposing limitations on the achievable data rates and transmission distances. Consequently, fundamentally new methods are required to increase the throughput and reach of fiber-optic networks.

This research proposes and, for the first time, demonstrates both numerically and in lab experiments, a fundamentally new outlook on optical fiber communications. Unlike classical approaches it allows end-to-end optimized transmission over the channel. The complete fiber-optic system is implemented as an end-to-end deep artificial neural network (ANN), obtaining transmitter and receiver optimized in a single process over all channel constraints. The approach is general and can be applied to different models and systems. The thesis is focused on the application to highly nonlinear short-reach optical fiber links, which are the preferred technology in data center, metro and access networks. Two different end-to-end deep learning-based designs are proposed – a low complexity system based on a simple feedforward ANN as well as an advanced design using recurrent neural networks for nonlinear processing of data sequences. These configurations are successfully verified in breakthrough experiments. Compared to systems based on state-of-theart digital signal processing, they can increase the reach or enhance the data rate at shorter distances, while operating with a lower computational complexity. In addition , the developed optimization algorithms for robustness to link variations and system self-learning during transmission enable the implementation of efficient, easily reconfigurable, and versatile transceivers that exploit the data carrying potential of the optical fiber to a greater extent. These are key features in keeping operational costs low and combined with an efficient hardware implementation can establish the technology as a prime candidate for deployment in future optical networks.

Acknowledgements

I wish to thank Dr.C.S.Narayanmurthy and Dr. Debashri Ghosh for their continuous support

Contents

1 Introduction 19

1.1 Deep learning in communication systems 19

1.1.1 Solving communication problems with deep learning 19

1.1.2 Applications 21

1.2 Thesis outline 26

1.3 Key contributions 27

2 Artificial neural networks and deep learning 32

2.1 Artificial neural networks 32

2.1.1 Artificial neuron 33

2.1.2 Feedforward ANN 34

2.1.3 Recurrent ANN 36

2.1.4 Generative adversarial network (GAN) 40

2.2 Deep learning 43

2.2.1 Gradient descent optimization 43

2.2.2 Backpropagation algorithm 46

3 Optical fiber communication systems for short reach 50

3.1 Transmitter 51

3.1.1 Digital-to-analogue conversion 51

3.1.2 Electro-optical conversion 52

3.2 Propagation over the optical fiber 53

3.3 Receiver 55

3.3.1 Opto-electrical conversion and amplification 55

3.3.2 Analogue-to-digital conversion 55

3.4 Communication channel model as a computational graph 56

4 Design of optical fiber systems using end-to-end deep learning 59

4.1 The optical fiber system as an end-to-end computational graph 60

4.2 Feedforward neural network-based design 61

4.2.1 Transmitter structure 61
4.2.2 Receiver structure 64
4.3 Optimization strategies 66
4.3.1 Learning at a fixed nominal distance 68
4.3.2 Multi-distance learning 69
4.4 Numerical investigation of the FFNN auto-encoder performance 70
4.4.1 Bit-to-symbol mapping 70
4.4.2 Generation of training and testing data 71
4.4.3 Simulation parameters 73
4.4.4 System performance 73
4.5 Summary 79

5 Recurrent neural network-based optical fiber transceiver 80
5.1 End-to-end system design 81
5.1.1 Transmitter 81
5.1.2 Receiver 85
5.1.3 Optimization procedure 88
5.1.4 Sliding window sequence estimation 90
5.1.5 Bit labeling optimization 93
5.2 Numerical investigation of the system performance 94
5.2.1 Simulation parameters 94
5.2.2 System performance 95
5.3 Comparisons with state-of-the-art digital signal processing 98
5.3.1 PAM and sliding window feedforward neural networkbased receiver 98
5.3.2 SBRNN auto-encoder enhancement via optimization of the sequence estimation and bit labeling functions 103
5.3.3 PAM and maximum likelihood sequence detection 105
5.4 Summary 109

6 Experimental demonstrations, comparisons with state-of-the-art DSP and optimization using measured data 110
6.1 Optical transmission test-bed 111
6.2 System implementation strategies 112
6.3 Collection of representative experimental data for optimization 114
6.4 Experimental demonstration of the FFNN-based auto-encoder 117
6.4.1 Experimental performance results 118
6.4.2 Performance comparison with conventional systems 121
6.4.3 Distance-agnostic transceiver 123
6.5 Experimental demonstration of the SBRNN auto-encoder 125

6.5.1 Experimental performance results 127
6.5.2 Reference PAM systems with state-of-the-art receiver DSP .128
6.5.3 Experimental performance comparison between the SBRNN auto-encoder and the reference PAM systems 131
6.6 End-to-end system optimization using a generative model 132
6.6.1 Generative adversarial network design 133
6.6.2 End-to-end optimization algorithm 135
6.6.3 Performance results 138
6.7 Summary 139

7 Conclusions and future work 141
7.1 Conclusions 141
7.2 Future work 144
7.2.1 Advanced optimization methods for distance-agnostic transceiver enhancement 144
7.2.2 Extending the auto-encoder framework 145
7.2.3 Auto-encoders for long-haul coherent optical fiber communications 146

Appendices 148

A FFNN auto-encoder transmitted signal characteristics 148

B Number of nodes and floating point operations per decoded bit in the SBRNN auto-encoder 152
B.1 Counting the number of nodes 152
B.2 Counting the floating point operations 153

C Data collection in the experiments 155
C.1 SBRNN auto-encoder 155
C.2 Reference PAM systems 156
C.2.1 Optimization of the SFFNN receiver 156
C.2.2 Optimization of the Volterra receiver 157
C.2.3 Optimization of the SBRNN receiver 157

D Acronyms 158

Bibliography 161

List of Figures

1.1 General conditions under which deep learning could be considered as a suitable solution to a communication engineering problem . . .20

1.2 Basic schematic of a conventional communication system design, showing some of the main signal processing modules at the transmitter and the receiver.....22

1.3 Communication system implemented as an end-to-end deep artifi cial neural network and optimized in a single process.....23

1.4 Diagram highlighting the scope and organization of the thesis. 26

2.1 Basic schematic of an artificial neuron which combines its inputs x_i into an output y.....33

2.2 Left: sigmoid and hyperbolic tangent (tanh) functions. Right: rectified linear unit (ReLU) function.....34

2.3 Schematic representation of a feedforward artificial neural network (FFNN) which transforms an input vector $\mathbf{x}^0$ to an output vector $\mathbf{x}^K$ .35

2.4 Schematic representation of a recurrent artificial neural network (RNN). 37

2.5 Schematic of an RNN “unfolded” in time. 37

2.6 Gated recurrent unit (GRU) variant of the long short-term memory (LSTM) cell in recurrent neural networks. 38

2.7 Schematic of a bidirectional recurrent neural network (BRNN).....39

2.8 Schematic representation of a generative adversarial network (GAN). The generator aims at transforming the random input vector $\mathbf{z}$ into a vector $\hat{\mathbf{y}}_k$, which is statistically identical to $\mathbf{y}_k$ – a sample vector from the distribution of a real data set. The discriminator
aims at correctly classifying real and fake samples.....41

2.9 Illustration of the gradient descent method for minimisation of the function $f(x) = x^2$. For $f'(x) < 0$ or $f'(x) > 0$, the input x is increased or decreased, respectively, by a small amount $\eta > 0$. The gradient descent algorithm stops when $f'(x) = 0$, where the function $f(x)$ is at a minimum. 44

2.10 Computational graph representation of the process of computing the loss in an artificial neural network. For illustrative purposes, the feedforward network described in Sec. 2.1.2 is used as an ANN reference. It transforms its input $\mathbf{x}^0$ to an output $\mathbf{x}^K$ by performing the operations $\mathbf{x}^k = \alpha_k(\mathbf{W}_k\mathbf{x}^{k-1} + \mathbf{b}_k)$, $k = 1, .., K$. This is followed by the computation of a scalar loss L between $\mathbf{x}^K$ and the training labels $\tilde{\mathbf{x}}^K$. Larger grey nodes are used to highlight the trainable parameters in the graph. 47

3.1 Schematic of an IM/DD communication system, indicating the signal transformations across the transmission link. Tx transmitter, DSP digital signal processing, D/A digital to-analogue conversion, E/O electro-optical conversion, O/E opto-electrical conversion, Amp. amplification, A/D analogue-to-digital conversion, Rx receiver 51

3.2 Schematic showing the communication channel model, which is considered as a segment of the end-to-end computational graph, representing the optical fiber communication system. 57

4.1 Schematic of the optical communication system implemented as an end-to-end computational graph, which represents the process of computing the loss between the transmitter input and the receiver output. 60

4.2 Schematic of the FFNN-based transmitter section of the optical fiber auto-encoder. The input messages $(\ldots m_{t-1}, m_t, m_{t+1})$ are represented as one-hot vectors $(\ldots, \mathbf{1}_{m,t-1}, \mathbf{1}_{m,t}, \mathbf{1}_{m,t+1})$, which are processed independently by the FFNN at each time instance to produce the encoded symbols (blocks of samples) $(\ldots \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \ldots)$, transmitted through the communication channel.62

4.3 Schematic of the FFNN-based receiver section of the optical fiber auto-encoder. The received symbols $(\ldots, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$ are processed independently by the FFNN at each time instance to produce the output probability vectors $(\ldots \mathbf{p}_{t-1}, \mathbf{p}_t, \mathbf{p}_{t+1} \ldots)$, which are used to make a decision on the recovered message as well as tocompute
the loss of the auto-encoder in the optimization stage. 64

4.4 Schematic of the IM/DD optical fiber communication system implemented as a deep feedforward artificial neural network. Optimization is performed using the loss between the input messages and the outputs of the receiver, thus enabling end-to-end deep learning
of the complete system. 67

4.5 Schematic of the training procedure for the FFNN-based autoencoder, showing how a mini-batch is formed from different transmitted sequences of length N over fiber lengths L_i and the corresponding losses, computed for the central message in every sequence.69

4.6 BER as a function of transmission distance for a system trained at 40 km. The horizontal dashed line indicates the 6.7% HD-FEC threshold. The BER with an ideal bit mapping, i.e. a symbol error results in a single bit error, is also shown. 74

4.7 BER as a function of transmission distance for systems trained at a fixed nominal distance of $(20 + i \cdot 10)$ km, with $i \in \{0, , 6\}$. The 6.7% HD-FEC threshold is indicated by a horizontal dashed line. Thin dashed lines below the curves give a lower bound on the achievable BER when an ideal bit mapping is assumed. 75

4.8 Bit error rate as a function of transmission distance for systems where the training is performed at normally distributed distances with mean μ = 40 km and standard deviation σ . The horizontal dashed line indicates the 6.7% HD-FEC threshold. 76

4.9 Bit error rate as a function of transmission distance for systems where the training is performed at normally distributed distances around the mean values μ of 20, 40, 60, 80 km and a fixed standard deviation σ = 4 km. The horizontal dashed line indicates the 6.7% HD-FEC threshold. 77

4.10 Bit error rate as a function of transmission distance for systems with different information rates. The training is performed at a fixed nominal distance. 78

5.1 Schematic of the IM/DD optical fiber communication system implemented as a bidirectional deep recurrent neural network. optimization is performed between the stream of input messages and the outputs of the receiver, thus enabling end-to-end optimization via deep learning of the complete system. Inset figures show an example of the transmitted signal spectrum both at the output of the neural network and before the DAC. 82

5.2 Schematic of the BRNN-based transmitter section of the optical fiber auto-encoder. The input messages (. . . m_{t-1}, m_t, m_{t+1}), represented as one-hot vectors (. . . , $\mathbf{1}_{m,t-1}$, $\mathbf{1}_{m,t}$, $\mathbf{1}_{m,t+1}$ . . .), are processed bidirectionally at each time instance by the neural network to produce the sequence of encoded symbols (blocks of samples) (. . . $\mathbf{x}_{t-1}$, $\mathbf{x}_t$, $\mathbf{x}_{t+1}$). Thick solid lines are used to highlight the the connections to symbols that have an impact on the processing of m_t. Note that a *vanilla* BRNN cell structure is adopted for illustrative purposes. 83

5.3 Schematic of the BRNN-based receiver section of the proposed optical fiber auto-encoder. The received symbols (. . . , $\mathbf{y}_{t-1}$, $\mathbf{y}_t$, $\mathbf{y}_{t+1}$) are processed bidirectionally by the BRNN to produce the output probability vectors (. . . $\mathbf{p}_{t-1}$, $\mathbf{p}_t$, $\mathbf{p}_{t+1}$). These are utilized in two ways: to make a decision on the recovered message as well as to compute the loss of the auto-encoder in the optimization stage. 86

5.4 Schematic of the sliding window sequence estimation technique in which the optimized BRNN transceiver is operated. Note that $W =$ 3 is chosen for illustration purposes. 90

5.5 Bit error rate as a function of transmission distance for the 42 Gb/s end-to-end vanilla and LSTM-GRU SBRNN auto-encoders compared to the 42 Gb/s end-to-end FFNN. 95

5.6 BER as a function of transmission distance for the FFNN and vanilla SBRNN auto-encoders for different neural network hyperparameters M, n (FFNN and SBRNN) and sliding window size W (SBRNN). 96

5.7 Bit error rate as a function of receiver estimation window for the 84Gb/s vanilla and LSTM-GRU SBRNN at 30 km. 97

5.8 Cross entropy loss as a function of optimization step for the 42 Gb/s (a) vanilla and (b) LSTM-GRU SBRNN systems at 30 km. 98

5.9 Schematic of the sliding window FFNN receiver for PAM symbols. 99

5.10 BER as a function of transmission distance for the end-to-end vanilla and LSTM-GRU SBRNN systems compared to the end-toend FFNN as well as the PAM2 system with multi-symbol FFNN receiver. The systems operate at 42 Gb/s. 101

5.11 Top: SER difference as a function of distance for the two approaches for final probability vector estimation in the sliding window algorithm ($W = 10$). SER_u is obtained assuming $a^{(q)} = \frac{1}{W}$, $q = 0, \ldots, W-1$ in Eq. (5.26), while for SER_{opt} the coefficients $a^{(q)}$ are optimized. Bottom: $a^{(q)}$ assignments after optimization for the system at 100 km.....103

5.12 Bit error rate as a function of distance for a reference vanilla SBRNN auto-encoder, which utilizes two different bit-to-symbol mapping approaches: assigning the Gray code to the transmitted message m, proposed in Sec. 4.4.1, and performing the combinatorial algorithm-based optimization, proposed in Sec. 5.1.5. As an indicator, the ideal bit-to-symbol mapping when a single symbol error gives rise to a single bit error is also displayed. 104

5.13 Schematic diagram of the system used to evaluate the performance of the MLSD receiver in IM/DD optical links.....105

5.14 Bit error rate as a function of transmission distance for the 42 Gb/s SBRNN auto-encoder and M-PAM & Rx MLSD systems ($M \in \{2,4\}$). In the case of MLSD $\eta = \mu \log_2(M)$, where μ represents the number of preand post-cursor PAM symbols defining one of M^{μ} channel states. In the case of SBRNN $\eta = W \log_2(M)$ is the number of bits inside the processing window 107

6.1 Diagram showing the three experiments, which were conducted for the demonstration of the end-to-end deep learning concept in optical fiber communication systems.....110

6.2 Schematic of the experimental optical IM/DD transmission test-bed used for investigation of the digital signal processing schemes in this chapter. LPF: low-pass filter, DAC: digital-to-analogue converter, MZM: Mach-Zehnder modulator, TDM: tunable dispersion module, PD: photodiode, TIA: trans-impedance amplifier, ADC: analogue-to-digital converter 111

6.3 Schematic of the experimental optical IM/DD transmission testbed, showing the methods for system optimization using i) numeri-cal simulation of the link; ii) & iii) experimental traces.....113

6.4 Basic schematic showing the data generation, collection and error counting. Each of the experiments involved multiple DAC loads with different random sequences. 114

6.5 Validation symbol error rate as a function of optimization step for different lengths l_v of the validation pattern. The receiver ANN of the system is optimized using traces from the transmission of training sequences with a short pattern $l_{tr} = 6 \cdot 10^2$ 115

6.6 Validation symbol error rate as a function of optimization step for different lengths l_v of the validation pattern. The receiver ANN of the system is re-trained using data from the transmission of training sequences with a long repetition pattern $l_{tr} = 10^4$ 116

6.7 Validation symbol error rate as a function of optimization step for continuously generated random validation messages and different lengths l_{tr} of the message pattern used for receiver optimization. 117

6.8 Experimental BER performance at 20, 40, 60, 80 km of the FFNNbased auto-encoder trained on an explicit transmission model and applied "as is" to the optical test-bed. 119

6.9 Comparison of the experimental BER performance for systems trained at (20, 4) km and (40, 4) km (i) without re-training of the receiver FFNN, (ii) re-training the receiver FFNN by fine-tuning, (iii) training the receiver FFNN by randomization. 120

6.10 Schematic of a conventional feedforward equaliser, used in the reference PAM transmission for experimental performance comparison. 121

6.11 Experimental BER performance for systems trained at (20, 4) km and (40, 4) km. The systems are compared to PAM2 and PAM4 systems with receivers based on conventional feedforward equalisation. 122

6.12 Experimental BER performance for systems trained at (60, 4) km and (80, 4) km. The systems are compared to PAM2 and PAM4 systems with receivers based on conventional feedforward equalisation. 123

6.13 Experimental BER performance as a function of transmission dis tance between 10 and 30 km for a (20, 4) km-trained system, whose receiver ANN parameters are tuned using measured data and the multi-distance learning method proposed in Sec. 4.3.2. 124

6.14 Experimental BER performance as a function of transmission distance between 30 and 50 km for a 42 Gb/s system trained on (40, 4) km, whose receiver ANN parameters are tuned using measured data and the multi-distance learning method proposed in Sec. 4.3.2. The BER of PAM2 and PAM4 schemes with an FFE receiver is shown as a performance indicator. Note that the parameters of the FFE receiver are optimized separately for each distance.125

6.15 a) Experimental BER performance at 20, 40, 60, 80 km for the 42 Gb/s SBRNN-based auto-encoder trained on a channel model and applied "as is" to the optical fiber transmission test-bed. The BER of the end-to-end FFNN system is also shown as a performance indicator. b) BERs at 50, 60 and 70 km as a function of the sliding window size W for the sequence estimation algorithm at the SBRNN receiver 127

6.16 Schematic of the reference sliding window FFNN receiver for PAM symbols, utilized in the experiment. 129

6.17 Schematic of the second-order nonlinear Volterra equaliser for PAM symbols, used as a reference system in the experiment. 130

6.18 BER as a function of transmission distance for 42 Gb/s and 84 Gb/s systems employing deep learning-based and classical DSP schemes optimized using experimental data.....131

6.19 Schematic of the utilized conditional GAN for approximating the function of the IM/DD transmission link. 133

6.20 Schematic of the experimental setup for end-to-end system optimization, showing the forward propagation through the IM/DD link and the optimization flow through the generative model, used in lieu of the actual link.....136

6.21 Flow chart of the proposed iterative algorithm for end-to-end deep learning using measured data. The algorithm includes transmission and optimization regimes. 136

6.22 Experimental bit error rate as a function of optimization iteration. Inset figures show: a) error probabilities at k = 0; b) 2D t-SNE representation of the waveforms output of the transmitter ANN at k = 10; c) error probabilities at k = 10. 138

7.1 Schematic representation of an auto-encoder system which parametrises and includes optimization of the distribution of transmitted symbols (p_{ϑ_m}) together with the transmitter (ϑ_{Tx}) and receiver (ϑ_{Rx}) ANN parameters. 145

7.2 Schematic diagram of the split step Fourier method used to simulate a small propagation step Δ along the optical fiber.....147

A.1 Top: Output of the transmitter ANN, trained at (40,4) km, after filtering with 32 GHz brick-wall LPF for the representative random sequence of 10 symbols $(m_t)_{t=1}^{10} = (2,36,64,40,21,53,42,41,34,13)$ transmitted at 7 GSym/s, i.e. $T \approx 143$ ps. Bottom: Un-filtered ANN output samples, 48 per symbol, for the sub-sequence $(m_t)_{t=6}^{7} = (53, 42)$. 149

A.2 All 64 possible outputs ($m = 1$ to $m = 64$, upper left to bottom right) of the transmitter FFNN before low-pass filtering.....150

A.3 t-SNE representation of the multi-dimensional waveforms output of the transmitter FFNN on the two-dimensional plane. The points are labeled with their respective message number m.....151

A.4 Spectrum of the 32 GHz brick-wall low-pass filtered waveform at the output of the transmitter FFNN, trained at (40,4) km. The figure was generated using 7 MHz spectral bins.....151

List of Tables

4.1 FFNN definition at the transmitter 62
4.2 FFNN definition at the receiver 65
4.3 Simulations parameters assumed for the FFNN-based auto-encoder .72

5.1 Vanilla and LSTM-GRU cell definitions at the transmitter (single direction) 83
5.2 Vanilla and LSTM-GRU cell definitions at the receiver (single direction) 87
5.3 Simulations parameters assumed for the examined systems 94
5.4 Summary of hyper-parameters for the compared ANN-based systems 102

6.1 Definitions of the transmitter and receiver FFNN used for the experimental verification of the end-to-end system optimization method using a generative model 133
6.2 Definitions of the generator and discriminator ANNs in the conditional GAN 134

Chapter 1

Introduction

1.1 Deep learning in communication systems

Through decades of innovation, machine learning has been establishing its presence as a data processing, optimization and analysis tool in various disciplines, ranging from social to natural and applied sciences [1–3]. In recent years, the exponential increase and availability of data and computational resources allowed this process to pick up an unprecedented speed. One particular class of machine learning methods
— deep learning using artificial neural networks— has become the dominant approach for utilizing the big data in many areas due to its computational capacity and scalability potential [4, 5]. Powerful deep learning techniques are revolutionising many aspects of the modern world, leading a tremendous technology leap in fields such as image and speech analysis [6, 7], and natural language processing [8]. The application to communications engineering follows a similar trajectory and after early work in the late 1980s and the 1990s [9], today there is a firm renewal of interest in ubiquitously employing machine learning, and deep learning in particular, on various layers of modern communication networks [10,11].

1.1.1 Solving communication problems with deep learning

Artificial neural networks (ANNs) are formed by interconnected processing elements called *neurons*, organized into multiple layers with nonlinear relation between them [12]. Using a collection of powerful optimization techniques, such computational systems can be optimized to approximate complex nonlinear functions — a process which is often referred to as *deep learning*. Deep learning is a *data-driven* optimization approach, which relies on the availability of a large set of examples, representative of the function's behaviour. It combines well-known algorithms such as backpropagation [13] and gradient descent [14, 15] with modern adaptive learning techniques, specifically developed for multi-layer (deep) ANNs [16–18].

Using artificial neural networks and deep learning for the design and optimi-

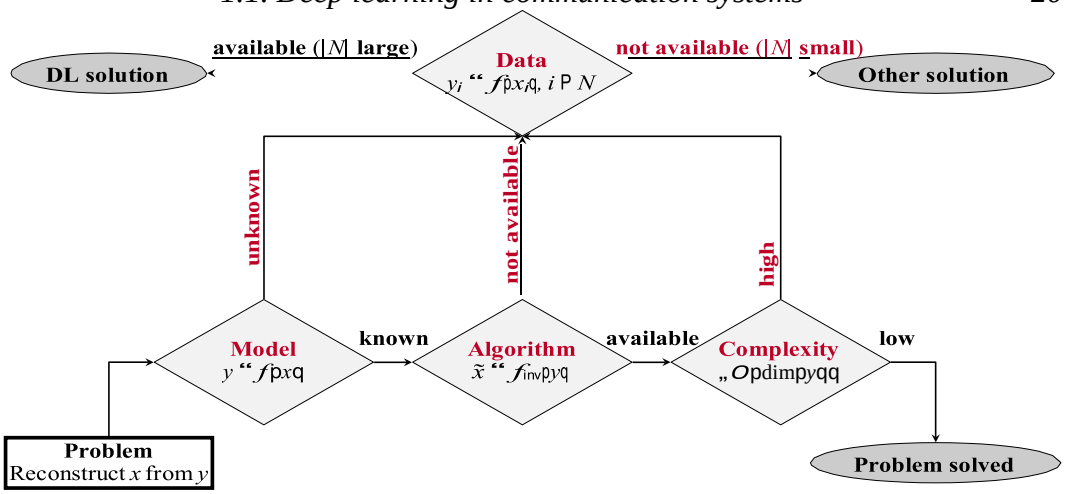

Figure 1.1: General conditions under which deep learning could be considered as a suitable solution to a communication engineering problem (adapted from [19]).

sation of an algorithmic solution to a communication problem can be viewed as a methodology which is an alternative to conventional domain knowledge-based approaches. Inspired by [19–21], Fig. 1.1 together with the discussion in this section provides some general instructions on how to identify communication tasks for which the deep learning perspective is particularly suitable.

First, in order for the traditional engineering flow to be applied, an *accurate mathematical model* which captures the underlying physical properties of the problem needs to be present. If the model is unknown, conventional model-based techniques cannot be designed and it is necessary to resort to alternative approaches. As already mentioned, a substantial condition that needs to be satisfied before considering a deep learning solution is the presence of sufficient *data* examples for the training of the algorithm. Importantly, in the case of a lack of a model, the data needs to be *collected* such that its distribution is representative for the encountered problem.

Next, in cases where the field of interest is characterised by an established mathematical model, the availability of existing *optimal algorithms*, engineered based on the model, needs to be considered. If such algorithms are not known or not available due to complexity reasons, the use of deep learning might be utilized, given that sufficient amount of data is available. Note that typically in this case large amounts of data can be *generated* using the physics-based model.

Finally, even in the presence of well-established models and optimal algorithmic solutions developed for them, a deep learning approach could still be beneficial in scenarios where its implementation would require a lower *complexity* without sacrificing the performance compared to the traditional techniques.

Although communication engineering is a field with well-established channel models [22–24] as well as algorithms for reliable communication over them [25], they apply only to a sub-class of all communication problems. Moreover, often the available domain knowledge can be integrated into the learning algorithm, while the collection of a sufficiently large amount of data for training in certain situations can be easier than applying traditional approaches. As a consequence, the advantages of applying deep learning in place of a conventional technique should be evaluated on an individual basis.

1.1.2 Applications

The unparalleled growth of interconnected devices nowadays leads to an enormous amount of generated Internet traffic which increases at an exponential rate [26]. This results in rapidly increasing complexity and dynamics in all modern networking systems, the core part of which is constituted by the optical fiber infrastructure, and presents great challenges for their configuration, management and control [27, 28]. Stringent requirements regarding the quality of service, latency and flexibility to frequent change in conditions are fastly becoming infeasible to realize using conventional approaches. In the same time, the ubiquitous collection of various networking data, for example from monitoring reports, has empowered wide interest in developing scalable, energy-efficient and optimized solutions using deep learning, artificial neural networks and machine learning in general [29–33]. For example, the techniques have been used for the modeling, prediction and classification of the data traffic flow [34–40], enabling a promising direction for substantially reducing the operational cost of the network [34].

From physical layer's perspective, the application of deep learning, as a collection of powerful optimization tools, is considered particularly suitable for digital signal processing (DSP). Advanced DSP algorithms enable the compensation for wide variety of transmission impairments and have become an effective ubiquitously employed solution for increasing the data rate and transmission reach of state-of-the-art communication systems [41, 42]. The conventional communication transceiver design consists of several DSP blocks, each employed to perform an individual task. Figure 1.2 shows a basic schematic representation of such a system. At the transmitter, signal processing functions such as coding, modulation, pre-compensation and pulse-shaping are applied to the incoming data, shaping the signal before it is launched into the transmission link. At the receiver, the signal, distorted by the communication channel, is typically equalised before functions such as demodulation and decoding are performed to recover the transmitted data. Such a modular transceiver implementation is convenient from an engineering perspective

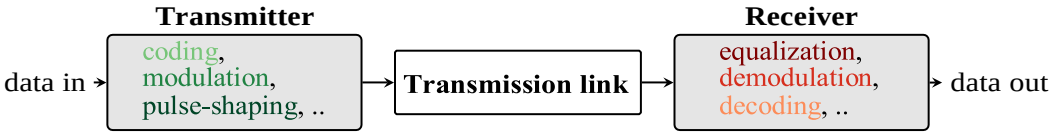

Figure 1.2: Basic schematic of a conventional communication system design, showing some of the main signal processing modules at the transmitter and the receiver.

as it allows separate analysis, optimization and control of the DSP blocks.

Due to their powerful function approximation capabilities, the use of artificial neural networks combined with deep learning in the DSP chain of modern communication transceivers has attracted great interest. Often the techniques are considered for the optimization or the computational complexity reduction of a specific DSP function. This has been achieved by incorporating varying degrees of the available domain knowledge for completing the DSP task. For example, artificial neural networks and deep learning enabled a low-complexity near-optimal solution for the demodulation of received symbols in additive white Gaussian noise (AWGN) channels [43]. Applications in the decoding process of forward error correction (FEC) codes have also been widely investigated [44–47]. In [44] deep learning was applied for the optimization of the existing belief propagation decoding algorithm for linear codes, leading to increased decoding capabilities. This prompted the development of methods for complexity reduction of the scheme [48]. In [47] it was shown that advanced recurrent neural network architectures can be used for the optimal (up to a specific memory) decoding of convolutional codes.

During propagation through the communication link, the transmitted signals undergo various distortions which are often mitigated using digital equalisation schemes at the receiver. Typically, the equalisation algorithms are designed us ing the available mathematical model, which describes the propagation through the transmission medium. For example, an efficient method for the compensation of deterministic distortions in long-haul coherent optical fiber systems is to solve the nonlinear Schrödinger equation, which governs signal propagation over the optical fiber, using negated medium parameters, a technique known as digital backpropagation (DBP) [49]. The algorithm involves multiple iterations of linear and nonlinear processing steps, whose number, and thus the computational complexity of the DBP, increases with the input power, transmission bandwidth and distance. Moreover, exact knowledge of the link parameters is required for optimal compensation. A method, known as learned DBP (LDBP), which consists in "unfolding" the iterative algorithm to interpret its steps as ANN layers and apply deep learning, allowed to reduce the complexity and tune the compensation to the specific link parameters [50–52]. Signal equalization using ANNs has been considered for satellite

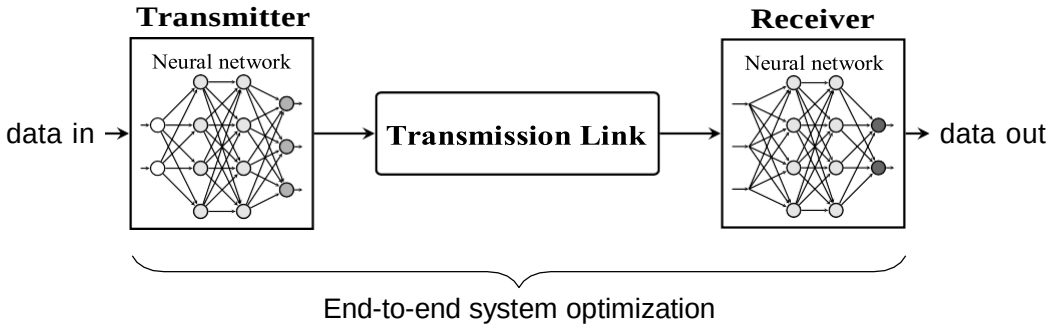

Figure 1.3: Communication system implemented as an end-to-end deep artificial neural network and optimized in a single process.

communications in the early 1990s [53, 54], while in recent years the method attracted a renewed interest in both long-haul [55, 56] and short-reach [57, 58] optical fiber communications.

All aforementioned applications of ANNs and deep learning are examples where the techniques were applied for the optimization of a specific DSP function in the transmitter or receiver, which themselves consist of signal processing modules with separated tasks. However, the approach of designing and optimizing the communication system independently on a module-by-module basis can be sub-optimal in terms of the end-to-end performance. A different approach to deep learning-based DSP, which utilizes the function approximation capabilities of artificial neural networks and deep learning to a greater extent, is to interpret the complete communication chain from the transmitter input to the receiver output as a single deep ANN and optimize its parameters in an end-to-end process. The idea of such fully learnable communication transceivers (known as *auto-encoders*) was introduced for the first time for wireless communications in [59–61]. The autoencoders avoid the modular design of conventional communication systems and have the potential to achieve an end-to-end optimized performance for a specific metric. Figure 1.3 shows a simple schematic of such an end-to-end deep learning based system.

The application of auto-encoders is particularly viable in communication scenarios where the optimum pair of transmitter and receiver or optimum DSP modules are not known or computationally prohibitive to implement. The approach was quickly utilized in optical fiber communications aiming at exploiting to a greater extent the potential for data transmission [62–65]. In particular, the achievable data rates and transmission distance in short reach optical fiber links based on the intensity modulation/direct detection (IM/DD) technology are severely limited by the presence of chromatic dispersion and nonlinear (square-law) signal detection [23].

The joint effects of dispersion, which causes intersymbol interference (ISI), and square-law detection render the optical IM/DD a nonlinear communication channel with memory for which optimal, computationally feasible algorithms are notavailable. State-of-the-art IM/DD systems employ techniques such as nonlinear Volterra equalisation [66] or maximum likelihood sequence detection [67–69] to address the channel distortions. However, as the channel memory increases, the implementation of such algorithms quickly becomes infeasible because of the associated computational complexity. Moreover, their performance can be significantly degraded when the complexity is constrained. On the other hand, deep learning, which can approximate complex nonlinear functions, can be used to provide carefully chosen transmitter and receiver for the optical IM/DD via end-to-end optimization. The IM/DD systems play an important role as the enabling technology in many data center, metro and access optical fiber networks [70] and the application of end-toend deep learning for enhancing the system performance is a promising direction for investigation. Moreover, it should also be noted that the relatively stable link conditions, typical for guided transmission media such as the optical fiber, further facilitate the prospect of end-to-end system learning.

In the research described in this thesis, the concept of end-to-end deep learning was applied for the first time in optical fiber communications. The work was focused on IM/DD systems, but it should be emphasized that the developed designs and methods are not restricted to this scheme and can be extended to different systems. In particular, two different transceiver designs were proposed — a simple structure using a feedforward ANN (FFNN) as well as an advanced design based on a recurrent ANN (RNN), tailored to the dispersive properties of the nonlinear channel. Their performance was extensively investigated in numerical simulations and successfully verified on an experimental optical IM/DD test-bed. The complexity and performance of the auto-encoder systems was compared to state-of-the-art IM/DD transmission schemes based on pulse amplitude modulation with receivers using classical DSP approaches such as nonlinear Volterra equalisation and maximum likelihood sequence detection. The research included development of novel methods for the optimization of the deep learning-based optical transceivers. Firstly, an approach for the generalization of the ANN parameters, which allowed transmission over varied distances without reconfiguration of the transceivers, was proposed and experimentally demonstrated. Moreover, the concept of end-to-end optical fiber system optimization using a generative adversarial network (GAN) and experimentally collected data was introduced and verified on the transmission test-bed.

The work in this thesis forms an important step towards establishing end-to end deep learning as a viable digital signal processing solution for future optical fiber communication networks.

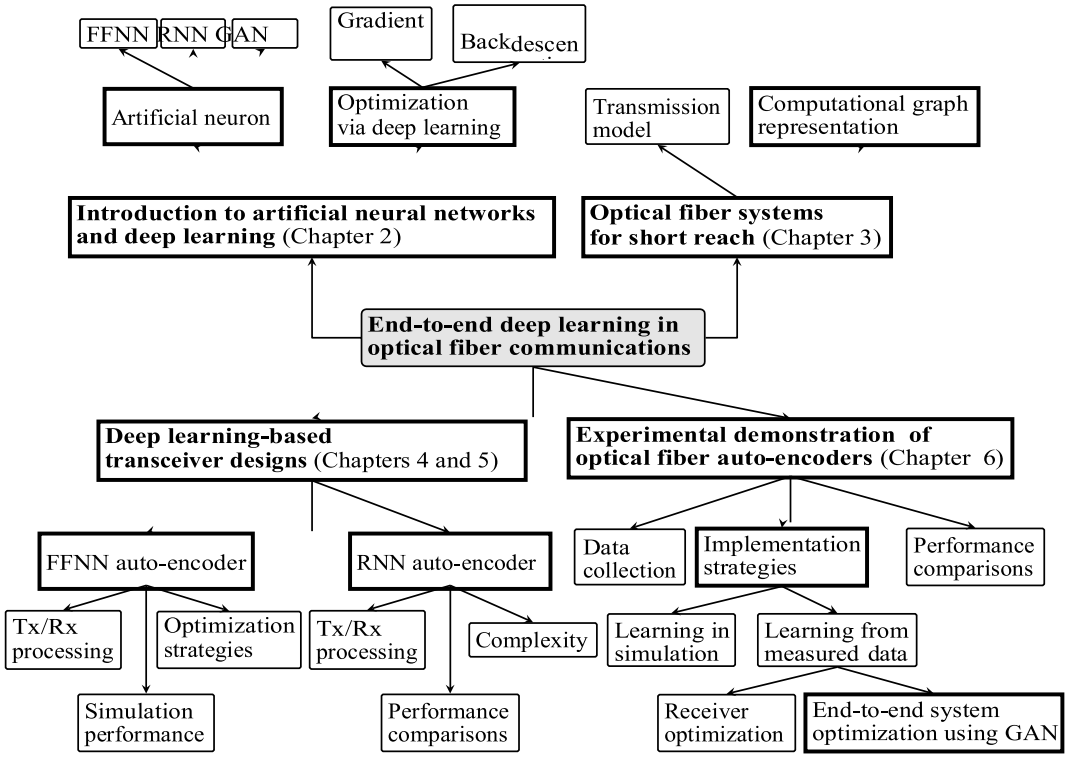

Figure 1.4: Diagram highlighting the scope and organization of the thesis.

1.2 Thesis outline

Figure 1.4 shows a diagram which summarises the organization and highlights the scope of this thesis. The remainder of the thesis is organized as follows:

Chapter 2 gives a detailed description of the artificial neural networks and deep learning techniques which are used throughout this thesis. It starts with an introduction to artificial neural networks (ANNs) of which the simplest feedforward ANN (FFNN) is presented first, followed by the recurrent ANN (RNN) and the generative adversarial network (GAN). The deep learning framework for training of the neural network parameters is summarised next. This includes introduction to the gradient descent (GD) optimization method, as well as the backpropagation algorithm for computing gradients. Then, the specific variant of a learning algorithm that is used in this work is introduced – a stochastic gradient descent optimization method, known as the Adam optimizer.

Chapter 3 presents the physical properties of the intensity modulation/direct detection optical fiber transmission system, used as a reference in the development of the end-to-end deep learning concept. It includes a detailed description of the mathematical modeling of all the system components.

In **Chapter 4** an optical fiber system design based on an end-to-end deep feedforward neural network is proposed and numerically investigated. The principles

of block-based transmission enabled by the FFNN are explained and the processing at the transmitter and receiver sections of the systems are described in detail. Simulation results for the system performance are also presented. Moreover, a novel method for training of the neural network parameters, developed to obtain transceivers robust to distance variations, is introduced in this chapter.

In **Chapter 5** an advanced system design based on a sequence processing using a bidirectional RNN (BRNN) is developed. The neural network functions within the transmitter and receiver are described together with the efficient sliding window sequence estimation algorithm in which the optimized system is operated. This chapter also includes the procedures developed for coefficient optimization for the estimation algorithm, as well as techniques for optimizing the bit-to-symbol mapping function. The performance of the end-to-end BRNN-based auto-encoder is numerically investigated and compared to conventional pulse amplitude modulation (PAM) systems with state-of-the-art nonlinear ANN equalizers as well as maximum likelihood sequence detection (MLSD). The complexity of the BRNN system is also analysed and compared to these benchmarks.

Chapter 6 focuses on the experimental demonstration of the FFNN and BRNN systems as well as the verification of the proposed optimization method for robustness to distance variations. It begins with a detailed description of the experimental optical IM/DD transmission test-bed. Then, the data collection procedures are explained. The system performance is examined at different transmission distances. A detailed experimental comparison with PAM transmission and the receivers based on classical digital signal processing is also carried out. Importantly, the chapter includes a description and experimental demonstration of the proposed concept for end-to-end system optimization using a GAN-based model of the actual transmission link obtained using collected data from measurements.

Chapter 7 summarizes the work presented in Chapters 4, 5 and 6, and outlines important future directions for development of the research towards flexible transceivers and applications in long-haul optical systems.

1.3 Key contributions

i) In Chapter 4 a novel framework for coding and detection in optical fiber communications using end-to-end deep learning is introduced and numerically investigated for the first time. The work was published in [2].

ii) A novel training method for the generalization of the neural network parameters is presented in Chapter 4 and its experimental verification is described in Chapter 6. It allows the optimization of transceivers robust to variations in the

transmission distance. This concept was published in [2], while its experimental demonstration was published in [12].

iii) In Chapter 5 an advanced transceiver design performing nonlinear sequence processing using recurrent neural networks is developed and its performance is numerically evaluated, outperforming state-of-the-art DSP algorithms. The work led to publications [1] and [11].

iv) In Chapter 5 an offline method for optimizing the weight assignments in the sliding window estimation algorithm in which the recurrent neural network transceiver is operated is developed. This method, which allowed additional performance improvement of the system, was published in [11].

v) In Chapter 5 a bit-to-symbol optimization method using a combinatorial algorithm is proposed. It was published in [11].

vi) In Chapter 5 a comparative study of performance and computational complexity with a benchmark maximum likelihood sequence detection system for the recurrent neural network-based transceiver is described. This investigation was published in [11].

vii) In Chapter 6 the first experimental demonstration of an optical fiber system implemented as an end-to-end deep neural network is presented. The setup and the results from the experiment were reported in [2].

viii) The first-in-field experiment of an end-to-end recurrent neural network-based optical system is described in Chapter 6. The system setup and performance were reported in [8] and [9].

ix) Comprehensive numerical (Chapters 4 and 5) and experimental (Chapter 6) comparisons with conventional IM/DD systems based on linear and nonlinear receiver digital signal processing were carried out. These results were published in [2],[8],[11], and [12].

x) The concept and experimental demonstration of an end-to-end optical system optimization using data collected from measurements are presented in Chapter 6. The proposed algorithm, which employs a generative adversarial network for modeling of the real transmission link, was published in [10].

Chapter 2

Artificial neural networks and deep learning

The combination of artificial neural networks (ANNs), known as universal function approximators, and deep learning is a powerful tool for the modeling of any complex function [12]. This chapter starts with an introduction to the data processing and optimization objective in the different types of artificial neural networks, used to perform the research described in the thesis. This is followed by a detailed description of the deep learning techniques enabling the optimization of the ANN parameters.

2.1 Artificial neural networks

The original concept of artificial neural networks, introduced in the 1940s, was inspired by the prospect of mimicking the biological neural networks in the human brain [71]. Nevertheless, their application as computational entities has been far reaching. An ANN consist of simple interconnected processing elements, called *artificial neurons*, which are typically organized into multiple layers. It has been mathematically proven in the late 1980s that such structures can be optimized to approximate an arbitrary mapping from one vector space to another [72–74]. The optimization of an artificial neural network can be performed using a set of labeled data examples for training, i.e. in the so-called *supervised* manner. For this purpose, the pairing of an input and a desired output is defined. The training set is formed by such pairings, whose number (set size) is chosen such that the data is representative of the input-to-output mapping function to be approximated by the ANN. In this section the data processing in the artificial neuron, the building block of every ANN is discussed. Then, three different types of networks and their optimization objectives are introduced — feedforward and recurrent ANNs as well as generative adversarial networks.

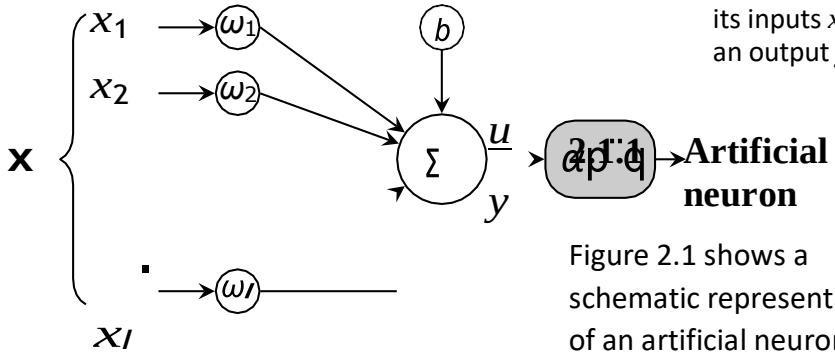

Figure 2.1: Basic schematic of an artificial neuron which combines its inputs x_i into an output y.

Figure 2.1 shows a schematic representation of an artificial neuron. The function of the neuron is to map its

inputs x_i to an output y through an operation given :-

$$y = \alpha\left(\sum_{i=1}^{l} \omega_i x_i + b\right) \tag{2.1}$$

where $x_i \in \mathrm{R}$, $i = 1 \ldots l$ are the inputs to the neuron, which can be denoted by the vector $\mathbf{x} \in \mathrm{R}^l$, $y \in \mathrm{R}$ is the neuron's scalar output, $\omega_i \in \mathrm{R}$, $i = 1 \ldots l$ and $b \in \mathrm{R}$ are the neuron's weights (one for each input) and bias parameters, while α is the so-called *activation function*. The neuron's activation function plays an important role in the structure of every ANN as it is typically chosen to introduce a nonlinear relation between the layers, enabling the approximation of nonlinear functions.

A commonly employed activation function in state-of-the-art ANNs is the rectified linear unit (ReLU) [75], which keeps the positive values and equates the negative to zero, i.e. $y = \alpha_{\mathrm{ReLU}}(u)$ with

$$y = \max(0, u). \tag{2.2}$$

Another function often used as an activation for the artificial neurons is the sigmoid $y = \alpha_{\mathrm{sigmoid}}(u)$ where

$$y = \frac{1}{1 + \exp(-u)}. \tag{2.3}$$

The sigmoid translates its input to an output in the (0; 1) region. A function that can output negative values and is also often used in practice is the hyperbolic tangent, expressed as $y = \alpha_{\tanh}(u)$ with

$$y = \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)}. \tag{2.4}$$

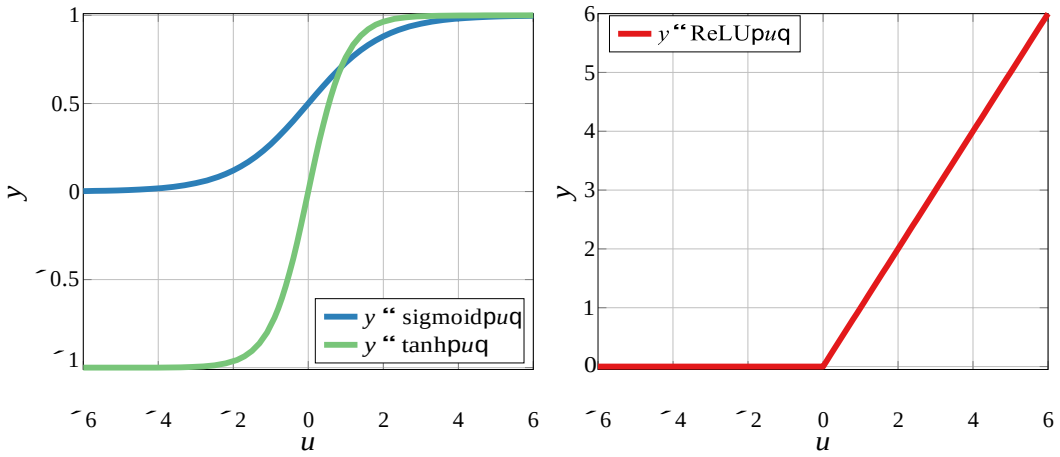

Figure 2.2: Left: sigmoid and hyperbolic tangent (tanh) functions. Right: rectified linear unit (ReLU) function.

The hyperbolic tangent function allows the neuron output to take values in the $(-1; 1)$ range. Figure 2.2 exemplifies the ReLU, sigmoid and hyperbolic tangent functions. The choice of an activation function for the layers of an ANN has an impact on the speed and convergence of the procedure for parameter optimization. A detailed description of the optimization procedure is provided in Section 2.2. Compared to the other two popular activation functions, the ReLU has a constant gradient, which renders training computationally less expensive and also partly avoids the effect of *vanishing gradients*. This effect occurs for activation functions with asymptotic behaviour, such as sigmoid and *tanh*, since the gradient can become small and consequently decelerate the convergence of the learning algorithm. A discussion on the vanishing gradient problem in the training of ANNs can be found in [12, Sec. 8.2]. Another important activation function, the *softmax* function, which normalises the whole layer of neurons in order to provide probability outputs, is introduced in the following section together with the simplest form of organization of artificial neurons – the feedforward ANN.

2.1.2 Feedforward ANN

2.1.2.1 Data processing

An artificial neural network consists of interconnected layers of neurons. Each layer is characterised by an activation function as well as a weight matrix and a bias vector where the parameters of its neurons are organized. Figure 2.3 shows a schematic of a feedforward ANN (FFNN), a network where the data is passed only in one direction – from the input to the output layer (forward). The network is fully connected, i.e. the output of each of the nodes (neurons) in one layer is used as an input to all nodes in the next layer. A K-layer FFNN maps an input vector $\mathbf{x}^0$ to an

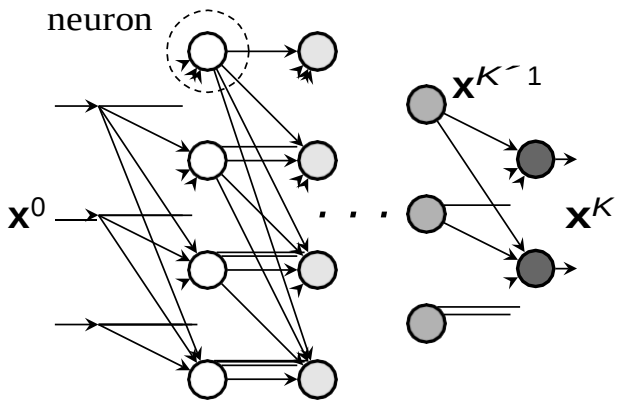

Figure 2.3: Schematic representation of a feedforward artificial neural network (FFNN) which transforms an input vector $\mathbf{x}^0$ to an output vector $\mathbf{x}^K$.

output vector $\mathbf{x}^K = f_{\text{FFNN}}(\mathbf{x}^0)$ through the iteratively applied operations of the form

$$\mathbf{x}^k = \alpha_k(\mathbf{W}_k\mathbf{x}^{k-1} + \mathbf{b}_k), \qquad k = 1,..,K, \tag{2.5}$$

where $\mathbf{x}^{k-1} \in \mathrm{R}^{l_{k-1}}$ is the output of the $(k-1)$-th ANN layer, $\mathbf{x}^k \in \mathrm{R}^{l_k}$ is the output of the k-th layer, $\mathbf{W}_k \in \mathrm{R}^{l_k \times l_{k-1}}$ and $\mathbf{b}_k \in \mathrm{R}^{l_k}$ are respectively the weight matrix and the bias vector of the k-th layer and α_k is its activation function.

As already mentioned, the ANN can be trained to approximate any complex function, which is enabled by the presence of a (typically nonlinear) activation function on each layer. In addition to the most common activation functions described in the previous section, the *softmax* is another important function. It is applied element-wise and is defined as $\mathbf{y} = \text{softmax}(\mathbf{u})$ with

$$y_i = \frac{\exp(u_i)}{\sum_j \exp(u_j)} \tag{2.6}$$

The *softmax* function is typically used on the final layer of an ANN in cases where a probability output is required (e.g. in classification tasks) and performs normalisation of the whole layer such that $\sum_i y_i = 1$.

2.1.2.2 Optimization objective

For the optimization of parameters in the FFNN, pairings between an input vector $\mathbf{x}^0$ and a desired output vector $\tilde{\mathbf{x}}^K$ are defined as elements in the training set. Let the set of all parameters of the FFNN is denoted by

$$\vartheta = \{\varphi_1,\ldots,\varphi_K\}, \tag{2.7}$$

where $\varphi_k = \{\mathbf{W}_k, \mathbf{b}_k\}$ is the set of parameters for the k-th layer. The objective of the training is to minimize, over the training set, the loss (error) $L(\vartheta)$ between the desired output $\tilde{\mathbf{x}}^{K,i}$ and the ANN output $\mathbf{x}^{K,i} = f_{\text{FFNN}}(\mathbf{x}^{0,i})$, with respect to the parameter set ϑ. The loss can be expressed as:-

$$\mathscr{L}(\theta) = \frac{1}{|S|} \sum_{(\mathbf{x}^{0,i}, \tilde{\mathbf{x}}^{K,i}) \in S} \ell\left(\tilde{\mathbf{x}}^{K,i}, f_{\text{FFNN}}\left(\mathbf{x}^{0,i}\right)\right) \tag{2.8}$$

where $A(\mathbf{x}, \mathbf{y})$ denotes the specific per-example loss function that is utilized and $|S|$ denotes the cardinality of the training set, i.e. the number of $(\mathbf{x}^{0,i}, \tilde{\mathbf{x}}^{K,i})$ pairings of inputs and labeled outputs used for training. A ubiquitously employed loss function in classification tasks, which is as well considered in this thesis, is the cross entropy defined as

$$A(\mathbf{x}, \mathbf{y}) = -\sum_{i} x_i \log(y_i). \tag{2.9}$$

To achieve a machine learning estimator, both the input to the neural network $\mathbf{x}^0$ and the desired output $\tilde{\mathbf{x}}^K$ are typically represented as a *one-hot vector* of size M, denoted as $\mathbf{1}_m \in \mathrm{R}^M$, where the m-th element equals 1 and the other elements are

0. Such one-hot encoding is the standard way of representing categorical values in most deep learning algorithms [12]. Accordingly, the final layer in the neural network uses the *softmax* activation such that its output $\mathbf{x}^K$ is a probability vector. Section 2.2 presents the approach for optimization of the parameter set ϑ aimed at minimising the computed loss.

2.1.3 Recurrent ANN

2.1.3.1 Data processing

As described in the previous section, feedforward neural networks process each input vector independently to obtain an output. The structure of such networks does not include mechanisms allowing information to persist when dealing with sequences of data. However, the solutions to many tasks, especially in communication systems, require processing of data sequences. In such cases, the application of an advanced ANN architecture – the recurrent neural network (RNN) [13, 76] is beneficial.

Figure 2.4 illustrates the basic concept of a recurrent neural network. As seen in the figure, the structure of an RNN is characterised by a loop which allows some information from the previous processing steps to be included in the processing of the current data input. This is further exemplified in Fig. 2.5, which presents a schematic of the RNN loop "unfolded" in time. The recurrent structure takes as an input the sequence $(\ldots, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, \ldots)$. At time t, the input vector $\mathbf{x}_t$ is processed

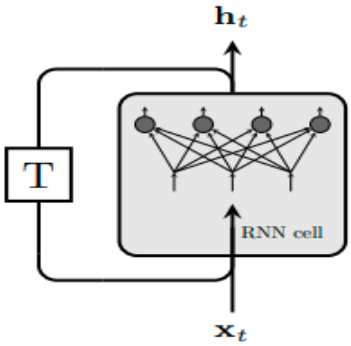

Figure 2.4: Schematic representation of a recurrent artificial neural network (RNN).

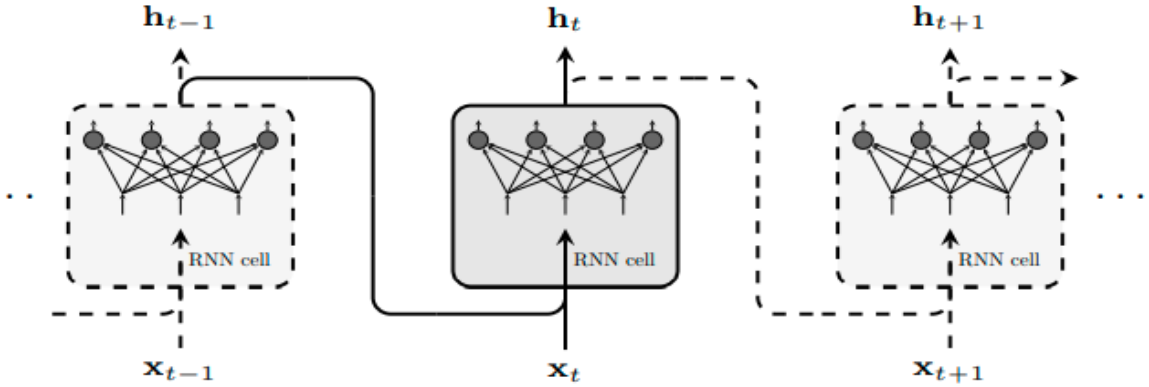

Figure 2.5: Schematic of an RNN "unfolded" in time.

by the recurrent cell together with the previous output $\mathbf{h}_{t-1}$ to produce an updated output $\mathbf{h}_t$. In its simplest (*vanilla*) variant, the processing in the recurrent cell is similar to the single-layer FFNN, acting on the concatenation between the current input and previous output. Thus, the current output of the recurrent cell can be expressed as

$$\mathbf{h}_t = \alpha\left(\mathbf{W}\left(\mathbf{x}_t^T \quad \mathbf{h}_{t-1}^T\right)^T + \mathbf{b}\right) \tag{2.10}$$

where $\mathbf{x}_t \in \mathrm{R}^l$ is the RNN input at time t, $\mathbf{h}_{t-1} \in \mathrm{R}^n$ is the RNN output at time $t-1$, $\mathbf{W} \in \mathrm{R}^{n\times(n+l)}$ and $\mathbf{b} \in \mathrm{R}^n$ are the weight matrix and bias vector, respectively, and ^{T} denotes the matrix transpose. The activation function is denoted by α.

In addition to the *vanilla* RNN cell, which processes the simple concatenation of the current input with the most recent output of the network, more advanced structures have been developed to adequately model long-term dependencies, characteristic for certain tasks [77]. The long short-term memory (LSTM) architecture is one of the most widely used designs for learning long-term dependencies in sequential data, achieved by preserving information in an internal cell state [78]. The

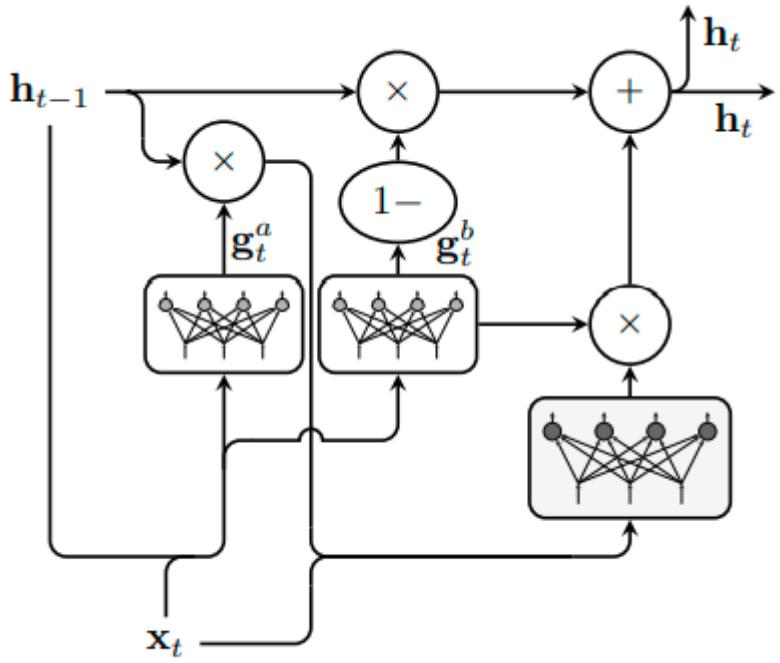

Figure 2.6: Gated recurrent unit (GRU) variant of the long short-term memory (LSTM) cell in recurrent neural networks.

LSTM is an enabling technique in many fields related to language processing [79] and different variants of the cell structure have been examined to enhance the performance, reduce the complexity or improve the convergence speed [80]. A particularly efficient LSTM architecture with state-of-the-art performance is the so-called *gated recurrent unit* (GRU), proposed in [81]. Next, the structure of the GRU cell is introduced in more detail.

Figure 2.6 shows a GRU schematic. Compared to the *vanilla* structure, it consists of two additional single-layer memory gates $\mathbf{g}_t^a$ and $\mathbf{g}_t^b$ employed to control the information flow in the cell. The gates take as an input the concatenation of the current input $\mathbf{x}_t$ with the previous output $\mathbf{h}_{t-1}$ and their function is to decide which elements of the previous output should be preserved for further processing. The current output of the GRU cell is obtained as

$$\mathbf{g}_t^a = \alpha_1\left(\mathbf{W}_1\left(\mathbf{x}_t^T \quad \mathbf{h}_{t-1}^T\right)^T + \mathbf{b}_1\right), \tag{2.11}$$

$$\mathbf{g}_t^b = \alpha_2\left(\mathbf{W}_2\left(\mathbf{x}_t^T \quad \mathbf{h}_{t-1}^T\right)^T + \mathbf{b}_2\right), \tag{2.12}$$

$$\mathbf{h}_t = (1-\mathbf{g}_t^b)\odot\mathbf{h}_{t-1} + \mathbf{g}_t^b \odot \alpha_3\left(\mathbf{W}_3\left(\mathbf{x}_t^T \quad (\mathbf{g}_t^a \odot \mathbf{h}_{t-1})^T\right)^T + \mathbf{b}_3\right), \tag{2.13}$$

where $\odot$ denotes the Hadamard product (element-wise multiplication of vectors) and $\mathbf{W}_i$ and $\mathbf{b}_i$ are the layers' weights and biases, whose sizes are identical to the size of the weight and bias of the *vanilla* cell. The activation functions of the two gates, i.e. α_1 and α_2, are typically chosen to be the sigmoid function, while α_3 can

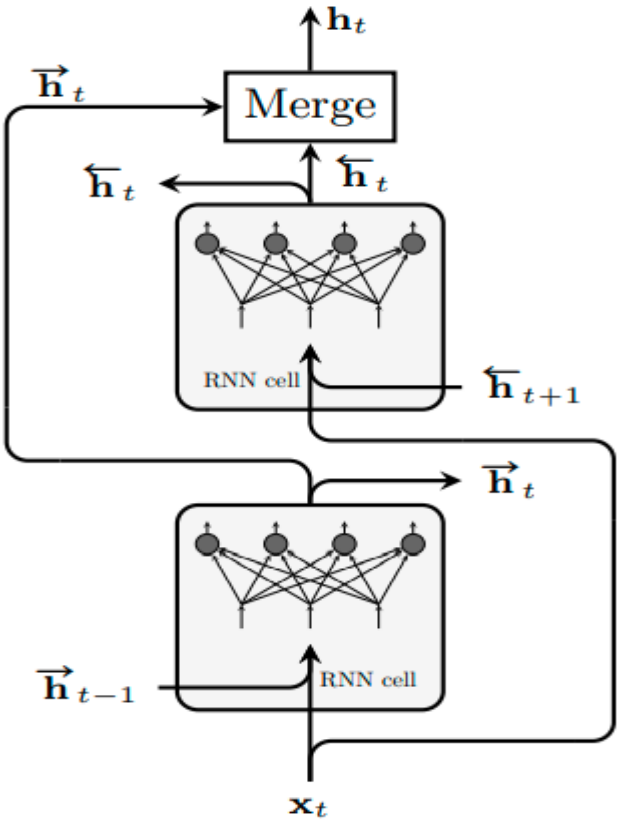

Figure 2.7: Schematic of a bidirectional recurrent neural network (BRNN).

be the ReLU or the hyperbolic tangent functions.

So far, all aforementioned RNN examples consider dependencies in the sequence $(\ldots, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, \ldots)$ which are stemming only from the preceding data, i.e. the transformation of $\mathbf{x}_t$ is performed using information from the previous inputs $\mathbf{x}_{t-k}$, $k = 1 \ldots t-1$. Such *unidirectional* processing is inherently unable to model any influence on the current input to the RNN occurring from the post-cursor data. To enable the modeling of both preceding and succeeding dependencies in a data sequence, the *bidirectional* recurrent neural network (BRNN) architecture was introduced in [82]. Subsequently, the BRNN mechanism has been widely applied for example in the field of speech processing as it allow to efficiently capture important contextual information [83].

Figure 2.7 shows a schematic of the BRNN concept. It combines two unidirectional RNNs which are responsible for the processing of the input sequence in its original (forward) and reversed (backward) order. In the forward direction, the concatenation of an input $\mathbf{x}_t$ at time t with the previous output $\overrightarrow{\mathbf{h}}_{t-1}$ is processed by the recurrent cell to produce an updated output $\overrightarrow{\mathbf{h}}_t$. The procedure is performed on the full data sequence. In order to adequately handle dependencies from the succeeding symbols, the structure is repeated in the backward direction, updating the outputs $\overleftarrow{\mathbf{h}}_t$. Assuming a simple vanilla cell structure in both directions, the process-

ing in the BRNN can be expressed as:-

$$\overrightarrow{\mathbf{h}}_t = \alpha_{\text{fw}}\left(\mathbf{W}_{\text{fw}}\left(\mathbf{x}_t^T \quad \overrightarrow{\mathbf{h}}_{t-1}^T\right)^T + \mathbf{b}_{\text{fw}}\right), \tag{2.14}$$

$$\overleftarrow{\mathbf{h}}_t = \alpha_{\text{bw}}\left(\mathbf{W}_{\text{bw}}\left(\mathbf{x}_t^T \quad \overleftarrow{\mathbf{h}}_{t+1}^T\right)^T + \mathbf{b}_{\text{bw}}\right), \tag{2.15}$$

$$\mathbf{h}_t = \text{merge}\left(\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t\right), \tag{2.16}$$

where $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$ denote the outputs of the RNN cells in the forward and the backward directions, respectively, $\mathbf{W}_{\text{fw}}$ and $\mathbf{W}_{\text{bw}}$ are the corresponding weight matrices in the forward (fw) and backward (bw) directions, while $\mathbf{b}_{\text{fw}}$ and $\mathbf{b}_{\text{bw}}$ are the bias vectors. The activation functions used in the two directions, which are typically identical, are denoted by α_{fw} and α_{bw}. The outputs of the forward and the backward passes at the same time instance t are combined by the *merge* function.

t

2.1.3.2 Optimization objective

Similarly to the FFNN, the optimization of the RNN (and BRNN) is also typically performed in a supervised manner by using a set of labeled data. The goal is to obtain a set of neural network parameters for the recurrent cell (denoted by ϑ), such that the loss $L(\vartheta)$, given by

$$\mathscr{L}(\boldsymbol{\theta}) = \frac{1}{|S|} \sum_{((\ldots,\mathbf{x}_{t-1},\mathbf{x}_t,\mathbf{x}_{t+1},\ldots),\tilde{\mathbf{x}}_t)\in S} \ell\left(\tilde{\mathbf{x}}_t, f_{\text{RNN},t}(\ldots,\mathbf{x}_{t-1},\mathbf{x}_t,\mathbf{x}_{t+1},\ldots)\right), \tag{2.17}$$

2.1.4 Generative adversarial network (GAN)

2.1.4.1 Data processing

The generative adversarial network (GAN) is an efficient tool for the learning of generative models [84]. Such models are used to generate new data instances by learning the probability distribution of a real dataset. In particular, the generative model is used to simulate a random vector $\mathbf{y}$, which follows some distribution $p(\mathbf{y})$ that is not known, but for which a representative set of data $\mathbf{y}_k$, $k = 1\ldots K$ (samples

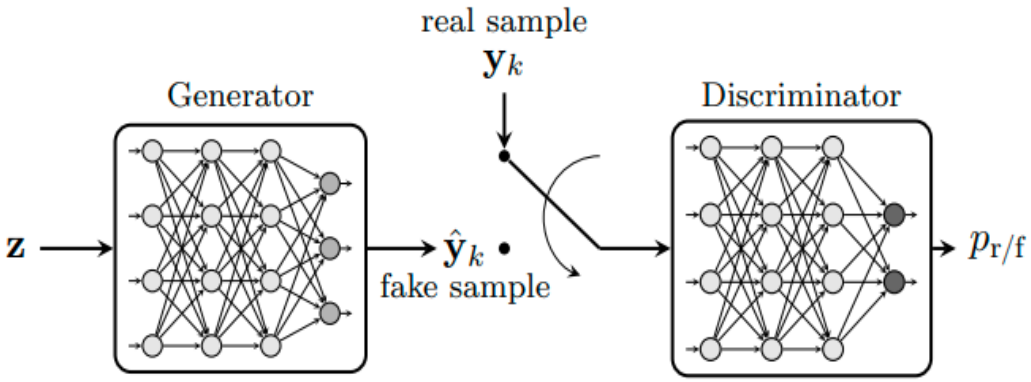

Figure 2.8: Schematic representation of a generative adversarial network (GAN). The generator aims at transforming the random input vector $\mathbf{z}$ into a vector $\hat{\mathbf{y}}_k$, which is statistically identical to $\mathbf{y}_k$ – a sample vector from the distribution of a real data set. The discriminator aims at correctly classifying real and fake samples.

from $p(\mathbf{y})$) is available.

The GAN model can be viewed as similar to the inverse transform method [85, Ch. 2] for generating random variables that follow a given distribution by "reshaping" random variables with simpler distributions such as the normal or uniform distributions [86]. Figure 2.8 shows a schematic of a generative adversarial network. The GAN employs two artificial neural networks, typically with feedforward architectures, that have competing objective functions (see Sec. 2.1.4.2). The generator ANN aims at translating its input into a (high-dimensional) output vector, mimicking the ground truth distribution of the real dataset. The discriminator acts as a binary classifier between real (ground truth) and fake (generated) vectors. To achieve its goal, the generator takes as an input an l-dimensional vector of random samples, which are typically uniformlyor normally-distributed, i.e. $\mathbf{z} \sim U_l^{(0,1)}$ or $\mathbf{z} \sim N_l^{(0,1)}$, where $U^{(0,1)}$ denotes the uniform distribution over the interval (0, 1) and $N^{(0,1)}$ denotes the standard normal distribution. The GAN generator transforms $\mathbf{z}$ into the fake vector sample

$$\hat{\mathbf{y}}_k = G_{\vartheta_\mathrm{g}}(\mathbf{z}), \tag{2.18}$$

where G_{ϑ_g} is an operator describing the function of the generator feedforward ANN, with ϑ_g denoting the set of all its weight and bias parameters. Subsequently, the discriminator is first fed with a sample $\mathbf{y}_k$ from the real dataset, yielding the output

$$p_\mathrm{r} = D_{\vartheta_\mathrm{d}}(\mathbf{y}_k), \tag{2.19}$$

where D_{ϑ_d} is the function of the discriminator feedforward ANN, with ϑ_d being the set of all its weight and bias parameters. Afterwards, the the generated (fake)

sample $\hat{\mathbf{y}}_k$ is also fed to the discriminator, producing the output

$$p_{\mathrm{f}} = D_{\vartheta_{\mathrm{d}}}(\hat{\mathbf{y}}_k). \tag{2.20}$$

Typically, the discriminator ANN is designed such that the outputs p_{r} and p_{f} are probabilities, indicating the confidence whether the input was real or fake.

2.1.4.2 Optimization objective

For optimization of the generator and discriminator ANN parameters, labels l_{r} and l_{f} are assigned for the real and fake samples, respectively. When obtaining the loss L_G of the generator ANN, the parameters ϑ_{d} at the discriminator are assumed to be fixed. As a result, the loss is a function of ϑ_{g} given by

$$\mathscr{L}_G(\boldsymbol{\theta}_{\mathrm{g}}) = \frac{1}{|S|} \sum_{\mathbf{z}_i, l_{\mathrm{r}} \in S} \ell\left(l_{\mathrm{r}}, D_{\boldsymbol{\theta}_{\mathrm{d}}}\left(G_{\boldsymbol{\theta}_{\mathrm{g}}}(\mathbf{z}_i)\right)\right), \tag{2.21}$$

where the training set S is formed by the pairings of a random input vector $\mathbf{z}$ and the label for real samples l_{r}, while $A(\cdot)$ denotes the utilized loss function such as the cross entropy. Note that when dealing with probability outputs, the labels can be conveniently set to $l_{\mathrm{r}} := 1$ and $l_{\mathrm{f}} := 0$, for the real and fake samples, respectively.

On the other hand, the loss of the discriminator L_D, assuming a fixed set ϑ_{g} of generator parameters, can be expressed as

$$\mathscr{L}_D(\boldsymbol{\theta}_{\mathrm{d}}) = \frac{1}{|S|} \sum_{(\mathbf{z}_i, l_{\mathrm{f}}), (\mathbf{y}_i, l_{\mathrm{r}}) \in S} \left[\ell\left(l_{\mathrm{r}}, D_{\boldsymbol{\theta}_{\mathrm{d}}}(\mathbf{y}_i)\right) + \ell\left(l_{\mathrm{f}}, D_{\boldsymbol{\theta}_{\mathrm{d}}}\left(G_{\boldsymbol{\theta}_{\mathrm{g}}}(\mathbf{z}_i)\right)\right)\right]. \tag{2.22}$$

Note that here the training set is formed by two pairings – one between the generator input vector $\mathbf{z}$ and the label for fake samples l_{f}, and another between the sample vector $\mathbf{y}_i$ from the real distribution and the label for real samples l_{r}.

The parameters of the generator and discriminator are optimized iteratively. The objective for the discriminator is to minimise $L_D(\vartheta_{\mathrm{d}})$, thus learning to classify its real ($\mathbf{y}$) and fake ($\hat{\mathbf{y}} = G_{\vartheta_{\mathrm{g}}}(\mathbf{z})$) inputs correctly. On the other hand, by minimising the loss $L_G(\vartheta_{\mathrm{g}})$, the generator model learns to translate its random input vector $\mathbf{z}$ into representations $\hat{\mathbf{y}} = G_{\vartheta_{\mathrm{g}}}(\mathbf{z})$ which are statistically similar to the samples $\mathbf{y}$ from the real distribution. The goal of such an adversarial ANN training algorithm is to converge to parameter sets for which the generator output, given a random input, approximates the distribution of the real data, while the discriminator is unable to differentiate between the two distributions (see [84, Sec. 4]).

2.2 Deep learning

Deep learning techniques, applied for the optimization of neural network parameters, play an integral role in exploiting their potential for universal function approximation. This section introduces the gradient descent algorithm – a key deep learning strategy for optimization of the ANN parameters. It also explains the backpropagation technique which enables efficient computation of gradients in deepANNs.

2.2.1 Gradient descent optimization

Gradient descent (GD) is a powerful and efficient method for minimisation of a function $f(\mathbf{x})$. The technique was proposed for the first time by Augustin Cauchy in 1847 [14] and later expanded by Haskell Curry for nonlinear optimization problems [15]. It has a fundamental role in many areas of optimization theory and deep learning [5, 12]. The GD method utilizes the information given by the gradient

$\nabla_{\mathbf{x}} f(\mathbf{x})$ about the slope of the function $f(\mathbf{x})$ at $\mathbf{x}$. It aims at reducing $f(\mathbf{x})$ by adjusting $\mathbf{x}$ in small steps with a sign (direction) opposite to that of the gradient. As a simple illustrative example, consider the function

$$f(x) = x^2, \tag{2.23}$$

of the scalar variable $x \in \mathrm{R}$, whose derivative is $f'(x) = 2x$. Figure 2.9 shows both $f(x)$ and $f'(x)$. For values $x < 0$, the derivative $f'(x)$ is negative, indicating that the function is decreasing, i.e. $f(x+\eta) < f(x)$, with $\eta > 0$ and thus the input x can be increased by a small amount η to reduce $f(x)$. On the other hand, $f'(x) > 0$ for $x > 0$ (the function is increasing) and thus $f(x-\eta) < f(x)$, i.e. $f(x)$ is reduced by reducing the input with a small amount η. At $x = 0$ the function reaches a stationary point where $f'(x) = 0$. There, the derivative provides no information how to adjust x and the GD method halts. For the considered example, reaching the point $x = 0$, i.e. the end of the GD, coincides with arriving at the global minimum of $f(x)$. This simple GD algorithm can be formally generalized and written using a vector notation as

$$\mathbf{x}_t = \mathbf{x}_{t-1} - \eta \cdot \nabla_{\mathbf{x}} f(\mathbf{x}_{t-1}). \tag{2.24}$$

The update of $\mathbf{x}$ depends on the steepness of the slope of $f(\mathbf{x})$, allowing greater changes for steeper slopes (high absolute value of the gradient) and finer steps around the stationary points. It is important to mention that often, especially in the framework of deep learning, the input is multidimensional and the optimization can encounter and converge to local minima. As a consequence, deep learning is associated with performing *approximate minimisation* of the given objective. In practice, state-of-the-art variants of the GD technique (described in the following

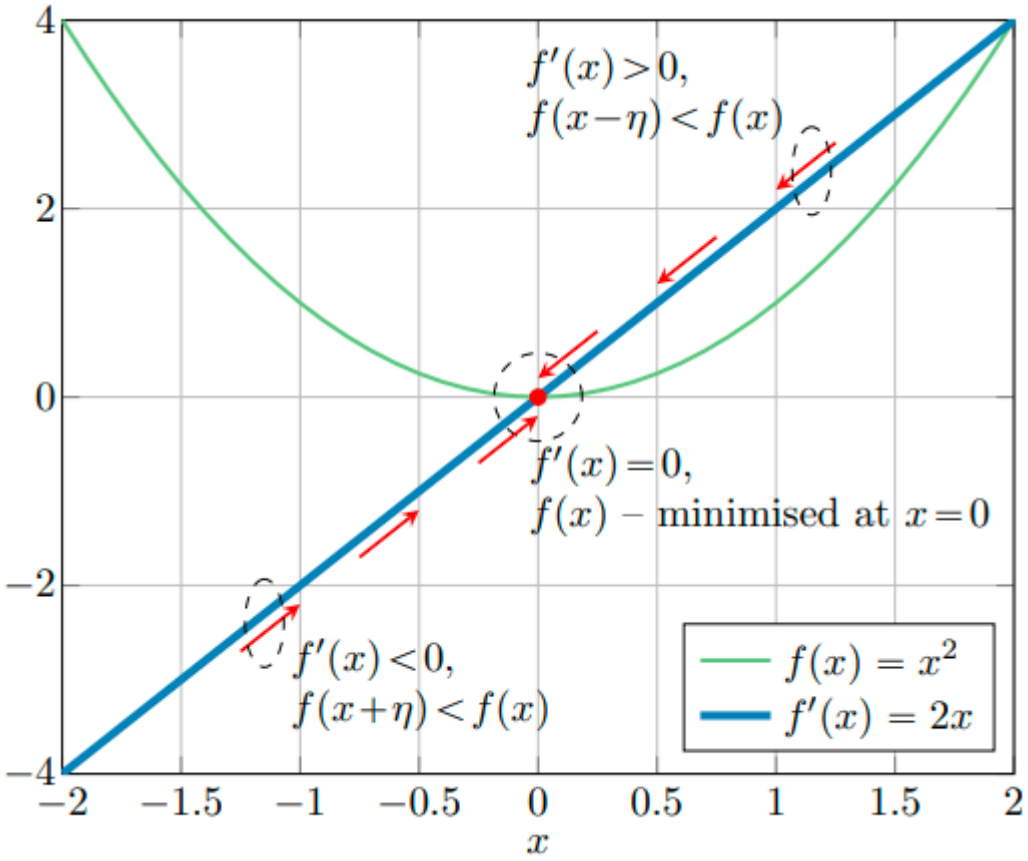

Figure 2.9: Illustration of the gradient descent method for minimisation of the function $f(x)=x^2$. For $f'(x)<0$ or $f'(x)>0$, the input x is increased or decreased, respectively, by a small amount $\eta>0$. The gradient descent algorithm stops when $f'(x)=0$, where the function $f(x)$ is at a minimum.

sections) have been developed to achieve convergence to solutions corresponding to very low function values [12, Sec. 4].

2.2.1.1 Stochastic gradient descent

In order to perform GD minimisation of the computed loss $L(\vartheta)$ in an ANN, the gradient $\nabla_\vartheta L(\vartheta)$ with respect to the ANN parameters needs to be calculated using the entire, typically very large, training set. This imposes a great computational burden to perform the GD algorithm. A commonly employed approach, which reduces the computational demands, is to use a stochastic approximation [87] of the GD, a method known as a stochastic gradient descent (SGD) [12, 18, 88]. The SGD performs optimization of the ANN parameters operating on estimates of the gradient obtained from small randomly-selected sub-sets (called *mini-batches*) of the training data. The algorithm aims at updating the set of ANN parameters ϑ such that the loss $\underline{L}(\vartheta)$ over a mini-batch $\underline{S}$, given by (for FFNN)

$$\underline{\mathscr{L}}(\boldsymbol{\theta})=\frac{1}{|\underline{S}|}\sum_{(\mathbf{x}^0,\tilde{\mathbf{x}}^K)\in\underline{S}}\ell\left(\tilde{\mathbf{x}}^K,f_{\text{FFNN}}\left(\mathbf{x}^0\right)\right), \tag{2.25}$$

is minimised. The mini-batch loss for BRNN (Sec. 2.1.3.2) or GAN (Sec. 2.1.4.2) is correspondingly expressed. On the first step of the SGD algorithm, the ANN parameters ϑ are randomly initialised and then iteratively updated as

$$\vartheta_t = \vartheta_{t-1} - \eta \cdot \nabla_\vartheta \underline{L}\,(\vartheta_{t-1}), \tag{2.26}$$

where η is referred to as the *step-size* of the learning algorithm and $\nabla_\vartheta \underline{L}\,(\vartheta)$ denotes the gradient of the mini-batch loss with respect to the parameters ϑ. The product $\eta \cdot \nabla_\vartheta \underline{L}\,(\vartheta)$ determines the learning rate and speed of convergence of the SGD algorithm and thus many variants for its adaptation has been considered. An

extensive overview and comparison of many popular implementations can be found in [18] and [12, Sec. 8.3-8.7]. In particular, the "adaptive moment" estimation (Adam) algorithm has been specifically developed for the training of deep ANNs and has become the de facto default optimization method in the field of deep learning [17, 18, 89]. The Adam optimizer is introduced in the following section and has been used for the learning of ANN parameters for the research covered by this thesis.

2.2.1.2 Adaptive moment estimation (Adam) optimizer

The Adam algorithm is a computationally efficient variant of the SGD strategy with an improved convergence achieved by the dynamic adaptation of the learning rate [16]. To control the learning rate, the method utilizes biased moving averages of the gradient μ_t and the squared gradient v_t, estimated as [16, Sec. 2]

$$\mu_t = \beta_1 \cdot \mu_{t-1} + (1-\beta_1) \cdot \nabla_\theta \underline{\mathscr{L}}(\theta_{t-1}), \tag{2.27}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot (\nabla_\theta \underline{\mathscr{L}}(\theta_{t-1}))^2, \tag{2.28}$$

respectively. Where $()^2$ denotes the element-wise square function, and β_1 and β_2 are the corresponding exponential decay rates, whose default values are empirically

evaluated and set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$ (see [16, Sec. 6.4]). Next, the terms are biased-corrected .

Subsequently, the update of the ANN parameters is governed by

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{\mu}_t}{(\sqrt{\hat{v}_t} + \varepsilon)} \tag{2.29}$$

where ϵ is a small constant used for numerical stability, which is typically set to $\epsilon = 10^{-8}$ as originally suggested in [16, Sec. 2]).

2.2.2 Backpropagation algorithm

The calculation of the gradient of the scalar mini-batch loss $\nabla_{\vartheta} \underline{L}\ (\vartheta)$ with respect to the ANN parameters is a key aspect in the optimization procedure. A simple and computationally efficient method for the computation of gradients, first proposed in [13], is the so-called *backpropagation* algorithm. In order to introduce the algorithm, this section starts with the representation of ANNs as computational graphs, followed by the description of backpropagation, which consists in recursively applying the chain rule of calculus [90] across the graph nodes.

2.2.2.1 Computational graph of an ANN

Figure 2.10 shows an example of a computational graph which represent the process of computing the loss function in an artificial neural network. The nodes on the graph indicate the *variables* (in scalar, vector, matrix or other multi-dimensional form) that are subject to *operations* (e.g. matrix multiplication, addition, etc.). In particular, for illustrative purposes, the presented graph uses as a reference a deep fully-connected feedforward ANN (see Sec. 2.1.2). As already discussed, this network translates its input $\mathbf{x}^0$ to an output $\mathbf{x}^K$ through the series of operations of the form $\mathbf{x}^k = \alpha_k(\mathbf{W}_k\mathbf{x}^{k-1} + \mathbf{b}_k)$, $k = 1, .., K$. Note that in the computational graph this transformation is expressed as

$$\mathbf{u}_{k1} = \mathbf{W}_k\mathbf{x}^{k-1}, \tag{2.32}$$

$$\mathbf{u}_{k2} = \mathbf{u}_{k1} + \mathbf{b}_k, \tag{2.33}$$

$$\mathbf{x}^k = \alpha_k(\mathbf{u}_{k2}), \tag{2.34}$$

such that all intermediate expressions are represented by a node. Importantly, in addition to the ANN operations, the scalar loss L (see Sec. 2.1.2, Eq. (2.5)) between the output $\mathbf{x}^K$ of the ANN and the corresponding training label $\tilde{\mathbf{x}}^K$ is also computed. This is indicated by the final node on the graph. Such a representation of an ANN and its loss computation, highlights the dependencies of the optimization objective on all nodes in the deep network. In particular, the goal of the optimization procedure is to update the trainable nodes in the graph (highlighted in grey colour on the schematic) such that L is minimised.

2.2.2.2 Backpropagation algorithm

The backpropagation algorithm consists in recursively applying the chain rule of calculus for computing gradients of functions [13], [12, Sec. 6.5]. As an example,

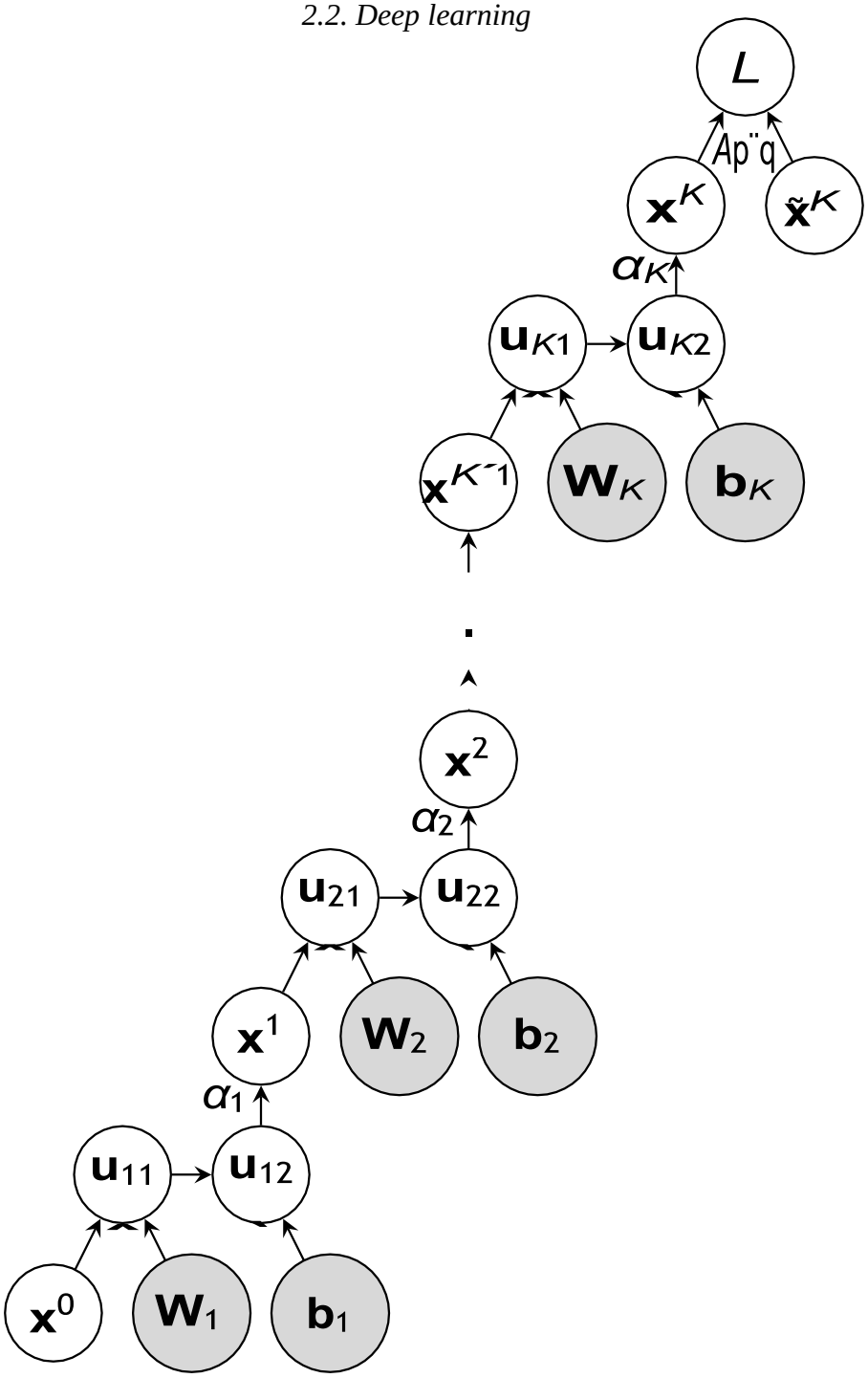

Figure 2.10: Computational graph representation of the process of computing the loss in an artificial neural network. For illustrative purposes, the feedforward network described in Sec. 2.1.2 is used as an ANN reference. It transforms its input $\mathbf{x}^0$ to an output $\mathbf{x}^K$ by performing the operations $\mathbf{x}^k = \alpha_k(\mathbf{W}_k\mathbf{x}^{k-1} + \mathbf{b}_k)$, $k = 1, .., K$. This is followed by the computation of a scalar loss L between $\mathbf{x}^K$ and the training labels $\tilde{\mathbf{x}}^K$. Larger grey nodes are used to highlight the trainable parameters in the graph.

assume that the vector $\mathbf{x} \in \mathrm{R}^m$ is mapped to $\mathbf{y} \in \mathrm{R}^n$, i.e. $\mathbf{y} = f_1(\mathbf{x})$, and then the scalar variable $Z \in \mathrm{R}$ is computed via a different mapping $Z = f_2(\mathbf{y})$. To obtain the gradient $\nabla_{\mathbf{x}} Z$ of Z with respect to $\mathbf{x}$, the chain rule can be utilized ANN graph need to be computed. Using the computational graph from Fig. 2.10 as a reference, the backpropagation algorithm allows to obtain the gradients $\nabla_{\mathbf{W}_k} L$ and $\nabla_{\mathbf{b}_k} L$, $k = 1 \ldots K$ for the network weights and bias parameters. They are subsequently utilized in the gradient descent optimization procedure, which was described in Section 2.2.1. The backpropagation algorithm, whose detailed stepby-step description is given in Algorithm 1, is initialised by computing the gradient $\nabla_{\mathbf{x}^K} L$ of the loss L with respect to the final ANN output $\mathbf{x}^K$. It then proceeds with computing gradients across the computational graph of the ANN, moving in the backward direction, until all gradients of L , from the K-th to the first layer of the ANN, have been computed. It is worth mentioning that in the case of a recurrent ANN, the computational graph will include dependencies between different time instances. In this case, the utilized algorithm for computation of gradients, identical in its essence to the presented example, is often referred to as *backpropagation through time*.

Chapter 3

Optical fiber communication systems for short reach

Short reach optical communications are found in many data center, metro and access network scenarios. Such systems crucially rely on the simplicity and costeffectiveness of a signal transmission scheme based on intensity modulation with direct detection (IM/DD) [23, 70]. The short reach links typically consist of a single fiber span, thus avoiding the associated costs of deploying in-line optical amplification. The IM/DD technology enables the utilization of electronic and optical components that are inexpensive, have a small footprint and low power consumption. It is thus considered as a prime candidate for *fiber to the home* (FTTH) systems, realized, for instance, via passive optical networks (PONs).

The optical IM/DD communication links are mainly characterised by the presence of fiber chromatic dispersion, which introduces inter-symbol interference (ISI), and nonlinear (square-law) photodiode (PD) detection. Moreover, due to the hardware imperfections and bandwidth limitations in the low-cost components, the signal is corrupted by a significant amount of noise and nonlinearity stemming from multiple sources at both transmitter and receiver, i.e. modulators, electrical amplification circuits as well as low resolution digital-to-analogue and analogue-to-digital converters. Meeting the ever-increasing data rate demands for such systems becomes quite a challenging task because of the limitations imposed by these impairments and the application of advanced DSP algorithms has attracted great interest in recent years [66, 91, 92].

In this chapter, the transformations and distortions undergone by the signal propagating between the output of the transmitter DSP module and the input to the DSP module at the receiver are described in detail. Figure 3.1 shows a general schematic of an optical intensity modulation/direct detection system, highlighting the parts of the transmission link where digital-to-analogue/analogue-to-

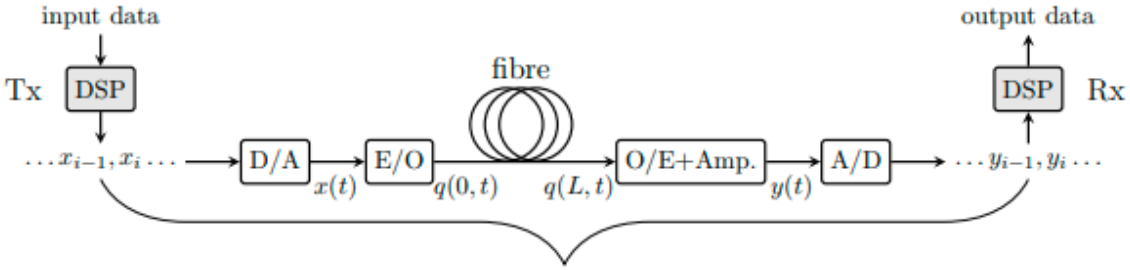

Figure 3.1: Schematic of an IM/DD communication system, indicating the signal transformations across the transmission link. Tx transmitter, DSP digital signal processing, D/A digital to-analogue conversion, E/O electro-optical conversion, O/E opto-electrical conversion, Amp. amplification, A/D analogueto-digital conversion, Rx receiver.

digital, electro-optical/opto-electrical conversions and optical fiber propagation are performed. The remaining chapters of this thesis are concerned with proposing DSP schemes allowing reliable communication over such a channel.

3.1 Transmitter

3.1.1 Digital-to-analogue conversion

The output samples $(\ldots x_{i-1}, x_i, x_{i+1} \ldots) \in \mathbb{R}_{\geq 0}$ from the DSP module at the transmitter are transformed into an analogue electrical signal $x(t)$ using a digital-toanalogue converter (DAC). The device is characterised by its sampling period T_s [s] from which the DAC sampling rate is obtained as

$$f_s = \frac{1}{T_s} \quad \text{[Hz]}. \tag{3.1}$$

Accordingly, the Nyquist frequency of the DAC is then given by

$$f_{\text{Nyq.}} = \frac{f_s}{2}. \tag{3.2}$$

The significance of this expression is that the digital signal applied for DAC conversion must have a frequency content limited to $f_{\text{Nyq.}}$ in order to be fully recoverable. In practice, due to hardware limitations of the device, the actual DAC bandwidth is smaller than $f_{\text{Nyq.}}$ [93] and typically the signal is low-passed filtered before the converter. The output of the low-pass filter (LPF) can be expressed as

$$x'(t) = s(t) * g_1(t) = \sum_{i=-\infty}^{\infty} x_i g_1(t - iT_s), \tag{3.3}$$

where $g_1(t)$ is the impulse response of the LPF, while the train of input samples is represented by the temporal train of T_s-spaced impulses

$$s(t) = \sum_{i=-\infty}^{\infty} x_i \delta(t - iT_s), \tag{3.4}$$

with $\delta(\cdot)$ denoting the Dirac delta function. The impulse response of the so called *brick-wall* LPF, which has a unit frequency gain and sets the the bandwidth of the real-valued signal to B_f is given by

$$g_{\text{LPF}}(t) = \text{sinc}(2B_f \pi t). \tag{3.5}$$

The DAC distorts $x'(t)$ as a consequence of its limited quantisation resolution. This distortion is modeled as an additive and uniformly distributed *quantisation noise* [94, 95] and the real-valued analogue electrical signal at the output of the DAC can correspondingly be expressed as

$$x(t) = x'(t) + n_1(t), \tag{3.6}$$

where $n_1(t)$ denotes the noise contribution. In particular, the output signal-to-noise ratio (SNR) of the DAC, and thus the variance of $n_1(t)$, is determined by the effective number of bits (ENOB) of the device. The theoretical output SNR of a digital-to-analogue converter with an ENOB of N_b bits is given by [95]

$$\text{SNR} = 6.02N_b + 1.76 \quad \text{[dB]}. \tag{3.7}$$

The noise $n_1(t)$ is zero mean, i.e.

$$\mathrm{E}(n_1(t)) = 0, \tag{3.8}$$

where $\mathrm{E}(\cdot)$ denotes the expectation operator. The corresponding quantisation noise variance then becomes

$$\sigma_{n_1}^2 = 3\mathbb{E}\left(|x'(t)|^2\right) \cdot 10^{-(6.02 \cdot N_b + 1.76)/10}. \tag{3.9}$$

For state-of-the-art high-speed DACs the typical N_b values are between 5 and 6 [96].

3.1.2 Electro-optical conversion

At the electro-optical conversion stage at the transmitter, a Mach-Zehnder modulator (MZM) is employed to externally modulate the amplitude of an optical carrier wave, generated by a laser source. The MZM is driven by the electrical signal $x(t)$.

Neglecting laser fluctuations, the electric field of the laser light is given by

$$\xi(t) = |E_0| \cdot \exp(-j\omega_c t), \tag{3.10}$$

where $|E_0|$ is the field amplitude and ω_c is the angular frequency. The MZM modulates $\xi(t)$ with the information-carrying signal $x(t)$, resulting in the complex-valued *bandpass* optical signal

$$q^{-}(0,t) = \sin(x(t)) \cdot |E_0| \cdot \exp(-j\omega_c t), \tag{3.11}$$

where $\sin(\cdot)$ is modeling the modulator electric field transfer function [97]. It acts on the information signal $x(t)$, whose amplitude thus needs to be restricted in the $[-\pi/2, \pi/2]$ interval, e.g. by proper scaling in the DSP. Omitting the carrier term from Eq. (3.11), the modulated real-valued *baseband* signal becomes

$$q(0,t) = |E_0| \cdot \sin(x(t)). \tag{3.12}$$

This scheme is referred to as *Intensity Modulation* (IM) since varying the amplitude of the laser field, results in modulation of the optical intensity $I(0, t)$ at the fiber input, with the two quantities related as

$$I(0, t) \propto |q(0,t)|^2. \tag{3.13}$$

3.2 Propagation over the optical fiber

The propagation of the optical signal $q(z, t)$ through the fiber medium can be described by the second-order nonlinear partial differential equation (PDE)

$$\frac{\partial q(z,t)}{\partial z} = -\frac{\alpha}{2} q(z,t) - j\frac{\beta_2}{2}\frac{\partial^2 q(z,t)}{\partial t^2} + j\gamma |q(z,t)|^2 q(z,t) \tag{3.14}$$

where α, β_2, and γ are the fiber attenuation, chromatic dispersion and Kerr nonlinearity coefficients, respectively, z denotes the distance along the fiber, and $j = \sqrt{-1}$ is the imaginary unit. Equation (3.14) is referred to as the nonlinear Schrödinger equation (NLSE).

The nonlinear Kerr effect is associated with local changes of the fiber refractive index due to the intensity of the optical field [24]. Typically, for IM/DD systems the intensity at the fiber input $I(0, t)$ is small and subsequently attenuated by the medium. Moreover, as already discussed, such systems do not employ in-line signal amplification. As a consequence, the impact of the Kerr nonlinearity during

propagation along the fiber can be neglected [23]. The fiber-induced attenuation of the signal is taken into account in the definition of the noise at the receiver, presented in Sec. 3.3.

Chromatic dispersion is the dominant fiber effect for IM/DD transmission systems. According to the NLSE, its operation on the propagating signal is expressed by

$$\frac{\partial q(z,t)}{\partial z} = -j\frac{\beta_2}{2}\frac{\partial^2 q(z,t)}{\partial t^2} \tag{3.15}$$

which is a second-order linear PDE that can be solved analytically in the frequency domain, where it is re-written as

$$\frac{\partial Q(z,\omega)}{\partial z} = j\frac{\beta_2}{2}\omega^2 Q(z,\omega), \tag{3.16}$$

with a solution given by

$$Q(z,\omega) = Q(0,\omega)\exp\left(j\frac{\beta_2}{2}\omega^2 z\right) \tag{3.17}$$

2

where

$$Q(0,\omega) = F\{q(0,t)\}, \tag{3.18}$$

is the Fourier transform of the transmit signal $q(0, t)$. The time domain optical signal at distance z can then be recovered as

$$q(z,t) = F^{-1}\{Q(z,\omega)\}, \tag{3.19}$$

where the operator F^{-1} denotes the inverse Fourier transformation. Often the dispersion of the optical fiber is specified by the parameter D, from which β_2 can be obtained as

$$\beta_2 = -\frac{\lambda^2}{2\pi \cdot c}D, \tag{3.20}$$

where λ denotes the wavelength, while c is the speed of light in vacuum. Equation (3.17) shows that chromatic dispersion induces a propagation delay between the different components in the frequency content of the propagating pulses. This causes their temporal broadening, an effect which accumulates linearly with distance and quadratically with bandwidth. As a result, a pulse will interact and interfere with other preceding and succeeding pulses during propagation through the fiber medium. Chapter 4 and Chapter 5 explain why such a distortion can lead to inter-symbol interference (ISI), introducing *memory* in the communication channel and imposing severe limitations on the IM/DD optical fiber transmission.

3.3 Receiver

3.3.1 Opto-electrical conversion and amplification

The signal $q(z, t)$ at the the output of a fiber span of length z is converted from the optical back to the electrical domain by a photodetector. In IM/DD systems, a simple *p-i-n photodiode* (PD) is used to perform the *square-law* detection of the incoming optical signal [23, 70]. The output photocurrent of the PD is proportional to the intensity of the impinging optical field, i.e.

$$y'(t) = \rho \, |q(z,t)|^2, \tag{3.21}$$

where $\rho \in (0, 1]$ is the responsivity of the photodetector. Such a nonlinear signal reception using a single PD is called *Direct Detection* (DD).

The detection is followed by the electrical amplification of the signal, attenuated from the propagation along the fiber link. The amplification is often performed by a *transimpedance amplifier* (TIA) and in practice the PD and TIA components are combined in a single package, referred to as a PIN+TIA module. The TIA amplification circuit is associated with corrupting the signal with an additive white Gaussian noise [70, 98], i.e.

$$y(t) = y'(t) + n_2(t), \tag{3.22}$$

The noise is modeled as zero-mean, while its variance $\sigma^2_{\text{ref.}}$ can be estimated using the measured signal-to-noise ratio $\text{SNR}_{\text{ref.}}$ at the output of the amplifier at a reference distance. Taking the fiber attenuation into account will yield different effective SNRs at each transmission distance z. To model this, the variance of the noise $n_2(t)$ added to the signal as a function of z is obtained as

$$\sigma^2_{n_2}(z) = \frac{\sigma^2_{\text{ref.}}}{\exp(-\alpha z)}, \tag{3.23}$$

where α [1/km] is the attenuation coefficient of the fiber in linear units. Typically fiber attenuation is specified in decibel units ($\alpha_{\text{dB/km}}$) and thus the conversion $\alpha = \alpha_{\text{dB/km}} / 4.343$ should be considered. Typically, $\alpha_{\text{dB/km}} = 0.2$ [dB/km] for a standard single mode fiber, effectively resulting in doubling of the noise every 15 km.

3.3.2 Analogue-to-digital conversion

In the next step, the amplified electrical signal $y(t)$ needs to be converted into a digital form. This is achieved using an analogue-to-digital converter (ADC). Prior to the ADC, similarly to the transmitter chain of operations, the signal is low-pass filtered to account for the hardware-induced bandwidth restrictions. The signal can thus be expressed as

$$\bar{y}(t) = y(t) * g_2(t) = \int_{-\infty}^{\infty} y(\tau) g_2(t-\tau), \tag{3.24}$$

where $g_2(t)$ is the impulse response of the receiver LPF. The function of the ADC can be modeled in two stages – first, it contributes a zero-mean additive uniformlydistributed quantisation noise $n_3(t)$ to $\hat{y}(t)$, i.e.

$$\hat{y}(t) = \bar{y}(t) + n_3(t), \tag{3.25}$$

whose variance, similarly to the DAC, is given by Eq. (3.9) and thus depends on the ENOB specification of the ADC. Second, it performs sampling of the continuous time analog electrical signal with period T_s (typically identical to the DAC), recovering the sequence of received samples as

$$\mathbf{y} = \sum_{i=-\infty}^{\infty} \hat{y}(t)\delta(t - iT_s), \tag{3.26}$$

where $\mathbf{y} = (\ldots y_{i-1}, y_i, y_{i+1} \ldots)$ is used to denote the digitised received signal. The obtained train of received samples is utilized by the receiver DSP module, where typical functions include equalisation of the channel distortions and recovery of the transmitted messages.

3.4 Communication channel model as a computational graph

In the context of end-to-end optimization of communication systems via deep learning, which is investigated in this thesis, the complete chain of transmitter DSP, communication channel model and receiver DSP is implemented as a deep artificial neural network. This novel concept, which is described in detail in the following chapters, allows to optimize the end-to-end signal processing over all constraints imposed by the channel. Such a framework can be readily applied to scenarios where the channel model is known and differentiable. As a consequence, the model can be interpreted as being a part (typically with un-trainable parameters) of a computational graph, similar to the one presented in Sec. 2.2.2. The graph represents the complete communication system and spans from the input messages at the transmitter to the output of the receiver, where the system loss is calculated during optimi-

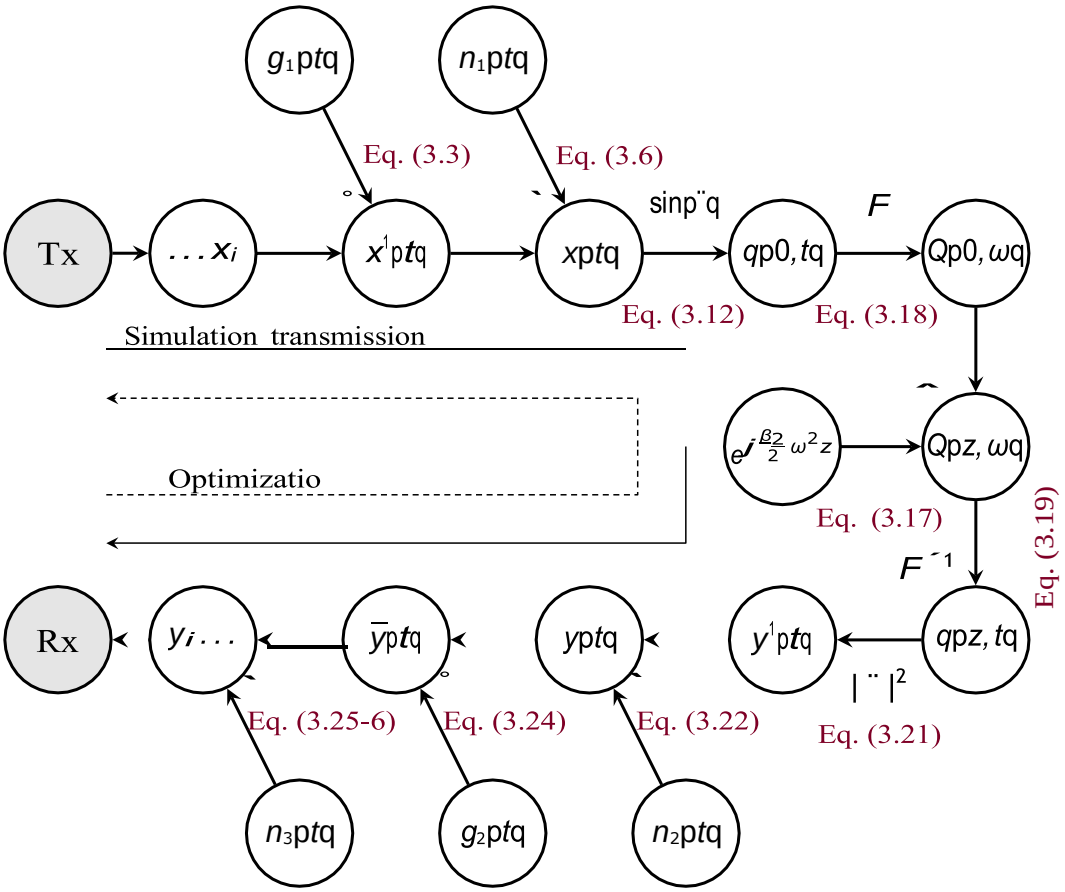

Figure 3.2: Schematic showing the communication channel model, which is considered as a segment of the end-to-end computational graph, representing the optical fiber communication system.

sation.

Figure 3.2 shows an illustrative schematic of such a representation for the optical IM/DD channel model described in this chapter. The filtering stages at transmitter and receiver as well as the chromatic dispersion contribution can be modeled as purely linear stages of the ANN, i.e. multiplications of the signal with correspondingly chosen matrices. On the other hand, the modulation and photodetection stages are modeled by purely nonlinear functions, while the effects of DAC, ADC and the electrical amplifier can be straightforwardly applied as noise layers. The fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT) algorithms, necessary for conversion between time and frequency domain, also form part of the end-to-end computational graph and are readily available in state-of-the-art deep learning libraries such as Tensorflow [99]. Crucially, a channel model which can be represented in such a manner allows to utilize the backpropagation algorithm and compute gradients for the trainable parameters in the transmitter DSP module with respect to an end-to-end loss metric computed at the receiver. Such a technique is

used for the numerical investigation of the developed optical fiber auto-encoders based on feedforward and recurrent neural networks in Chapter 4 and Chapter 5, respectively. Moreover, it is also utilized in Chapter 6, where the experimental performance of transceivers optimized in simulation-only using the channel model and applied “as is” to the actual transmission link was investigated.

Chapter 4

Design of optical fiber systems using end-to-end deep learning

In this chapter, the concept of end-to-end deep learning-based transceiver for communication over optical fiber channels is developed. In particular, the design, optimization and performance of a feedforward ANN (see Sec. 2.1.2) optical fiber auto-encoder is investigated for the IM/DD communication channel described in Chapter 3. Adopting Claude Shannon's famous definition [100], engineering a communication system over a channel can be viewed as the task of designing a digital transmitter and receiver pair which allows reliable transmission of messages. In particular, the transmitter takes a sequence of messages as an input and encodes it into a sequence of robust samples, fed to the channel, while the receiver aims at reproducing (decoding) with maximum accuracy the transmitted messages from the received distorted samples.

Artificial neural networks and deep learning provide the framework to design the optical communication system by carrying out the optimization of the encoding and decoding functions in a single end-to-end process. As explained in Sec. 1.1.2, such fully learnable communication systems (auto-encoders) are based on the idea of implementing the complete chain of transmitter, channel and receiver as an endto-end deep artificial neural network, as first proposed in [59–61]. Using deep learning, the transmitter and receiver sections are optimized jointly, such that a set of ANN parameters is obtained for which the end-to-end system performance is optimized for a specific metric.

This chapter begins with a more detailed description of the auto-encoder concept applied to optical fiber communications in Sec. 4.1. The remaining sections describe the system implementation and optimization as end-to-end FFNN as well as the simulation performance over the optical IM/DD transmission model.

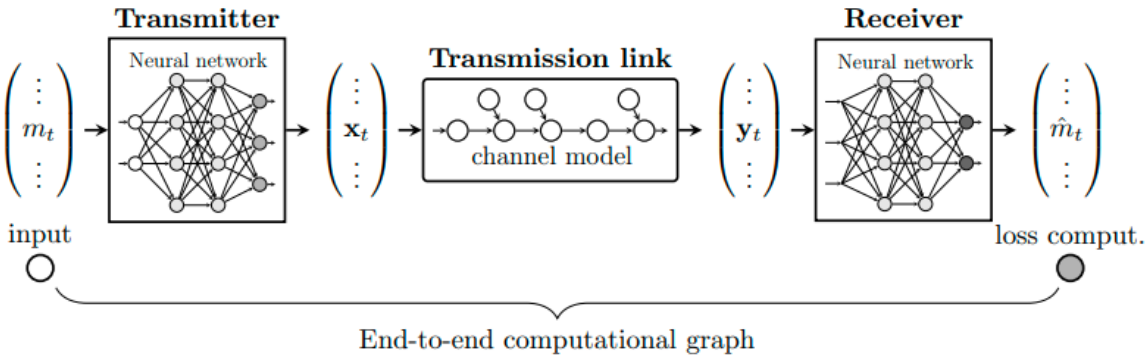

Figure 4.1: Schematic of the optical communication system implemented as an end-toend computational graph, which represents the process of computing the loss between the transmitter input and the receiver output.

4.1 The optical fiber system as an end-to-end computational graph

Figure 4.1 shows a schematic of the entire optical fiber communication chain of transmitter, channel model and receiver implemented as an end-to-end deep ANN. The transmitter encodes the sequence of random messages $(\ldots m_{t-1}, m_t, m_{t+1} \ldots)$ from a finite alphabet M into a sequence of symbols $(\ldots \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \ldots)$, with each symbol $\mathbf{x}$ being a vector (block) of n digital *waveform* samples. The produced digital waveform is fed to the transmission link, where each sample acquires noise as well as interference from preceding and succeeding samples, a process that is governed by the IM/DD transmission model described in Chapter 3. The output of the channel is distorted waveform samples. These are accordingly organized into the sequence of received symbols $(\ldots \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$. The symbols are decoded by the receiver in order to obtain the sequence of recovered messages $(\ldots \hat{m}_{t-1}, \hat{m}_t, \hat{m}_{t+1} \ldots)$. At this point during the optimization stage, a loss between the transmitted and received messages $L = \sum_t A(m_t, \hat{m}_t)$ is computed. Because both the ANN transceiver architecture (see Sec. 2.2.2.1) and the communication chan-
nel model (see Sec. 3.4) are suitable for a computational graph representation, the complete communication system spanning from the transmitter input to the receiver output can be considered as an end-to-end computational graph. Utilizing the dependencies on the graph, the backpropagation algorithm (see Sec. 2.2.2.2) can be readily applied to obtain a gradient of the loss with respect to every trainable parameter at both transmitter and receiver. The transceiver is optimized via gradient descent, described in Sec. 2.2.1, aimed at minimising L , e.g. the message (symbol) error rate of the system. The auto-encoder systems are particularly suitable in scenarios, such as optical fiber communication, where the optimum transmitterreceiver pair is unknown or computationally prohibitive.

4.2 Feedforward neural network-based design

This section proposes and describes the design of an optical IM/DD communication system implemented as an end-to-end deep feedforward artificial neural network (FFNN). The performance of the FFNN auto-encoder is first investigated numerically as a function of transmission distance. Additionally, different training strategies for obtaining optimized transmitter and receiver parameters are discussed. Since FFNN is the simplest and most common ANN architecture, the performance and the implementation of this auto-encoder formed the benchmark for development of other, eventually more advanced end-to-end deep learning-based communication schemes, which are described in Chapter 5 and Chapter 6. These two chapters also present detailed comparisons both in simulations and transmission experiments between the auto-encoder and other state-of-the-art DSP algorithms for IM/DD.

4.2.1 Transmitter structure

Figure 4.2 shows a schematic of the FFNN-based transmitter. As already discussed, its function is to encode the sequence of input messages $(\ldots m_{t-1}, m_t, m_{t+1}\ldots)$ into a corresponding sequence of robust symbols (blocks of digital waveform samples) $(\ldots \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}\ldots)$. Each message in the sequence $m \in \{1, \ldots, M\}$ is independently chosen from a set of $|M| = M$ total messages. It is encoded by the transmitter into a block $\mathbf{x}$ of n transmit samples.

4.2.1.1 Block-by-block encoding

As explained in Sec. 2.1.2, the feedforward neural network processes each input independently. Therefore, an FFNN architecture at the transmitter can be used to encode the input message m_t into a symbol $\mathbf{x}_t$ in a process that does not utilize any information from the encoding of m_{t-k} into $\mathbf{x}_{t-k}$, $k = 1 \ldots t-1$ or the encoding of m_{t+k} into $\mathbf{x}_{t+k}$, $k = 1 \ldots N_{seq.} - t$, where $N_{seq.}$ is the length of the message sequence. Such an FFNN-based encoding scheme is shown in Fig. 4.2. Next, the processing of messages by the specific FFNN used as a transmitter in the presented investigation is described.

4.2.1.2 FFNN processing

Table 4.1 lists the FFNN *hyper-parameters*, defined as the neuron layers, their dimensions and activation functions, which specify the transmitter FFNN. The encoding of messages by this network is performed in the following way: First, the message m_t is represented as a one-hot vector (see Sec. 2.1.2.2) of size M, denoted as $\mathbf{1}_{m,t} \in \mathrm{R}^M$, where the m-th element equals 1 and the other elements are 0. The one-hot vector $\mathbf{1}_{m,t}$ is then fed to the first hidden layer of the FFNN, whose weight matrix and bias vector are $\mathbf{W}_1 \in \mathrm{R}^{2M \times M}$ and $\mathbf{b}_1 \in \mathrm{R}^{2M}$, respectively. This is fol-

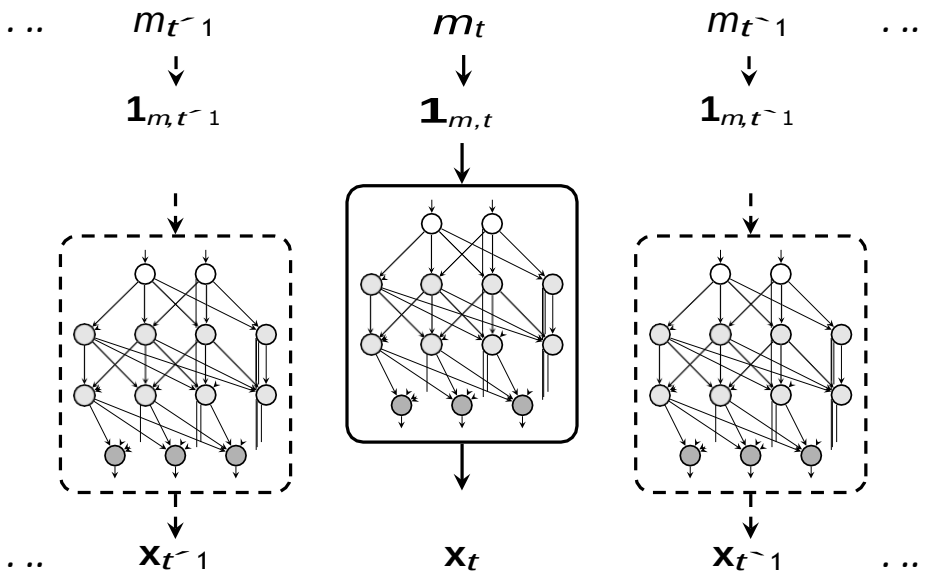

Figure 4.2: Schematic of the FFNN-based transmitter section of the optical fiber autoencoder. The input messages $(\ldots m_{t-1}, m_t, m_{t+1}\ldots)$ are represented as one-hot vectors $(\ldots, \mathbf{1}_{m,t-1}, \mathbf{1}_{m,t}, \mathbf{1}_{m,t+1}\ldots)$, which are processed independently by the FFNN at each time instance to produce the encoded symbols (blocks of samples) $(\ldots \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}\ldots)$, transmitted through the communication channel.

Table 4.1: FFNN definition at the transmitter

Layer	Activation	Output dimension
Input	one-hot encoding	M
Hidden 1	ReLU	$2M$
Hidden 2	ReLU	$2M$
Final	clipping	$n \cdot s$

lowed by a second hidden layer that has parameters $\mathbf{W}_2 \in \mathrm{R}^{2M\times 2M}$ and $\mathbf{b}_2 \in \mathrm{R}^{2M}$. The ReLU activation function (see Sec. 2.1.1) is applied in both hidden layers. The final layer prepares the data for transmission and its output is the encoded block of samples $\mathbf{x}_t$. The layer parameters correspondingly are $\mathbf{W}_3 \in \mathrm{R}^{2M\times n\cdot s}$ and $\mathbf{b}_3 \in \mathrm{R}^{n\cdots}$, where n denotes the number of block samples representing the message, while s is an integer accounting for oversampling, which is discussed in the next section. As explained in Chapter 3, typically for optical systems based on IM/DD, a unipolar signaling is considered. Thus, the output of the final transmitter layer needs to be positive. Moreover, due to the function of the modulator (see Sec. 3.1.2) this output should be within the operational range $[-\pi/2; \pi/2]$ of the MZM. To limit the elements of $\mathbf{x}_t$ in the relatively linear positive region of MZM operation $[0; \pi/4]$, a novel *clipping* activation function, which combines two ReLUs as follows

$$\alpha_{\text{Clipping}}(\mathbf{x}) = \alpha_{\text{ReLU}}(\mathbf{x}) - \alpha_{\text{ReLU}}\left(\mathbf{x} - \frac{\pi}{4}\right) \tag{4.1}$$

is introduced. In terms of deep learning, it is associated with the advantages of a pure ReLU function, while also taking the practical signal design considerations into account. The encoding function of the FFNN-based transmitter $\mathbf{x}_t = f_{\text{Enc.-FFNN}}(\mathbf{1}_{m,t})$ can thus be formally described by the series of operations

$$\mathbf{x}_t^{I} = \alpha_{\text{ReLU}}(\mathbf{W}_1\mathbf{1}_{m,t} + \mathbf{b}_1), \tag{4.2}$$

$$\mathbf{x}_t^{II} = \alpha_{\text{ReLU}}\left(\mathbf{W}_2\mathbf{x}_t^{I} + \mathbf{b}_2\right), \tag{4.3}$$

$$\mathbf{x}_t = \alpha_{\text{Clipping}}\left(\mathbf{W}_3\mathbf{x}_t^{II} + \mathbf{b}_3\right), \tag{4.4}$$

where $\mathbf{x}_t^{I}$ and $\mathbf{x}_t^{II}$ denote the outputs of the first and second hidden layers at the transmitter, respectively. For illustrative purposes, Fig. 4.2 indicates only the transmitter input $\mathbf{1}_{m,t}$ and the corresponding output $\mathbf{x}_t$.

4.2.1.3 Coding rate

Every transmitted message m represents the equivalent of $\log_2(M)$ bits of information, since it is one of M equiprobable choices. The message is encoded into n transmit samples (excluding oversampling), forming a symbol. The system can thus be viewed as an auto-encoder with a coding rate that can be expressed as

$$R_{\text{cod.}} = \frac{\log_2(M)}{n} \quad \text{[b/Sa]}. \tag{4.5}$$

Denoting the DAC sampling rate in the system as $R_{\text{samp.}}$ [Sa/s], the information rate of the auto-encoder becomes

$$R_{\text{inf.}} = R_{\text{cod.}} \cdot R_{\text{samp.}} \quad \text{[b/s]}. \tag{4.6}$$

These two expressions show that the coding rate and consequently the information rate of the communication system can be controlled by adjusting the dimensionality of the input and the final layers of the FFNN. Increasing the size M of the input layer results in an increased information rate. In contrast, expanding the dimension n of the final layer reduces the number of bits per second transmitted through the link. Controlling the dimensionality of the final layer in the transmitter can also be used for accommodating an oversampling rate s of the signal, a technique that is often adopted in simulation for improving resolution. This is necessary for capturing the possible spectral broadening effects, which in the optical IM/DD can arise from nonlinearities in the MZM or photodiode functions. If oversampling is considered, the message is effectively mapped onto a symbol of $n \cdot s$ samples. Note that in such a case the simulation sampling rate will accordingly be increased to $s \cdot R_{\text{samp.}}$, leaving

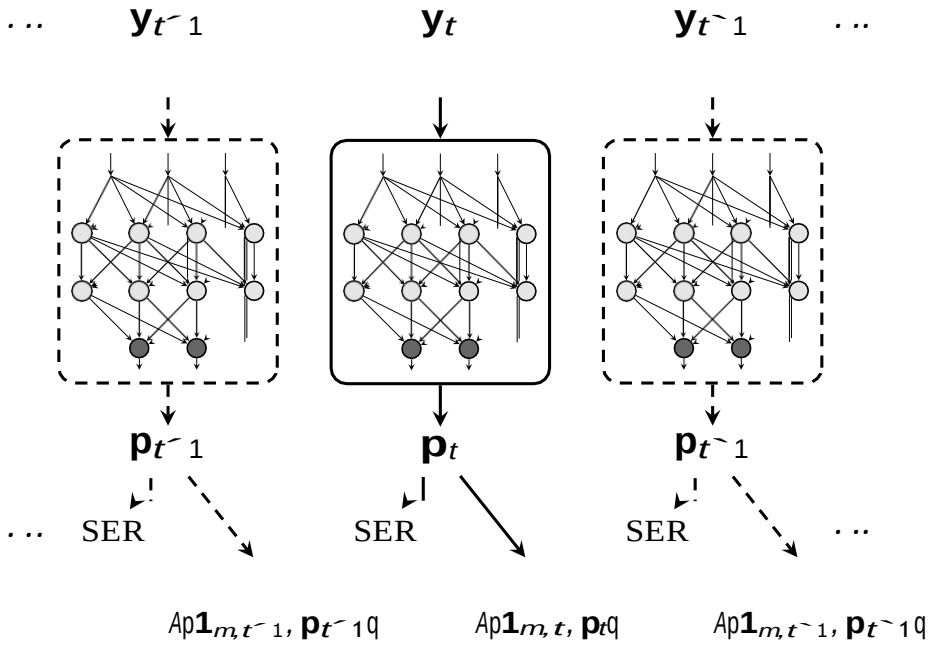

Figure 4.3: Schematic of the FFNN-based receiver section of the optical fiber auto-encoder. The received symbols $(\ldots, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$ are processed independently by the FFNN at each time instance to produce the output probability vectors $(\ldots \mathbf{p}_{t-1}, \mathbf{p}_t, \mathbf{p}_{t+1} \ldots)$, which are used to make a decision on the recovered message as well as to compute the loss of the auto-encoder in the optimization stage.

the information rate of the system unchanged.

4.2.2 Receiver structure

After propagation through the communication channel, the sequence of trans mit symbols $(\ldots \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \ldots)$ is transformed into to the distorted sequence $(\ldots \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$ which is processed by the receiver.

4.2.2.1 Block decoding

The receiver decoding follows the block-by-block procedure employed at the transmitter. The message $\hat{m}_t$ is recovered from the received symbol $\mathbf{y}_t$ without taking into account any of the preceding $\mathbf{y}_{t-k}$ or the succeeding $\mathbf{y}_{t+k}$ symbols. This decoding scheme is depicted in Fig. 4.3. The received block of samples $\mathbf{y}_t$ is transformed to an output probability vector $\mathbf{p}_t$ from which the recovered message is deduced. In the following the details of the receiver FFNN processing as well as the performance metric for message recovery are presented.

4.2.2.2 FFNN processing

Table 4.2 shows the hyper-parameters of the receiver FFNN. The architecture of the layers is identical to those at the transmitter side in a reverse order. The input symbol $\mathbf{y}_t \in \mathbb{R}^{n \cdot s}$ is transformed by the first hidden layer, whose parameters are $\mathbf{W}_4 \in \mathbb{R}^{2M \times n \cdot s}$ and $\mathbf{b}_4 \in \mathbb{R}^{2M}$, utilizing the ReLU activation function. The ReLU

Table 4.2: FFNN definition at the receiver

Layer	Activation	Output
dimension Input		N/A
	n s	
Hidden 1	ReLU	$2M$
Hidden 2	ReLU	$2M$
Final	Softmax	M

is also applied on the next hidden layer, which has the parameters $\mathbf{W}_5 \in \mathbb{R}^{2M\times 2M}$, $\mathbf{b}_5 \in \mathbb{R}^{2M}$. The parameters of the final layer in the receiver FFNN are $\mathbf{W}_6 \in \mathbb{R}^{2M\times M}$ and $\mathbf{b}_6 \in \mathbb{R}^{M}$. The final layer's activation is the *softmax* function (see Sec. 2.1.2) and thus the output is a probability vector $\mathbf{p}_t \in \mathbb{R}^{M}$ with the same dimension as the one-hot vector encoding of the message. Thus, the series of operations

$$\mathbf{y}_t^{'} = \alpha_{\mathrm{ReLU}}\left(\mathbf{W}_4\mathbf{y}_t + \mathbf{b}_4\right), \tag{4.7}$$

$$\mathbf{y}_t^{''} = \alpha_{\mathrm{ReLU}}\left(\mathbf{W}_5\mathbf{y}_t^{'} + \mathbf{b}_5\right), \tag{4.8}$$

$$\mathbf{p}_t = \alpha_{\mathrm{softmax}}\left(\mathbf{W}_6\mathbf{y}_t^{''} + \mathbf{b}_6\right), \tag{4.9}$$

describe the decoding function $\mathbf{p}_t = f_{\text{Dec.-FFNN}}(\mathbf{y}_t)$ of the FFNN receiver. Note that $\mathbf{y}^{J}$ and $\mathbf{y}^{JJ}$ were used to denote the outputs of the first and second hidden layer at

receiver, respectively. For illustrative purposes, only the receiver input $\mathbf{y}_t$ and the corresponding output $\mathbf{p}_t$ are indicated in Fig. 4.3.

4.2.2.3 Performance metrics

The vector of probabilities which is output of the receiver FFNN can be utilized in two ways: Firstly, a decision on the received symbol can be made as

$$\hat{m}_t = \underset{i=1\ldots M}{\operatorname{argmax}}(\mathbf{p}_{t,i}), \tag{4.10}$$

choosing the index $\hat{m}_t$ of the greatest element in $\mathbf{p}_t$ as the recovered message, i.e. choosing the most probable hypothesis. Such a decoding operation applied to the *softmax* output of the neural network directly corresponds to the initial *one-hot* vector encoding of messages at the transmitter input. A symbol error occurs at time

t when $m_t \mathrel{!}= \hat{m}_t$. The symbol decision is counted in the symbol error rate (SER) estimation for the transmitted sequence of N_{seq} messages, which is given by

$$\mathrm{SER}_{\mathrm{seq.}} = \frac{1}{N_{seq.}}\sum_{t=1}^{N_{seq.}} \mathbb{1}\left\{m_t \neq \hat{m}_t\right\}, \tag{4.11}$$

where $\mathbb{1}$ denotes the indicator function, equal to 1 when the condition in the brackets is satisfied, i.e. an error has occurred, and 0 otherwise. During optimization and testing of the system both in simulation and experiment, a data set S consisting of multiple sequences is utilized. As a consequence, the total symbol error rate of the system becomes

$$\mathrm{SER} = \frac{1}{|S|} \sum_{i \in S} \mathrm{SER}_{\mathrm{seq.},i}, \tag{4.12}$$

where $|S|$ denotes the cardinality (the number of elements) in the set S.
The probability vector $\mathbf{p}_t$ is also utilized for computing the loss in the procedure of end-to-end system optimization, described next.

4.3 Optimization strategies

The complete IM/DD optical fiber communication system is implemented as an end-to-end deep FFNN. As explained in the previous sections, such a system is associated with encoding and decoding in a block-by-block manner. This transmission scheme has multiple practical advantages. Firstly, it is computationally simple, making it attractive for low-cost, high-speed implementations. Secondly, it allows massive parallel processing of the single blocks (symbols). Nevertheless, it is important to note that the proposed structure is only able to compensate for the channel-induced interference within a block of $n \cdot s$ receiver samples, as there is no connection between neighboring blocks. The effect of dispersion from neighboring blocks, i.e. the intersymbol interference (ISI), remains uncompensated with such a transceiver design and can thus be viewed as simply introducing extra noise in the encoding and decoding processes. The block size $n \cdot s$ will hence limit the achievable distance with the proposed system. As discussed in Sec. 4.4.4.3 and Chapter 5, increasing the block size and thus accounting for more interference can quickly result in unfeasibly large neural networks. This necessitated the development of transceiver designs tailored for recovering symbol sequences. It allowed to keep the ANN dimensions manageable, while enabling computational structures better suited for the approximation of the encoding and decoding functions in communication systems over dispersive channels. In particular, in Chapters 5 and 6 advanced sequence processing transceiver solutions were proposed using a recurrent neural network structure to encode and decode the blocks or, alternatively, extending the size of the receiver portion of the FFNN to jointly process multiple blocks and thus diminish the influence of dispersion. This resulted in communication systems with improved resilience to ISI, however, at the expense of higher computational complexity.

The optimization of the transmitter and receiver is performed by minimising

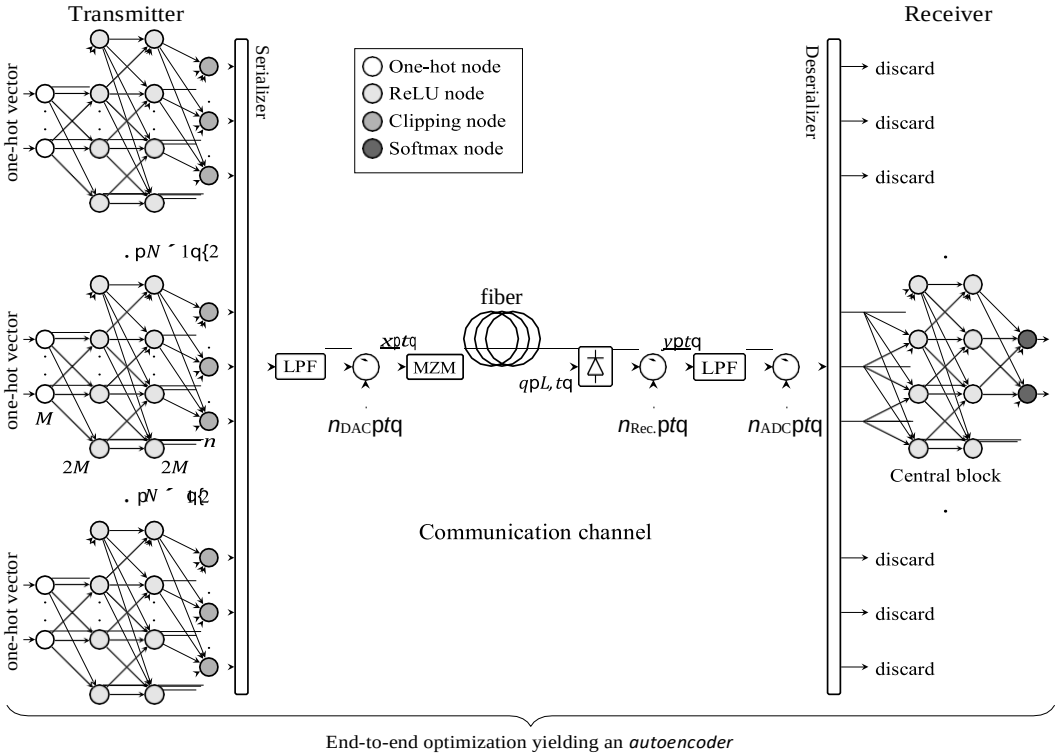

Figure 4.4: Schematic of the IM/DD optical fiber communication system implemented as a deep feedforward artificial neural network. Optimization is performed using the loss between the input messages and the outputs of the receiver, thus enabling end-to-end deep learning of the complete system.

the loss computed at the receiver output between (appropriately chosen) representations of the transmitted m and recovered $\hat{m}$ messages, effectively minimising the end-to-end symbol error rate. Figure 4.4 depicts a schematic of the FFNN-based IM/DD system which shows all components affecting the computation of the endto-end loss at a time instance t, which is obtained as

$$L(\vartheta) = A(\mathbf{1}_{m,t}, f_{\text{AE-FFNN}}(\mathbf{1}_{m,t})) = A(\mathbf{1}_{m,t}, \mathbf{p}_t), \tag{4.13}$$

where

$$\vartheta = \{\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_6, \mathbf{b}_6\}, \tag{4.14}$$

and the function $f_{\text{AE-FFNN}}(\cdot)$ is used to express the complete input-to-output mapping function of the FFNN-based autoencoder, which can be defined as

$$\mathbf{p}_t = f_{\text{FFNN-AE}}(\mathbf{1}_{m,t}) = f_{\text{Dec.-FFNN}}(H_{\text{IMDD}}\{\ldots, f_{\text{Enc.-FFNN}}(\mathbf{1}_{m,t}), \ldots\}), \tag{4.15}$$

with the operator $H_{\text{IMDD}}\{\cdot\}$ describing the channel transformations. Note that the transmitted and recovered messages are conveniently represented by their one-hot

vector encoding $\mathbf{1}_{m,t}$ and output probability vector $\mathbf{p}_t$, respectively.

The dispersion-induced interference accumulates with distance and can span between several consecutive symbols. As a result, multiple transmitted blocks from both preceding and succeeding messages have an impact on the recovery of the message m_t and thus on the computed loss. Such a communication scenario is referred to as a system with *memory*. As already mentioned, for the investigated FFNN design the symbol memory of the channel is treated as extra noise since there is no connection in the processing of neighbouring blocks.

In simulation, multiple blocks need to be concatenated to model a realiztic transmission for m_t. Hence, the output samples of N neighboring blocks (that encode potentially different inputs) are serialised to form a sequence of $N \cdot n \cdot s$ samples for transmission over the channel. Note that N needs to be chosen sufficiently large such that the number of co-propagating messages exceeds the channel memory. At the receiver, the central block is extracted for processing and loss calculation.

The goal is to find a set of trainable parameters from the end-to-end computational graph, i.e. the weight matrices and bias vectors at the transmitter and receiver, that minimise the system loss. The parameters are iteratively updated using stochastic gradient descent (see Sec. 2.2.1.1) and in particular, the Adam algorithm (see Sec. 2.2.1.2). For this reason, a mini-batch of training examples is used on each iteration. Figure 4.5 shows a schematic of the procedure. The mini-batch is formed by $\underline{S}$ sequences, each consisting of N messages. At transmission step i, $i = 1 \ldots \underline{S}$, the messages in the i-th sequence are encoded by the transmitter FFNN and propagated through the channel. At the receiver, the central block is extracted to calculate the loss L_i defined in Eq. (4.13). Obtaining the corresponding losses for each of the central blocks in the sequences from the mini-batch, the loss of the batch can be estimated as the average

$$\underline{\mathscr{L}}(\boldsymbol{\theta}) = \frac{1}{\underline{S}} \sum_{i=1}^{\underline{S}} \mathscr{L}_i. \tag{4.16}$$

Each iteration of the optimization algorithm aims at minimising $\underline{L}(\vartheta)$ computed over a randomly-generated mini-batch. Note that for the purposes of optimizing the system performance, only the central blocks are considered in this training scheme. During propagation through the channel these are subject to the strongest ISI. Moreover, as shown in the following sections, such a scheme enables the generalization of transmitter and receiver parameters over varied distances.

4.3.1 Learning at a fixed nominal distance

In this simple training setting, a nominal fiber length L is set. Every sequence in each of the mini-batches used for training is transmitted through the fixed length of

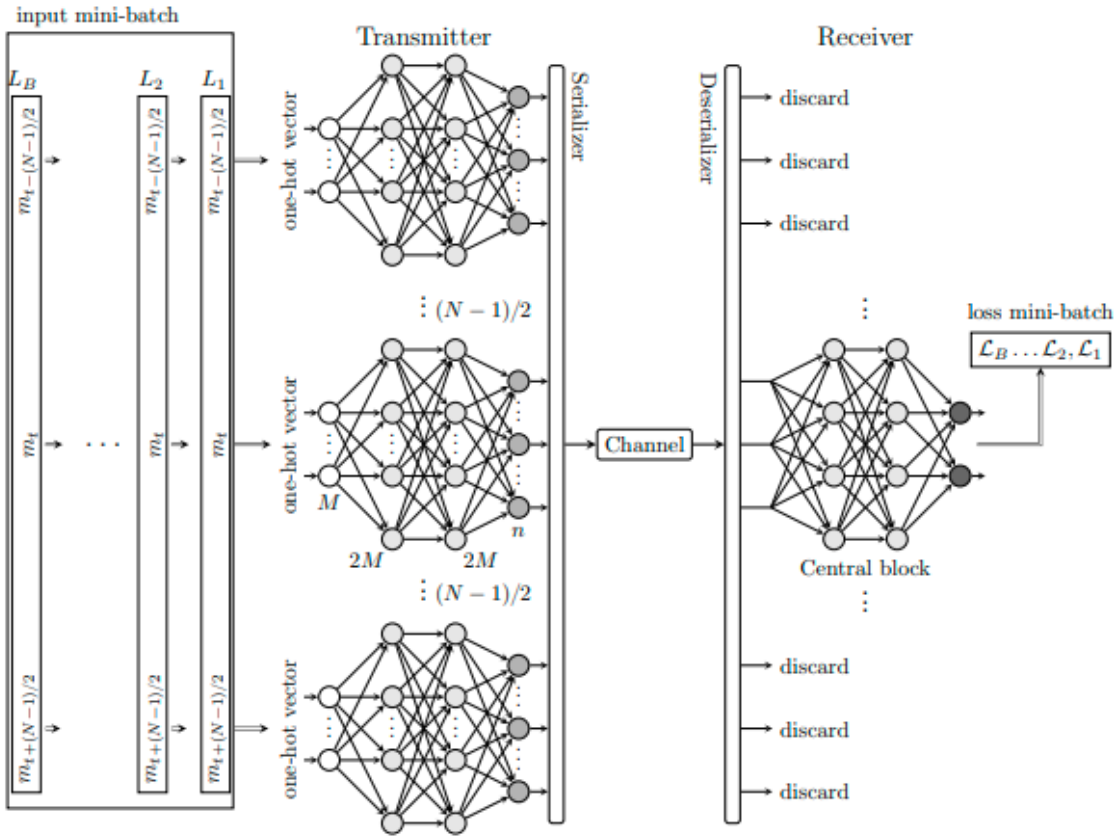

Figure 4.5: Schematic of the training procedure for the FFNN-based auto-encoder, showing how a mini-batch is formed from different transmitted sequences of length N over fiber lengths L_i and the corresponding losses, computed for the central message in every sequence.

fiber, thus accumulating identical amounts of ISI. As a consequence, the transmitter and receiver are optimized exclusively for communication at the fixed nominal distance. Operation at distances, significantly different from L would require reinitiating the training stage.

4.3.2 Multi-distance learning

Since every sequence i from the mini-batch is transmitted independently through the communication channel, it can be associated with a specific distance L_i, which dictates the amount of ISI-induced distortion on the central block. This has a direct impact on the calculated loss L_i, which would increase with distance. As a result, the total batch loss $\underline{L}$ (ϑ) comprises of training examples from multiple transmission scenarios. Performing deep learning aimed at minimising $\underline{L}$ (ϑ), could thus be viewed as optimizing the set of transmitter and receiver parameters ϑ for transmission over all different L_i. A simple scheme would involve assigning to the i-th mini-batch sequence a link length L_i which is randomly drawn from a probability

distribution with appropriately chosen parameters, e.g.

$$L_i \sim N^{(\mu,\sigma)}, \qquad i = 1 \ldots |\underline{S}|, \tag{4.17}$$

where $N^{(\mu,\sigma)}$ denotes the Gaussian distribution, whose mean is μ and standard deviation is σ, and $|\underline{S}|$ is the number of sequences in the mini-batch.

4.4 Numerical investigation of the FFNN autoencoder performance

4.4.1 Bit-to-symbol mapping

As explained throughout the chapter, the auto-encoder communication system is designed such that the end-to-end symbol error rate is minimised. Nevertheless, a common metric examined as an indication of the communication system performance is the bit-error rate (BER). Thus, whenever a symbol is received in error, the number of incorrect bits that have occurred needs to be counted.

The input and output of the end-to-end ANN are non-binary messages $m \in \{1, \ldots, M\}$ and $\hat{m}$. In order to guarantee the low bit error rates in the range 10^{-12} to 10^{-15}, forward error correction (FEC) is required. For complexity reasons, FEC schemes in optical communications are usually binary [101] and often based on hard-decision decoding (HDD), in particular for IM/DD applications. An overview of HDD decoding schemes and their decoding capabilities is given in [102]. To convert between the ANN messages and the FEC encoder/decoder output/input, a *bit labeling* function $\phi: \{1, \ldots, M\} \rightarrow \mathbb{F}_2^B$ is needed. It maps a message m to a binary vector $\mathbf{b} = (b_1, \ldots, b_B)$, $b_i \in \mathbb{F}_2 = \{0, 1\}$ of length $B = \lfloor \log_2(M) \rfloor$. Usually, M and B are selected such that $M = 2^B$.

For computing the BER, presented for the numerical auto-encoder investigation in this chapter, a simple *ad hoc* bit mapping was used. It consisted in assigning the Gray code to the integer input messages $m \in \{1, \ldots, M\}$ [103]. Note that this simple approach is sub-optimal as the deep learning algorithm will only minimise the SER and a symbol error may not necessarily lead to a single bit error. In the presentation of simulation results in this chapter, a lower bound on the achievable BER will hence be provided with an ideal bit mapping by assuming that at most a single bit error occurs during a symbol error. In Chapter 5 an algorithm for the optimization of the bit-to-symbol mapping function was proposed and investigated for the recurrent neural network-based auto-encoder.

4.4.2 Generation of training and testing data

4.4.2.1 Random number generation

It was highlighted in [104] that special care should be taken when applying neural networks in optical communication systems to avoid learning representations of a sequence, e.g. pseudo-random binary sequence (PRBS). The authors of the paper provided guidelines to prevent biasing the performance results, which were meticulously followed when designing both the simulations and transmission experiments part of the investigation described in this thesis. In particular, for the simulations, new random input messages were continuously generated during the optimization stage using a Mersenne twister [105] – a random number generator with a long sequence. In order to obtain the results presented throughout the thesis, the neural network models are trained and stored to perform testing with independent data, generated by a different random number generator. More specifically, the Tausworthe generator (see [106] for details) was used in the test stage. Chapter 6 provides more information on the data generation and collection procedures used in the experimental verification of the developed systems.

4.4.2.2 Optimization stage

The cross entropy loss (see Sec. 2.1.2.2) was used during optimization. The autoencoder was trained on a set of $|S|=25\cdot10^6$ randomly chosen central messages (and $(N-1)\cdot|S|$ random messages of the neighbouring transmit blocks, discarded at the receiver) using the Adam optimizer with a learning rate of 0.001 (default setting). The deep learning library TensorFlow [99] was used to perform the optimization. The mini-batch size was set to $\underline{S}=250$, from where the number of optimization iterations can be accordingly expressed as

$$I = \frac{|S|}{\underline{S}}. \tag{4.18}$$

Thus, $I = 10^5$ iterations of the optimization algorithm were performed in the accommodated setting. The initialisation of the trainable parameters was carried out as recommended in [12, Sec. 8.1.3]: a truncated normal distribution with standard deviation $\sigma=0.1$ was used for the weight matrices $\mathbf{W}$. The bias vectors $\mathbf{b}$ were initialised with $\mathbf{0}$. Validation of the training was performed during the optimization process every 5000 iterations. The validation set of data was independently generated and had the size $|S_v| = 15\cdot10^6$. It is worth noting that in most cases, convergence in the loss and validation symbol error rate of the trained models was obtained after significantly less than 10^5 iterations, which was used as a fixed stopping criterion – a common way to terminate ANN training (see [12, Sec. 8.3]). The

Table 4.3: Simulations parameters assumed for the FFNN-based auto-encoder

Parameter	Value
Transceiver:	
M	64
n	12
N	11
Oversampling (s)	4
$R_{\text{cod.}}$	$\frac{1}{2}$ b/Sa
Channel:	
$R_{\text{samp.}}$	84 GSa/s
Sampling rate ($s \cdot R_{\text{samp.}}$)	336 GSa/s
$R_{\text{inf.}}$	42 Gb/s
LPF bandwidth (B_f)	32 GHz
DAC/ADC ENOB (N_b)	6
Fiber length (L)	varied
Fiber dispersion (D)	17 ps/(nm.km)
Fiber attenuation ($\alpha_{\text{dB/km}}$)	0.2 dB/km
Receiver noise variance ($\sigma_{\text{ref.}}$)	$2.455 \cdot 10^{-4}$

convergence of the results was also confirmed for mini-batch sizes of $\underline{S} = 125$ and 500. More details on how to choose an appropriate amount of mini-batch examples for SGD optimization can be found in [12, Sec. 8.4]. Finally, the validation results were confirmed in the case when the training set was increased to $|S| = 50 \cdot 10^6$ input messages.

4.4.2.3 Testing stage

The trained model, i.e. the set ϑ of optimized weight matrices and bias vectors representing the transmitter and receiver ANNs, was saved and then loaded separately for testing. Testing was performed over a set of different random input messages, generated by a different random number generator (see Sec. 4.4.2.1). In order to obtain a reliable estimation of the BER, the size of the testing set was increased to $|S_t| = 15 \cdot 10^8$. Note that these messages were used to evaluate the performance at a specific distance. They were encoded by the optimized transmitter FFNN, transmitted through the channel and decoded by the optimized receiver FFNN, computing the symbol and bit error rates over the testing set, as described in Sec. 4.2.2.3 and Sec. 4.4.1, respectively. The BER results from the testing phase are shown on the system performance figures presented in this chapter.

4.4.3 Simulation parameters

Table 4.3 lists the simulation parameters for the end-to-end deep-learning-based optical fiber system under investigation. Input messages from a set of $|M| = M = 64$ total messages, each carrying 6 bits, were encoded by the FFNN at the transmitter into symbols of $n \cdot s = 48$ samples at 336 GSa/s in simulation. This rate corresponded to the $R_{\text{samp.}} = 84$ GSa/s sampling rate of the DAC used in experiment multiplied by the assumed simulation oversampling factor of $s = 4$. This resulted in an auto-encoder system with a data rate equal to $R_{\text{inf.}} = 42$ Gb/s. The bandwidth of the signal was restricted at both transmitter and receiver by a brick-wall LPF with a cut-off frequency of $B_f = 32$ GHz to account for the hardware limitations. Note that in practice, down-sampling by a factor of $s = 4$ of the filtered series of symbols can be performed without loss of information. Due to the low-pass filtering, the original series of symbols, each of $n \cdot s = 48$ samples at 336 GSa/s, can be exactly reconstructed from the down-sampled symbol series running at the DAC rate of 84 GSa/s. Down-sampling was performed during the experimental verification of the system reported in Chapter 6. More details on the temporal and spectral representations of the ANN-optimized transmitted signal are provided in Appendix A. For the transmission simulation, the variance of the additive receiver white Gaussian noise was set to the experimentally obtained value of $\sigma^2 = 2.455 \cdot 10^{-4}$, as explained in Sec. 3.3.1. Chapter 6 provides more details on the experiment. The fiber attenuation and dispersion parameters were $\alpha_{\text{dB/km}} = 0.2$ dB/km and $D = 17$ ps/(nm·km), respectively, typical values for a standard single mode fiber [23]. The ENOB of both the DAC and the ADC was set to $N_b = 6$.

4.4.4 System performance

4.4.4.1 Transceivers optimized at a fixed nominal distance

The BER performance of the FFNN auto-encoder optimized at the fixed nominal distance of 40 km was computed over the testing set as described in Sec. 4.4.1 and the results obtained at distances between 37.5 and 42.5 km are shown in Fig. 4.6. The $4.5 \cdot 10^{-3}$ hard-decision forward error correction (HD-FEC) threshold [107] (6.7% overhead) was used as an indicator of the system performance. BER values below the HD-FEC were achieved at distances in the range of 38 – 41.25 km. Moreover, around the nominal training distance of 40 km the BER was below 10^{-4}. For distances longer or shorter than the optimum 40 km, the bit error rate increased. The figure also displays the lower bound, denoted “ideal”, on the achievable BER for each distance. This lower bound was obtained by assuming that a block error gives rise to a single bit error. The results confirmed that utilizing the Gray code of m as a bit-to-symbol mapping function, an approach discussed in 4.4.1 and used

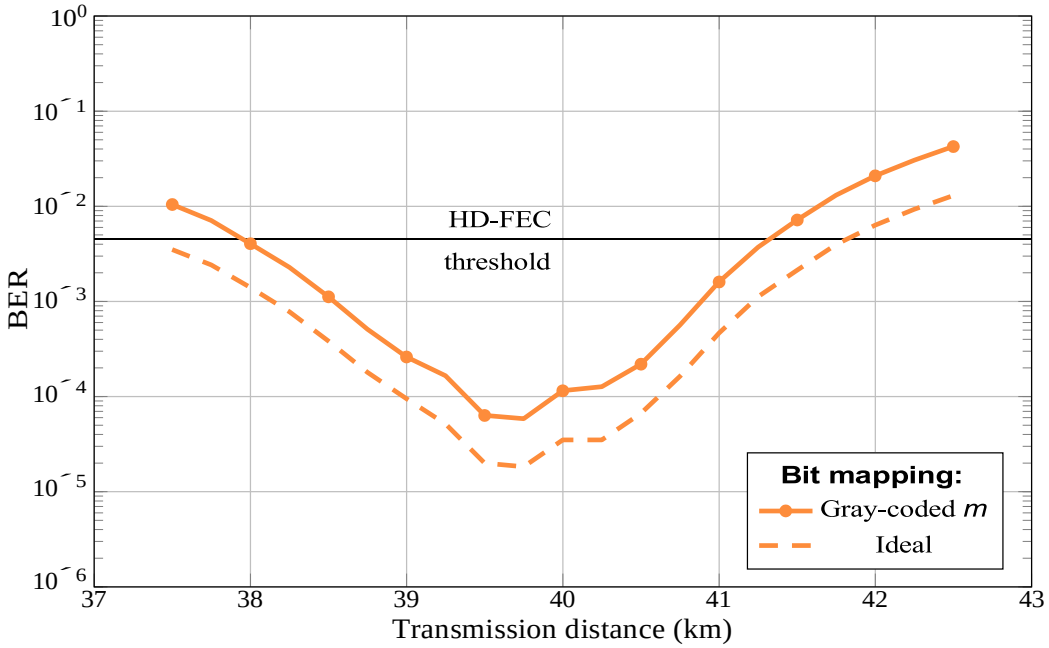

Figure 4.6: BER as a function of transmission distance for a system trained at 40 km. The horizontal dashed line indicates the 6.7% HD-FEC threshold. The BER with an ideal bit mapping, i.e. a symbol error results in a single bit error, is also shown.

in the current investigation, is not optimal for the FFNN auto-encoder. Hence, optimizing the mapping function can lead to additional performance enhancement. This was demonstrated for the developed recurrent neural network-based systems in Chapter 5, where the optimization of the bit labeling function was performed.

The BER performance of systems trained at different distances is shown in Fig. 4.7. For this set of results, seven FFNN-based transceivers were optimized for different distances in the range of 20 to 80 km in steps of 10 km. As before, the optimization was performed at the fixed nominal distance. BERs below the 6.7% hard decision FEC threshold are achieved for all examined distances between 20 and 50 km. Moreover, up to 40 km the BER was below 10^{-4}. Systems trained at distances longer than 50 km achieved BERs above 10^{-2}. The lower bound on the achievable BER was also shown, indicating that the assumed bit-to-symbol mapping was sub-optimal for all of the examined distances. An important observation is that the systems achieved their lowest BER values at the nominal distances at which their training was performed. A rapid increase in their error rate was observed when the distance changed. For example, the BER of auto-encoder optimized at 40 km was degraded from $5 \cdot 10^{-5}$ to above $1 \cdot 10^{-2}$, when the distance was changed from
39.75 km to either 38 or 41.5 km. Such a behaviour is a direct consequence of the implemented training approach which optimizes the system ANN parameters for

operation at a particular distance, without any incentive for robustness to variations. As the amount of dispersion changes with distance, the optimal neural network parameters differ accordingly. Thus, the system BER will increase when the distance is varied without accordingly updating the transceiver parameters.

4.4.4.2 Transceivers optimized for operation at multiple distances

To address the limitations of the training at a fixed nominal distance related to robustness to distance variation, the optimization method proposed in Sec. 4.3.2 is considered next. For such an investigation, the transceiver was trained in a process where the distance for every training message was randomly drawn from a Gaussian distribution with a mean μ and a standard deviation σ . During optimization, this will allow the deep learning algorithm to converge to a set of generalizedANN
parameters, robust to certain variation of the dispersion. Figure 4.8 shows the test BER performance of the system trained at a mean distance μ = 40 km and different values of the standard deviation. It can be seen that for the cases of σ = 4 km and σ = 10 km this training method allowed BER values below the HD-FEC threshold in wider ranges of transmission distances than for σ = 0 km, the scenario covered in the previous investigation. For instance, when σ = 4 km, BER values below the
$4.5 \cdot 10^{-3}$ threshold were achievable between 30.25 km and 49.5 km, yielding an

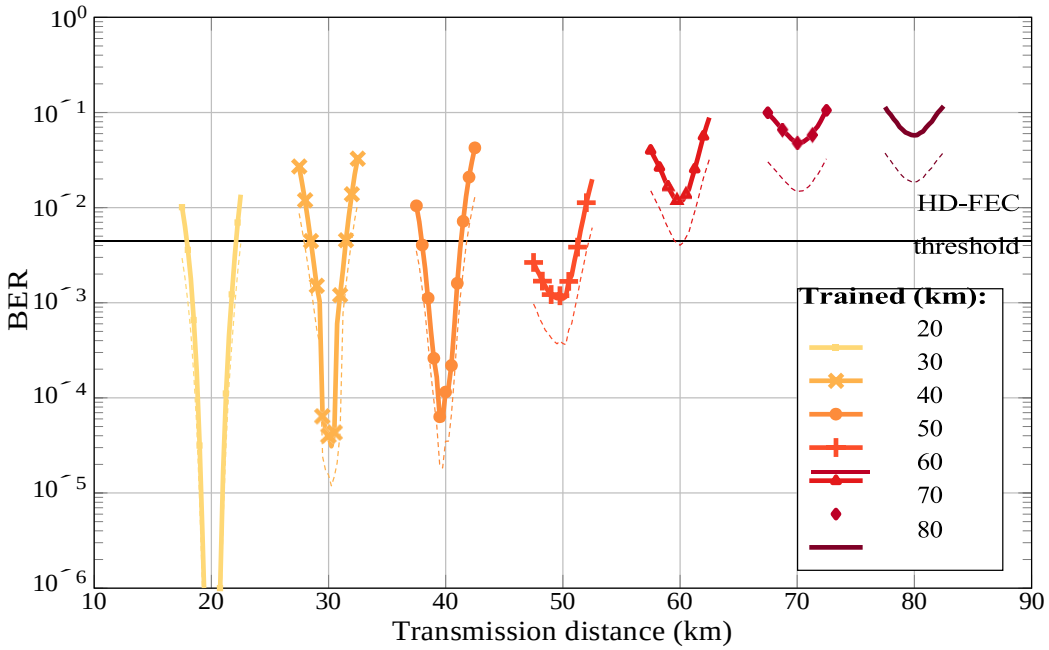

Figure 4.7: BER as a function of transmission distance for systems trained at a fixed nominal distance of (20+i 10) km, with i 0, . . . , 6 . The 6.7% HD-FEC threshold is indicated by a horizontal dashed line. Thin dashed lines below the curves give a lower bound on the achievable BER when an ideal bit mapping is assumed.

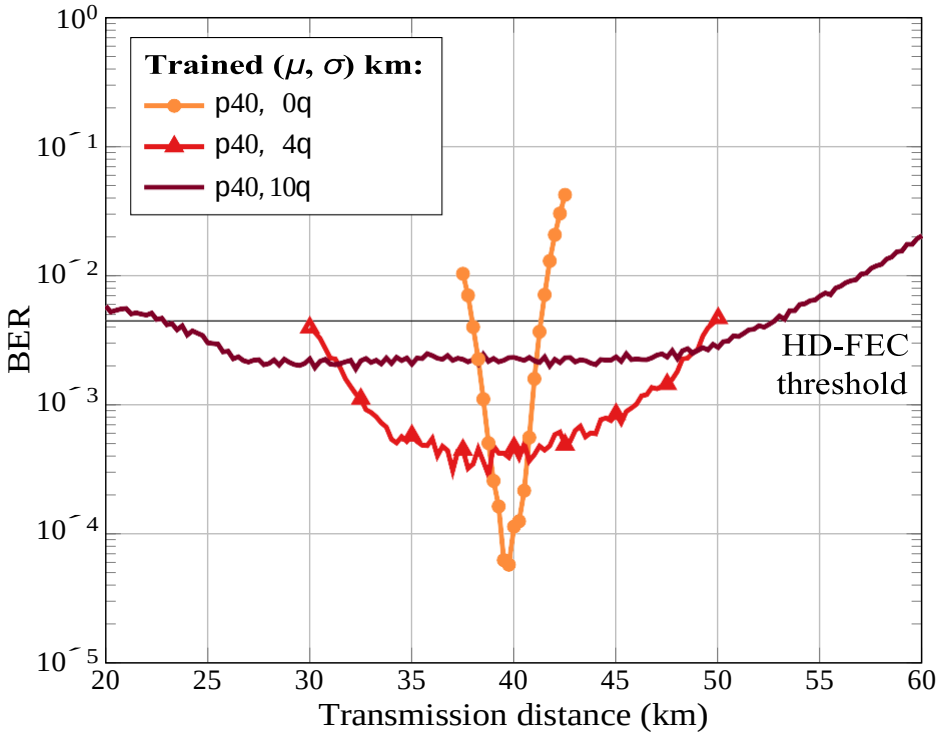

Figure 4.8: Bit error rate as a function of transmission distance for systems where the training is performed at normally distributed distances with mean μ = 40 km and standard deviation σ. The horizontal dashed line indicates the 6.7% HD-FEC threshold.

operating range of 19.25 km. The distance tolerance was further increased when σ = 10 km was used for the optimization. It is worth noting that in this case the obtained BERs were higher, due to the increased difficulty in converging to parameters valid for the significantly varied distances with the standard deviation of 10 km. Nevertheless, performance below the HD-FEC threshold was achieved by such a transceiver for a distance range of 27.75 km, spanning from 24 km up to 51.75 km. The lowest BER value of the system was significantly increased compared to the auto-encoder trained at the fixed nominal distance of 40 km, i.e. $2 \cdot 10^{-3}$ compared to $5 \cdot 10^{-4}$. Thus, there exists a trade-off between system robustness and performance. The multi-distance learning method can be hugely beneficial for practical applications as it introduces both robustness and flexibility of the system to variations in the link distance, without requiring any reconfiguration of the transceiver.

To further investigate the levels of flexibility that such an optimization technique can offer, the BER performance of the *distance-agnostic* auto-encoder was examined at different values of the mean training distance μ. Figure 4.9 shows the BER performance of systems trained at the mean distances μ of 20, 40, 60 and

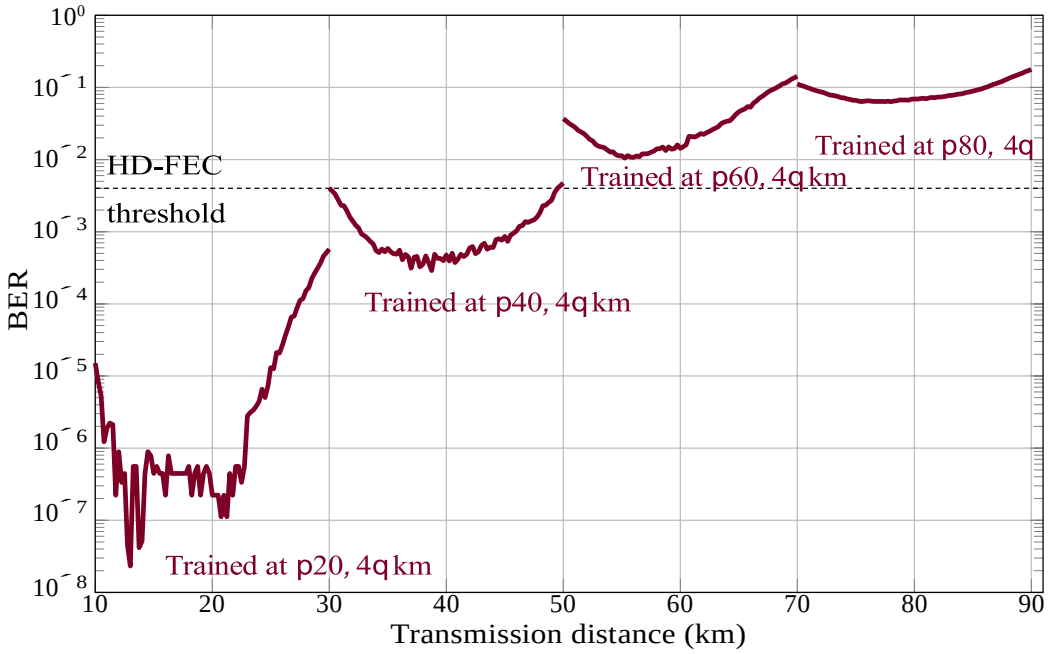

Figure 4.9: Bit error rate as a function of transmission distance for systems where the training is performed at normally distributed distances around the mean values μ of 20, 40, 60, 80 km and a fixed standard deviation σ = 4 km. The horizontal dashed line indicates the 6.7% HD-FEC threshold.

80 km with a fixed standard deviation of σ = 4 km. It can be seen that generalization of the transceiver parameters was achieved for all setups. An interesting implication is that only two sets of optimized transceiver parameters, i.e. trained at (20,4) km and (40,4) km, are required to achieve BER values below the HD-FEC threshold at all distances up to 50 km. In contrast, the system performances shown in Fig. 4.7, indicate that at least 8 different transceivers would need to be used to cover similar distances.

4.4.4.3 Transceivers with different coding rates

The investigation of the end-to-end deep learning-based optical fiber system reported in the previous sections covered the scenario where an input message carrying 6 bits of information (M = 64) was encoded into a band-limited symbol of 48 samples ($n \cdot s$ = 48 with an oversampling factor of s = 4) at 336 GSa/s. Thus, the result was an auto-encoder operating at the information rate of 42 Gb/s. In the following, the implementation of systems with different rates is demonstrated. This is achieved by varying the input and the output dimension of the ANNs, i.e. changing the hyper-parameters M and n. For this investigation, the oversampling factor and the sampling rate of the simulation are kept fixed and equal to 4 and 336 GSa/s, respectively. In Figure 4.10 solid lines show the BER performance of the system at

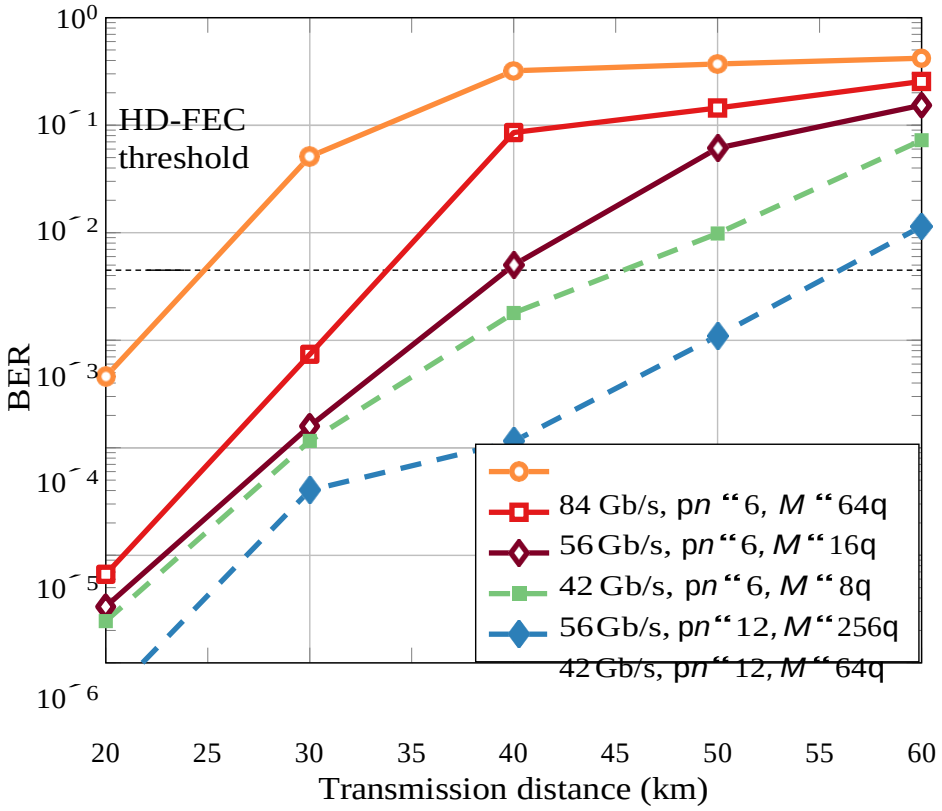

Figure 4.10: Bit error rate as a function of transmission distance for systems with different information rates. The training is performed at a fixed nominal distance.

different rates when the number of digital waveform samples into which the input message is encoded was decreased, in particular n = 6 was used. In such a way bit rates of 42 Gb/s, 56 Gb/s and 84 Gb/s were achieved for M = 8 ($R_{\text{cod.}}$ = 1/2), M = 16 ($R_{\text{cod.}}$ = 2/3), and M = 64 ($R_{\text{cod.}}$ = 1), respectively (see Sec. 4.2.1.3). The BER for the 84 Gb/s system rapidly increased as a function of distance and error rates below the HD-FEC could be achieved only up to 20 km. On the other hand, 42 Gb/s and 56 Gb/s were transmitted reliably with such a hyper-parameter setting at 30 km. An alternative approach to decreasing the transmitted samples in a block, is to increase the information rate of the system by considering input messages with a greater information content. Dashed lines in Fig. 4.10 show the cases of M = 64, n = 12 ($R_{\text{cod.}}$ = 1/2) and M = 256, n = 12 ($R_{\text{cod.}}$ = 2/3), corresponding to information rates $R_{\text{inf.}}$ of 42 Gb/s and 56 Gb/s, respectively. In comparison to the cases where n = 6, such systems exhibited an extended operational reach below the BER threshold, due to the larger block size leading to a reduced influence of the chromatic dispersion. For example, the 56 Gb/s system could achieve BER value below the HD-FEC at 40 km, while for 42 Gb/s, this distance was 50 km. Thus, increasing the auto-encoder information rate by implementing a larger M instead of a smaller n enabled additional reach of around 10 km at 56 Gb/s and around 20 km at 42 Gb/s.

However, a drawback of such a solution is the larger FFNN size since there is an exponential dependence between M and the coding rate, i.e. $M = 2^{n \cdot R_{\text{cod.}}}$, thus increasing the computational and memory demands as well as the training times of the auto-encoder. The investigation shown in Fig. 4.10 highlights that the optical fiber auto-encoder concept can be easily applied for designing systems with different information rates and gives an insight on the possible implementation approaches and the trade-offs between performance and complexity associated with their implementation.

4.5 Summary

In this chapter, it was described how the optical fiber communication system can be implemented as an end-to-end computational graph to perform system optimization in a single process. In particular, the optical fiber transceiver was designed as a simple feedforward neural network and deep learning was applied to find transmitter and receiver ANN parameters that minimise the symbol error rate in the system. The benefits of this method were illustrated by applying it to intensity modulation/direct detection (IM/DD) systems, showing that bit error rates below the 6.7% hard-decision forward error correction threshold can be achieved at distances beyond 50 km – links of practical interest for short reach optical fiber communications. A multi-distance transceiver optimization method was also proposed in the chapter. It was verified in simulation that the method yields robust and flexible transceivers that can enable — without reconfiguration — reliable transmission over a large range of link distances. The research was published in [62]. To complement the simulation results described in this chapter, an experimental verification of the proposed FFNN-based auto-encoder system as well as the multi-distance learning method was conducted and described in Chapter 6. Before that, the investigation in Chapter 5 addresses the limitations of the FFNN design related to compensating for the intersymbol interference at longer transmission distances.

Chapter 5

Recurrent neural network-based optical fiber transceiver

Chapter 3 introduced the transformations and distortions caused by the optical IM/DD communication channel on an input sequence of digital samples. In particular, it was described how chromatic dispersion leads to interference between preceding and succeeding samples, thus imposing severe limitations on the achievable system reach.

To optimize the transceiver for communication over such a channel, a feedforward artificial neural network (FFNN)-based auto-encoder was investigated in Chapter 4. As explained in Sec. 4.2, this system employed a block-based transmission strategy using a transmitter which encodes input messages m from a finite alphabet with size $|M| = M$, each carrying $\log_2(M)$ bits, into blocks (symbols) of n digital waveform samples. At the receiver, the distorted n-dimensional symbols are decoded block-by-block, recovering the transmitted messages. The described system encodes/decodes each symbol *independently*, i.e. without information from the previous or future encoding/decoding. However, as explained in Sec. 3.2, the dispersion-induced interference is an effect which accumulates linearly with distance and quadratically with the signal bandwidth, quickly extending across multiple adjacent symbols. The presence of intersymbol interference (ISI) renders the optical IM/DD link a communication channel with memory.

The FFNN auto-encoder system is inherently unable to compensate for ISI, i.e. interference outside of the symbol block, which is treated as extra noise. As a consequence, the achievable performance, in terms of chromatic dispersion that can be compensated and hence transmission distance, of such a system is limited by the block size. The presented results in Sec. 4.4.4.3 investigated the solution of expanding the FFNN by including more samples within a symbol block, i.e. implementing larger n. Nevertheless, in addition to increasing the output FFNN dimension n, for

a given coding rate $R_{\text{cod.}}$ (defined in Sec. 4.2.1.3), following this approach would require an exponential increase of the input neural network dimension M, which is related to n as $M = 2^{n \cdot R_{\text{cod.}}}$. The result is an auto-encoder system that can quickly become computationally infeasible to implement due to the rapidly increasing amount of transceiver ANN parameters as a function of compensated interference.

In this chapter, the limitations of the FFNN-based auto-encoder design for communication over channels with memory were addressed by developing a deep learning-based transceiver for *sequence processing* using a bidirectional recurrent neural network (BRNN). The BRNN-based auto-encoder allows to utilize information from both preand post-cursor symbols in the encoding and decoding processes. The optimized system was combined with an efficient sliding window sequence estimation algorithm, yielding a sliding window bidirectional recurrent neural network (SBRNN) auto-encoder. This allowed to control the processing memory in the BRNN receiver via the estimation window W, external to the end-to-end neural network architecture. As a result, the resilience to ISI could be enhanced, while keeping the number of ANN parameters at the transceiver fixed.

In addition to introducing the SBRNN auto-encoder design, this chapter includes an extensive numerical investigation of the system performance. In particular, Sec. 5.3.1 and 5.3.3 describe the study of the SBRNN performance and complexity in comparison to classical pulse amplitude modulation (PAM) transmission with state-of-the-art nonlinear equalisation or maximum likelihood sequence detection receivers, respectively.

5.1 End-to-end system design

The complete optical fiber communication system is implemented as an end-to-end computational graph using ANN-based transmitter and receiver, a concept which was introduced in Chapter 4. The transceiver design described here uses a bidirectional recurrent neural network to handle the intersymbol interference effects stemming from preceding and succeeding symbols, induced by the dispersive optical channel. Figure 5.1 shows a schematic of the proposed BRNN-based communication system. This section describes the encoding and decoding performed by the scheme.

5.1.1 Transmitter

5.1.1.1 Bidirectional recurrent encoding

Within the optical fiber auto-encoder framework, described in Sec. 4.1, the function of the artificial neural network at the transmitter is to encode the stream of input messages $(\ldots, m_{t-1}, m_t, m_{t+1}, \ldots)$, $m_t \in \{1, \ldots, M\}$, each of which drawn indepen-

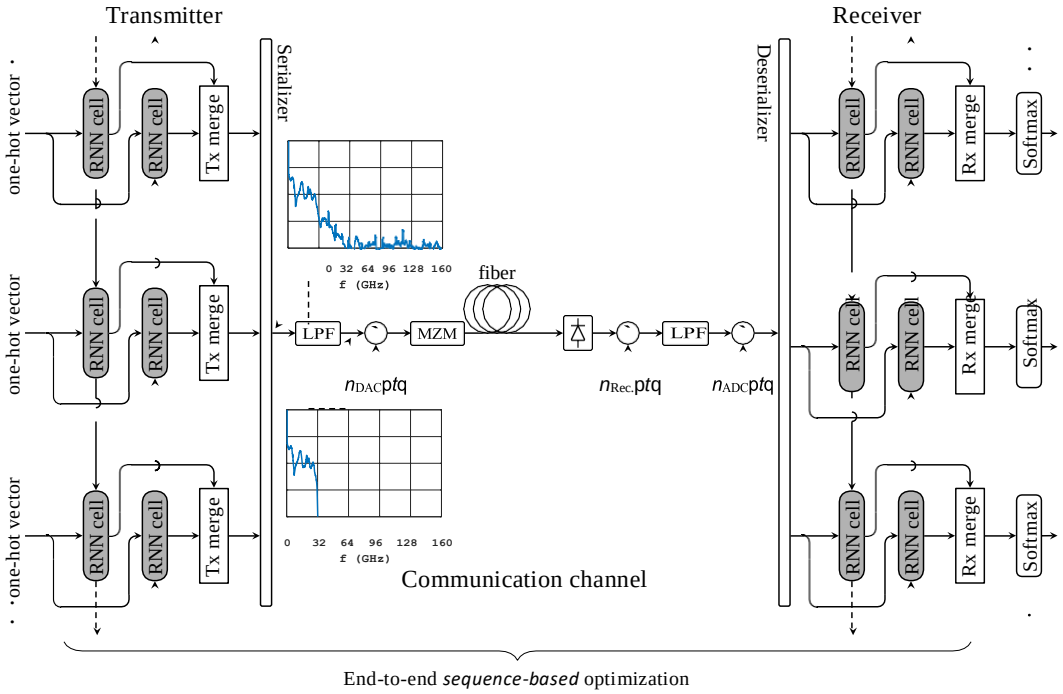

Figure 5.1: Schematic of the IM/DD optical fiber communication system implemented as a bidirectional deep recurrent neural network. optimization is performed between the stream of input messages and the outputs of the receiver, thus enabling endto-end optimization via deep learning of the complete system. Inset figures show an example of the transmitted signal spectrum both at the output of the neural network and before the DAC.

dently from an alphabet M of $|M| = M$ total messages, into a sequence of transmit blocks of digital waveform samples (symbols) $(\ldots, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1}, \ldots)$ ready for transmission over the communication channel. Unlike the FFNN system proposed in Sec. 4.2, the BRNN transmitter structure, proposed and investigated in this chapter, allows to utilize information from both preand post-cursor symbols, i.e. $\mathbf{x}_{t-k}$ or $\mathbf{x}_{t+k}$, into the encoding of the message m_t at time t. Thus the BRNN transmitter performs encoding of input *sequences*, which is necessary for communication over communication channels with ISI.

5.1.1.2 BRNN processing

Figure 5.2 shows a schematic of the BRNN encoding procedure. First, the messages m_t are represented as one-hot vectors $\mathbf{1}_{m,t} \in \mathrm{R}^M$, which are fed into the recurrent structure for bidirectional encoding. Two variations of the recurrent cell in the RNN structure were investigated – a straightforward vanilla concatenation as well as a long short-term memory gated recurrent unit structure (LSTM-GRU). The vanilla and LSTM-GRU processing cells were introduced in Sec. 2.1.3.1. In the following, the message encoding performed by each of these cells is described.

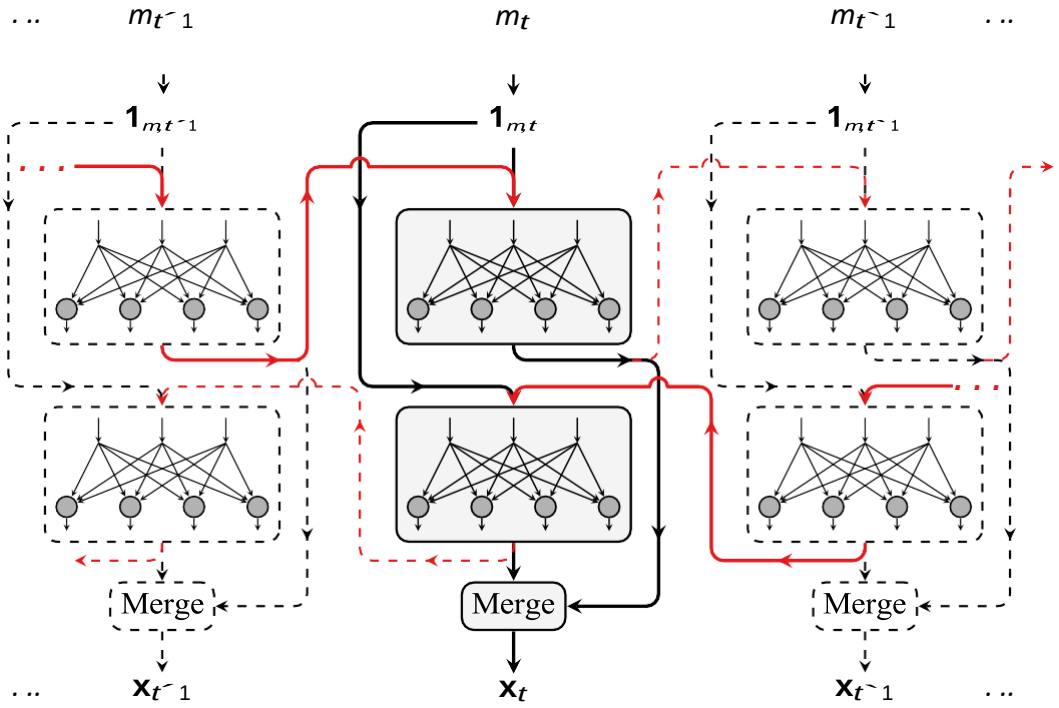

Figure 5.2: Schematic of the BRNN-based transmitter section of the optical fiber autoencoder. The input messages (. . . m_{t-1}, m_t, m_{t+1} . . .), represented as one-hot vectors (…, $\mathbf{1}_{m,t-1}$, $\mathbf{1}_{m,t}$, $\mathbf{1}_{m,t+1}$…), are processed bidirectionally at each time instance by the neural network to produce the sequence of encoded symbols (blocks of samples) (…$\mathbf{x}_{t-1}$, $\mathbf{x}_t$, $\mathbf{x}_{t+1}$…). Thick solid lines are used to highlight the the connections to symbols that have an impact on the processing of m_t. Note that a *vanilla* BRNN cell structure is adopted for illustrative purposes.

Table 5.1: Vanilla and LSTM-GRU cell definitions at the transmitter (single direction)

	Layer	Activation	Output dimension
Vanilla:			
	Input	one-hot enc. & concat.	$M+n\cdot s$
	Final	Clipping	$n\cdot s$
LSTM-GRU:			$n\cdot s$
	Gate b	Sigmoid	$n\cdot s$
	Final	Clipping	$n\cdot s$

Table 5.1 lists the neural network hyper-parameters used for the investigation presented in this chapter. The recurrent processing in the backward and forward directions is performed using identical network hyper-parameters and architectures. For the vanilla BRNN, in the forward direction the input $\mathbf{1}_{m,t}$ at time t is concatenated.

The message encoding performed by the alternative LSTM-GRU variant of the BRNN transmitter is described next. As explained in Sec. 2.1.3.1, compared to the vanilla variant, it consists of two additional memory gates. Their function is to decide which elements of the current input/previous output concatenation should

be preserved for further processing. Each of the gates in the proposed design is modeled by a single layer of trainable parameters and the sigmoid activation. The encoding function $\mathbf{x}_t = f_{\text{Enc.-BRNN}}(\ldots \mathbf{1}_{m,t-1}, \mathbf{1}_{m,t}, \mathbf{1}_{m,t+1} \ldots)$ for the LSTM-GRU BRNN transmitter is the series of operations

$$\overrightarrow{\mathbf{g}}_t^a = \alpha_{\text{sigmoid}}\left(\mathbf{W}_{\text{fw}}^a \left(\mathbf{1}_{m,t}^T \quad \overrightarrow{\mathbf{x}}_{t-1}^T\right)^T + \mathbf{b}_{\text{fw}}^a\right), \tag{5.4}$$

$$\overrightarrow{\mathbf{g}}_t^b = \alpha_{\text{sigmoid}}\left(\mathbf{W}_{\text{fw}}^b \left(\mathbf{1}_{m,t}^T \quad \overrightarrow{\mathbf{x}}_{t-1}^T\right)^T + \mathbf{b}_{\text{fw}}^b\right), \tag{5.5}$$

$$\overrightarrow{\mathbf{x}}_t = (1 - \overrightarrow{\mathbf{g}}_t^b) \odot \overrightarrow{\mathbf{x}}_{t-1} + \overrightarrow{\mathbf{g}}_t^b \odot \alpha_{\text{Clipping}}\left(\mathbf{W}_{\text{fw}} \left(\mathbf{1}_{m,t}^T \quad (\overrightarrow{\mathbf{g}}_t^a \odot \overrightarrow{\mathbf{x}}_{t-1})^T\right)^T + \mathbf{b}_{\text{fw}}\right), \tag{5.6}$$

$$\overleftarrow{\mathbf{g}}_t^a = \alpha_{\text{sigmoid}}\left(\mathbf{W}_{\text{bw}}^a \left(\mathbf{1}_{m,t}^T \quad \overleftarrow{\mathbf{x}}_{t+1}^T\right)^T + \mathbf{b}_{\text{bw}}^a\right), \tag{5.7}$$

$$\overleftarrow{\mathbf{g}}_t^b = \alpha_{\text{sigmoid}}\left(\mathbf{W}_{\text{bw}}^b \left(\mathbf{1}_{m,t}^T \quad \overleftarrow{\mathbf{x}}_{t+1}^T\right)^T + \mathbf{b}_{\text{bw}}^b\right), \tag{5.8}$$

$$\overleftarrow{\mathbf{x}}_t = (1 - \overleftarrow{\mathbf{g}}_t^b) \odot \overleftarrow{\mathbf{x}}_{t+1} + \overleftarrow{\mathbf{g}}_t^b \odot \alpha_{\text{Clipping}}\left(\mathbf{W}_{\text{bw}} \left(\mathbf{1}_{m,t}^T \quad (\overleftarrow{\mathbf{g}}_t^a \odot \overleftarrow{\mathbf{x}}_{t+1})^T\right)^T + \mathbf{b}_{\text{bw}}\right), \tag{5.9}$$

$$\mathbf{x}_t = \frac{1}{2}\left(\overrightarrow{\mathbf{x}}_t + \overleftarrow{\mathbf{x}}_t\right), \tag{5.10}$$

where $\odot$ denotes element-wise multiplication of vectors and $\mathbf{W}_{\text{fw}} \in \mathbb{R}^{n\cdot s\times(M+n\cdot s)}$, $\mathbf{b}_{\text{fw}} \in \mathbb{R}^{n\cdot s}$, $\mathbf{W}_{\text{fw}}^a \in \mathbb{R}^{n\cdot s\times(M+n\cdot s)}$, $\mathbf{b}^a \in \mathbb{R}^{n\cdot s}$ and $\mathbf{W}^b \in \mathbb{R}^{n\cdot s\times(M+n\cdot s)}$, $\mathbf{b}^b \in \mathbb{R}^{n\cdot s}$ (forward direction), and $\mathbf{W}_{\text{bw}} \in \mathbb{R}^{n\cdot s\times(M+n\cdot s)}$, $\mathbf{b}_{\text{bw}} \in \mathbb{R}^{n\cdot s}$, $\mathbf{W}^a_{\text{bw}} \in \mathbb{R}^{n\cdot s\times(M+n\cdot s)}$, $\mathbf{b}_{\text{bw}}^a \in \mathbb{R}^{n\cdot s}$ and $\mathbf{W}^b_{\text{bw}} \in \mathbb{R}^{n\cdot s\times(M+n\cdot s)}$, $\mathbf{b}^b_{\text{bw}} \in \mathbb{R}^{n\cdot s}$ (backward direction) are all trainable weight matrices and bias vectors in the LSTM-GRU BRNN transmitter.

Note that the BRNN transmitter encodes the *full* sequence of input messages. Thus, the forward/backward recurrent processing at the transmitter introduces extra latency of the order of a full data sequence. By termination of the sequences, this latency can be limited to a practically manageable level (which can be in the range of thousands of symbols). Moreover, as described later in Sec. 5.1.4, the auto-encoder is operated in an efficient sliding window algorithm for sequence estimation, significantly reducing the end-to-end processing delay. The sequence lengths and sliding window size used in the numerical investigation of the BRNN system are specified in Sec. 5.3. For the experimental verification of the system, which is described in Chapter 6, these parameters are specified in Sec. 6.5.

5.1.2 Receiver

The function of the ANN receiver is to decode the sequence of received symbols (. . . $\mathbf{y}_{t-1}$, $\mathbf{y}_t$, $\mathbf{y}_{t+1}$ . . .) obtained at the output of the channel, producing a sequence of recovered messages $(\ldots \hat{m}_{t-1}, \hat{m}_t, \hat{m}_{t+1} \ldots)$. The design based on a bidirectional recurrent neural network allows to perform the operation by utilizing information from previous and future decoding.

5.1.2.1 Recurrent decoding

The BRNN decoding scheme is shown schematically in Fig. 5.3. The BRNN-based receiver processing follows an identical recurrent procedure to the one used at the transmitter. The message $\hat{m}_t$ is recovered from the received symbol $\mathbf{y}_t$ via decoding that involves both preceding $\mathbf{y}_{t-k}$ and succeeding $\mathbf{y}_{t+k}$ symbols. In particular, the received block of samples $\mathbf{y}_t$ is processed to an output probability vector $\mathbf{p}_t$ from which the per-example loss is calculated during optimization and also a decision for the recovered message $\hat{m}_t$ is made. The specific receiver BRNN processing, adopted for the proposed auto-encoder design is presented next.

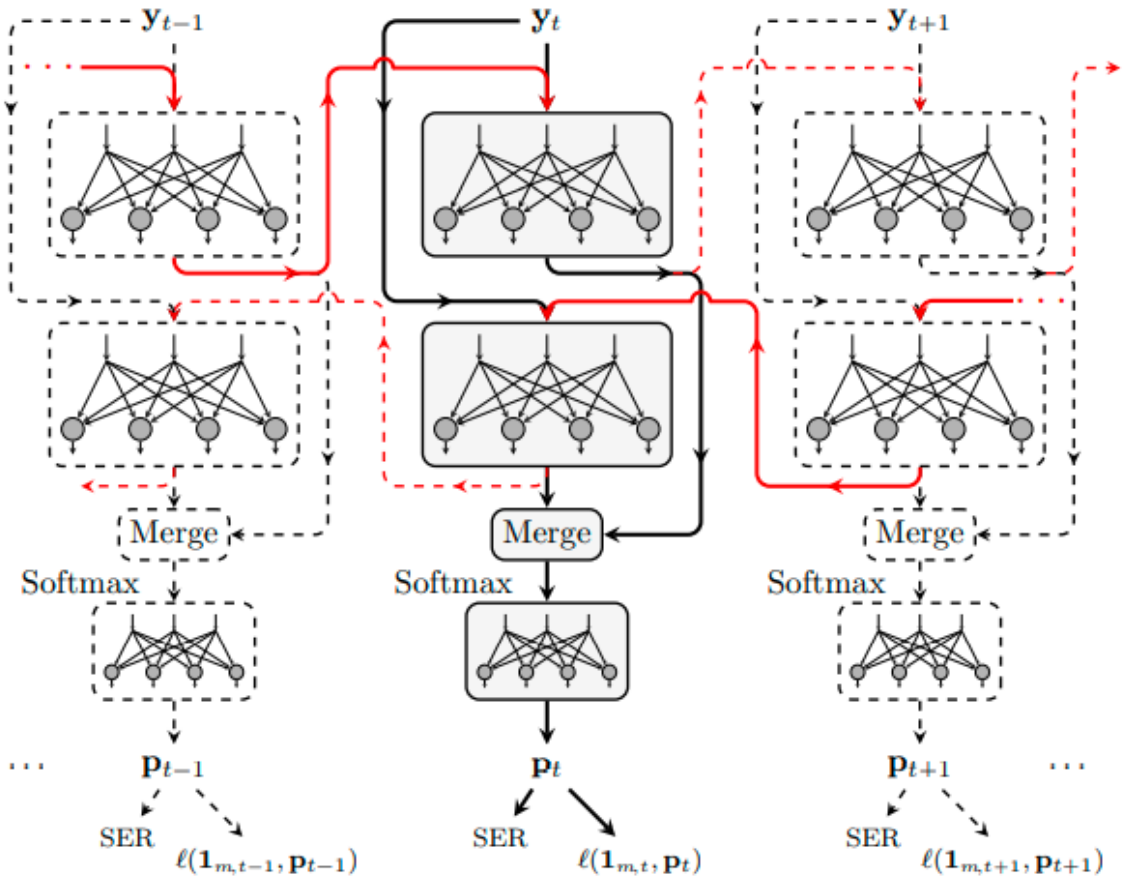

Figure 5.3: Schematic of the BRNN-based receiver section of the proposed optical fi bre auto-encoder. The received symbols $(\ldots, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$ are processed bidirectionally by the BRNN to produce the output probability vectors $(\ldots \mathbf{p}_{t-1}, \mathbf{p}_t, \mathbf{p}_{t+1} \ldots)$. These are utilized in two ways: to make a decision on the recovered message as well as to compute the loss of the auto-encoder in the optimization stage.

5.1.2.2 BRNN processing

Table 5.2 lists the hyper-parameters for the neuron layers used at the BRNN receiver. For decoding using the vanilla BRNN, the current input $\mathbf{y}_t$ is concatenated with the previous output in the forward direction $\overrightarrow{\mathbf{h}}_{t-1}$ and processed by the cell with a ReLU activation function, producing $\overrightarrow{\mathbf{h}}_t \in \mathrm{R}^{2M}$. Identically in the backward direction $\mathbf{y}_t$ is concatenated with $\overleftarrow{\mathbf{h}}_{t+1}$ and transformed to $\overleftarrow{\mathbf{h}}_t \in \mathrm{R}^{2M}$. The two current outputs $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$ are concatenated in the *Merge* module, yielding $\mathbf{h}_t \in \mathrm{R}^{4M}$.

Subsequently, the output $\mathbf{h}_t$ of the cell is applied to a *softmax* layer to obtain the output probability vector $\mathbf{p}_t \in \mathrm{R}^{M}$, having the dimensionality of the input one-hot vector, and utilized for loss computation and message recovery. The decoding function $\mathbf{p}_t = f_{\text{Dec.-BRNN}}(\ldots \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$ of the vanilla BRNN receiver is, hence,

Table 5.2: Vanilla and LSTM-GRU cell definitions at the receiver (single direction)

	Layer	Activation	Output dimension
Vanilla:			
	Input	concat.	$n \cdot s + 2M$
	Hidden	ReLU	$2M$
	Final	Merge & Softmax	M
LSTM-GRU:			
	Input	concat.	$n \cdot s + 2M$
	Gate a	Sigmoid	$2M$
	Gate b	Sigmoid	$2M$
	Hidden	ReLU	$2M$
	Final	Merge & Softmax	M

described by the series of transformations

$$\overrightarrow{\mathbf{h}}_t = \alpha_{\text{ReLU}}\left(\mathbf{W}^r_{\text{fw}}\left(\mathbf{y}_t^T \quad \overrightarrow{\mathbf{h}}_{t-1}^T\right)^T + \mathbf{b}^r_{\text{fw}}\right), \tag{5.11}$$

$$\overleftarrow{\mathbf{h}}_t = \alpha_{\text{ReLU}}\left(\mathbf{W}^r_{\text{bw}}\left(\mathbf{y}_t^T \quad \overleftarrow{\mathbf{h}}_{t+1}^T\right)^T + \mathbf{b}^r_{\text{bw}}\right), \tag{5.12}$$

$$\mathbf{h}_t = \left(\overrightarrow{\mathbf{h}}_t^T \quad \overleftarrow{\mathbf{h}}_t^T\right)^T, \tag{5.13}$$

$$\mathbf{p}_t = \alpha_{\text{softmax}}\left(\mathbf{W}_{\text{soft.}}\mathbf{h}_t + \mathbf{b}_{\text{soft.}}\right), \tag{5.14}$$

$\mathbf{b}^r_{\text{bw}} \in \mathbb{R}^{2M}$ (backward) are the weight matrices and bias vectors in the bidirectional vanilla cell at the receiver, while $\mathbf{W}_{\text{soft.}} \in \mathbb{R}^{4M\times M}$ and $\mathbf{b}_{\text{soft.}} \in \mathbb{R}^{M}$ are the weight matrix and bias vector of the final softmax layer. These constitute the set of trainable parameters in the vanilla BRNN receiver.

Alternatively, for the BRNN receiver designed as an LSTM-GRU, the decoding $\mathbf{p}_t = f_{\text{Dec.-BRNN}}(\ldots \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots)$ is given by the series of operations

$$\overrightarrow{\mathbf{g}}_t^{r,a} = \alpha_{\text{sigmoid}}\left(\mathbf{W}^{r,a}_{\text{fw}}\left(\mathbf{y}_t^T \quad \overrightarrow{\mathbf{h}}_{t-1}^T\right)^T + \mathbf{b}^{r,a}_{\text{fw}}\right), \tag{5.15}$$

$$\overrightarrow{\mathbf{g}}_t^{r,b} = \alpha_{\text{sigmoid}}\left(\mathbf{W}^{r,b}_{\text{fw}}\left(\mathbf{y}_t^T \quad \overrightarrow{\mathbf{h}}_{t-1}^T\right)^T + \mathbf{b}^{r,b}_{\text{fw}}\right), \tag{5.16}$$

$$\overrightarrow{\mathbf{h}}_t = (1 - \overrightarrow{\mathbf{g}}_t^{r,b}) \odot \overrightarrow{\mathbf{h}}_{t-1} + \overrightarrow{\mathbf{g}}_t^{r,b} \odot \alpha_{\text{ReLU}}\left(\mathbf{W}^r_{\text{fw}}\left(\mathbf{y}_t^T \quad \left(\overrightarrow{\mathbf{g}}_t^{r,a} \odot \overrightarrow{\mathbf{h}}_{t-1}\right)^T\right)^T + \mathbf{b}^r_{\text{fw}}\right), \tag{5.17}$$

$$\overleftarrow{\mathbf{g}}_t^{r,a} = \alpha_{\text{sigmoid}}\left(\mathbf{W}^{r,a}_{\text{bw}}\left(\mathbf{y}_t^T \quad \overleftarrow{\mathbf{h}}_{t+1}^T\right)^T + \mathbf{b}^{r,a}_{\text{bw}}\right), \tag{5.18}$$

$$\overleftarrow{\mathbf{g}}_t^{r,b} = \alpha_{\text{sigmoid}}\left(\mathbf{W}^{r,b}_{\text{bw}}\left(\mathbf{y}_t^T \quad \overleftarrow{\mathbf{h}}_{t+1}^T\right)^T + \mathbf{b}^{r,b}_{\text{bw}}\right), \tag{5.19}$$

$$\overleftarrow{\mathbf{h}}_t = (1 - \overleftarrow{\mathbf{g}}_t^{r,b}) \odot \overleftarrow{\mathbf{h}}_{t+1} + \overleftarrow{\mathbf{g}}_t^{r,b} \odot \alpha_{\text{ReLU}}\left(\mathbf{W}^r_{\text{bw}}\left(\mathbf{y}_t^T \quad \left(\overleftarrow{\mathbf{g}}_t^{r,a} \odot \overleftarrow{\mathbf{h}}_{t+1}\right)^T\right)^T + \mathbf{b}^r_{\text{bw}}\right), \tag{5.20}$$

$$\mathbf{h}_t = \left(\overrightarrow{\mathbf{h}}_t^T \quad \overleftarrow{\mathbf{h}}_t^T\right)^T, \tag{5.21}$$

$$\mathbf{p}_t = \alpha_{\text{softmax}}\left(\mathbf{W}_{\text{soft.}}\mathbf{h}_t + \mathbf{b}_{\text{soft.}}\right), \tag{5.22}$$

←

where $\mathbf{W}^r_{\text{fw}} \in \mathbb{R}^{2M\times(2M+n\cdot s)}$, $\mathbf{b}^r_{\text{fw}} \in \mathbb{R}^{2M}$, $\mathbf{W}^{r,a}_{\text{fw}} \in \mathbb{R}^{2M\times(2M+n\cdot s)}$, $\mathbf{b}^{r,a}_{\text{fw}} \in \mathbb{R}^{2M}$ and $\mathbf{W}^{r,b}_{\text{fw}} \in \mathbb{R}^{2M\times(2M+n\cdot s)}$, $\mathbf{b}^{r,b}_{\text{fw}} \in \mathbb{R}^{2M}$ (forward direction), and $\mathbf{W}^r_{\text{bw}} \in \mathbb{R}^{2M\times(2M+n\cdot s)}$,

$\mathbf{b}^{r}_{\text{bw}} \in \mathbb{R}^{2M}$, $\mathbf{W}^{r,a}_{\text{bw}} \in \mathbb{R}^{2M\times(2M+n\cdot s)}$, $\mathbf{b}^{r,a}_{\text{bw}} \in \mathbb{R}^{2M}$ and $\mathbf{W}^{r,b}_{\text{bw}} \in \mathbb{R}^{2M\times(2M+n\cdot s)}$, $\mathbf{b}^{r,b}_{\text{bw}} \in \mathbb{R}^{2M}$ (backward) are the weight matrices and bias vectors in the bidirectional LSTMGRU cell at the receiver. The weight matrix $\mathbf{W}_{\text{soft.}} \in \mathbb{R}^{4M\times M}$ and the bias vector $\mathbf{b}_{\text{soft.}} \in \mathbb{R}^{M}$ specify the final softmax layer. The parameters listed above constitute the set of trainable parameters in the LSTM-GRU BRNN receiver.

The probability vectors $\mathbf{p}_t$ output of the BRNN receiver are then used in two ways: during the optimization stage described in Sec. 5.1.3 they were used to compute the system loss, while for the testing they were employed within the sliding window sequence estimation algorithm, as explained in Sec. 5.1.4.

5.1.3 Optimization procedure

The set of transmitter and receiver BRNN parameters (denoted here by ϑ) is iteratively updated via the Adam optimizer with a learning rate of 0.001 (default setting). Similarly to the investigation of the FFNN auto-encoder, described in Chapter 4, the deep learning library TensorFlow [99] was used to perform the optimization. The objective was to minimise the average loss $\underline{L}(\vartheta)$ over a mini-batch $\underline{S}$ from the training set, given by

$$\underline{L}(\vartheta) = \frac{1}{|\underline{S}|} \sum_{\mathbf{1}_{m,t}\in\underline{S}} A\left(\mathbf{1}_{m,t}, f_{\text{BRNN-AE},t}(\ldots \mathbf{1}_{m,t-1}, \mathbf{1}_{m,t}, \mathbf{1}_{m,t+1}\ldots)\right), \tag{5.23}$$

where $A(\cdot)$ denotes the cross entropy loss function, while $f_{\text{AE-BRNN},t}(\cdot)$ is used to denote the input-to-output mapping function of the BRNN-based auto-encoder at time instance t, which can be expressed as

$$\mathbf{p}_t = f_{\text{BRNN-AE},t}(\ldots,\mathbf{1}_{m,t},\ldots) = f_{\text{Dec.-BRNN}}\left(H_{\text{IMDD}}\{f_{\text{Enc.-BRNN},t}(\ldots,\mathbf{1}_{m,t},\ldots)\}\right), \tag{5.24}$$

with H_{IMDD} denoting an operator which describes the effects of the optical IM/DD channel. In the following a detailed step-by-step description of an iteration of the optimization procedure is provided.

First, the training set S consisting of $Z = 250$ different sequences of $T_{\text{train}} = 10^6$ random input messages $m_{i,j}$ was generated, with $i \in \{1,\ldots, Z\}$ and $j \in \{1,\ldots, T_{\text{train}}\}$. At the beginning of the optimization procedure, the outputs $\overrightarrow{\mathbf{x}}_{t-1}$ and $\overleftarrow{\mathbf{x}}_{t+1}$ in the forward and backward directions of the transmitter BRNN as well as the the outputs $\overrightarrow{\mathbf{h}}_{t-1}$ and $\overleftarrow{\mathbf{h}}_{t+1}$ in the forward and backward directions of the receiver BRNN were initialised to $\mathbf{0}$. At an optimization step s, the mini-batch $\underline{S}$ of messages $m_{i,(s-1)V+1}, \ldots, m_{i,sV}$, for $1 \leq i \leq Z$ and a fixed window $V = 10$ (to keep the batch size computationally manageable), was processed by the transmitter BRNN to obtain the sequence of encoded symbols $\mathbf{x}_{i,(s-1)V+1}, \ldots, \mathbf{x}_{i,sV}$. Before feeding them into the communication channel, these symbols were transformed into the single long sequence $\mathbf{h}_{1,(s-1)V+1}, \ldots, \mathbf{h}_{1,sV}$, $\mathbf{h}_{2,(s-1)V+1}, \ldots, \mathbf{h}_{2,sV}, \ldots, \mathbf{h}_{Z,(s-1)V+1}, \ldots, \mathbf{h}_{Z,sV}$. At the input of the receiver, this transformation was reversed and the received blocks $\mathbf{y}_{i,(s-1)V+1}, \ldots, \mathbf{y}_{i,sV}$ were applied to the BRNN, obtaining output probability vectors $\mathbf{p}_{i,(s-1)V+1}, \ldots, \mathbf{p}_{i,sV}$. Then, in accordance with Eq. (5.23) from Sec.2.1.3.2, the cross entropy loss was computed and averaged over the whole mini-batch, enabling the step of backpropagation and SGD and thus concluding a single iteration of the optimization algorithm.
Every 100 steps of the optimization, the outputs $\overrightarrow{\mathbf{x}}_{t-1}$ and $\overleftarrow{\mathbf{x}}_{t+1}$, and $\overrightarrow{\mathbf{h}}_{t-1}$ and $\overleftarrow{\mathbf{h}}_{t+1}$ in the forward and backward passes of the transmitter and receiver BRNN, respectively were re-initialised to $\mathbf{0}$ in an attempt to avoid local minima. Using validation data, the convergence of the loss during training was confirmed well-within 100 000 iterations, the maximum number of iterations that were allowed.

After the auto-encoder parameters were optimized, the transceiver was combined with the sliding window sequence estimation algorithm described in Sec. 5.1.4. It is important to mention that, following the discussion in Sec. 4.4.2.1, for the training set a Mersenne twister was used as a random number generator. To ensure that during training parts or construction rules of a pseudo-random sequence are not learnt, and that training and testing datasets originate from different sources, the Tausworthe random number generator [106] was used to generate an independent testing set of data using different 250 sequences of 10^4 randomly chosen messages.

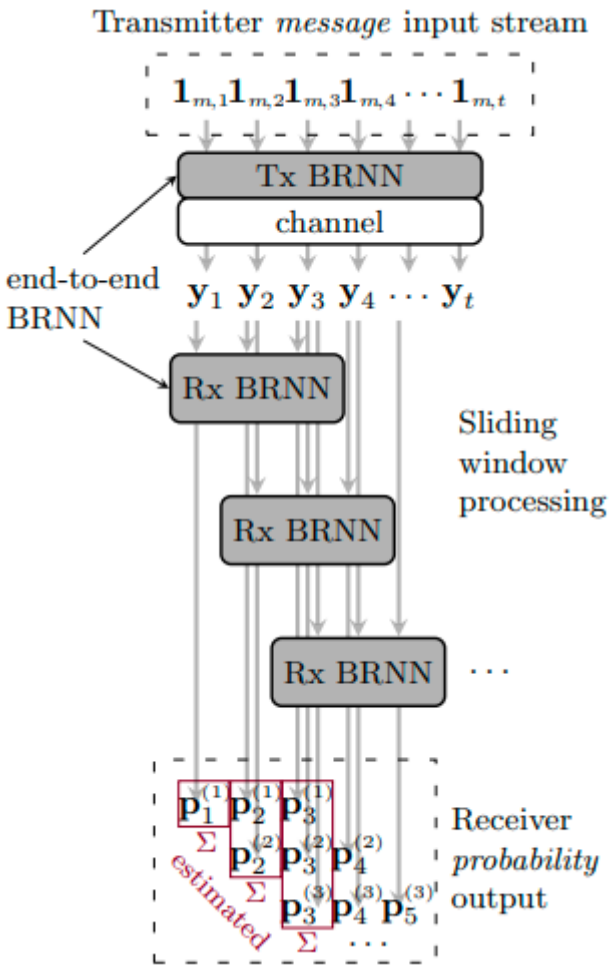

Figure 5.4: Schematic of the sliding window sequence estimation technique in which the optimized BRNN transceiver is operated. Note that $W = 3$ is chosen for illustration purposes

.

5.1.4 Sliding window sequence estimation

5.1.4.1 Algorithm description

Figure 5.4 shows a basic schematic of the sliding window sequence estimation algorithm applied on the optimized system. This algorithm was proposed in [108, 109] for the detection of sequences in molecular communication systems. The autoencoder is represented by the modules *Tx BRNN*, *channel* and *Rx BRNN*. For a given sequence of $T + W - 1$ test messages, the transmitter BRNN encodes the *full* stream of input one-hot vectors $\mathbf{1}_{m,1}, \ldots, \mathbf{1}_{m,T+W-1}$. The obtained sequence of symbols is then subject to the channel, yielding the sequence of received symbols $\mathbf{y}_1, \ldots, \mathbf{y}_{T+W-1}$. At this stage, the sliding window technique is employed to efficiently obtain probability vectors and recover the transmitted messages.

At a time t, the receiver BRNN processes the window of W symbols

$\mathbf{y}_t,\ldots,\mathbf{y}_{t+W-1}$, transforming them into W probability vectors $\mathbf{p}^{(t)}_t,\ldots,\mathbf{p}^{(t)}_{t+W-1}$ via its final *softmax* layer. Then it slides one time slot ahead to process the symbols $\mathbf{y}_{t+1},\ldots,\mathbf{y}_{t+W}$. Notice that this enables the scheme to provide multiple estimates of the probability vector at time t, for $t \geq 2$. In particular, the final output probability vectors for the first $W-1$ symbols from the received sequence are given by

$$\mathbf{p}_i = \frac{1}{i}\sum_{k=0}^{i-1} \mathbf{p}_i^{(i-k)}, \quad i = 1,\ldots W-1, \tag{5.25}$$

In Sec. 5.1.4.2 a method for the weight optimization is proposed, whose performance is examined in Sec. 5.2. Note that the final $W-1$ symbols $\mathbf{y}_{T+1},\ldots,\mathbf{y}_{T+W-1}$ from the received sequence are not fully estimated and are thus not included in the subsequent symbol error counting. For this reason the sequence length is chosen such that $T \gg W$ and thus there is only a negligible reduction in the data rate of the scheme.

In the following, a concrete example of the first few steps in the operation of the sliding window processor is presented with the window size set to $W = 3$ based on Fig. 5.4: At $t = 1$, the receiver BRNN processes the symbols $(\mathbf{y}_{t=1}, \mathbf{y}_2, \mathbf{y}_3)$ and estimates the output probability vectors $\mathbf{p}^{(t=1)}_1, \mathbf{p}^{(1)}_2, \mathbf{p}^{(1)}_3$. The receiver has generated all estimates for the received symbol at $t = 1$, i.e. $\mathbf{y}_1$, and assigns $\mathbf{p}_1 = \mathbf{p}^{(1)}_1$. Next at $t = 2$, the receiver BRNN has shifted a single slot to process $(\mathbf{y}_{t=2}, \mathbf{y}_3, \mathbf{y}_4)$ and generates $\mathbf{p}^{(t=2)}_2, \mathbf{p}^{(2)}_3, \mathbf{p}^{(2)}_4$. All estimates for $\mathbf{y}^{(2)}$ are obtained at this step and the final probability vector $\mathbf{p}_2 = \frac{1}{2}\mathbf{p}^{(1)}_2 + \frac{1}{2}\mathbf{p}^{(2)}_2$ is computed. Similarly, at $t = 3$, a final probability vector $\mathbf{p}_3 = a^{(0)}\mathbf{p}^{(3)}_3 + a^{(1)}\mathbf{p}^{(2)}_3 + a^{(2)}\mathbf{p}^{(1)}_3$ is computed for $\mathbf{y}_3$. The sliding window processing carries on for the remainder of the symbols from the received sequence.

5.1.4.2 Optimization of the weight coefficients

The *weight coefficients* $a^{(q)}$ assigned to each of the probability vector estimates provided by the sliding BRNN receiver can be optimized in order to enhance the se-

quence estimation algorithm and thus improve the overall error rate performance of the system. This section proposes an offline optimization method which consists in picking a representative test sequence of length $T+W-1$ for which all corresponding BRNN output probability vectors $\mathbf{p}_i^{(i-k)}$, with $i = W,\ldots T$ and $k = 0,\ldots,W-1$ are collected. Note that the first $W-1$ and the final $W-1$ symbols from the sequence were not considered in the optimization since the number of weight coefficients required for their estimation is lower than W (see Eq.(5.25)). The best set of coefficients $\mathbf{a} = a^{(0)} \ldots a^{(W-1)}$ is then obtained by minimising the average cross entropy, computed as

$$\bar{c}(\mathbf{a}) := -\frac{1}{T-W+1}\sum_{i=W}^{T}\mathbf{1}_{m,i}^{T}\log\left(\sum_{q=0}^{W-1}a^{(q)}\mathbf{p}_i^{(i-q)}\right) \tag{5.27}$$

where $q = 0,\ldots W-1$ and $\bar{c}$ was used to denote the average cross entropy between the input one-hot vectors and the estimated final output probability vectors. The optimization problem is convex and can easily be solved numerically. Specifically, the MATLAB `fmincon` function for finding the minimum of constrained nonlinear multivariable function was used. Section 5.2 presents the results from the performed weight coefficient optimization and reports the achieved performance improvement using the method.

5.1.4.3 Performance metric

After the sliding window receiver estimates the final probability vector for a given symbol, a decision on the transmitted message can be performed as

$$\hat{m}_t = \underset{j=1\ldots M}{\operatorname{argmax}}(\mathbf{p}_{i,j}). \tag{5.29}$$

A symbol error is counted when $m_t \neq \hat{m}_t$. The symbol error rate (SER) for the transmitted sequence is given by

$$\mathrm{SER} = \frac{1}{\lceil T \rceil_e}\sum_{i\in T_e}\mathbf{1}\{m_t \neq \hat{m}_t\}, \tag{5.30}$$

where $|T_e|$ is the number of fully estimated messages in the test sequence and the indicator function is denoted by $\mathbb{1}$, and is equal to 1 when the argument is satisfied and 0 otherwise.

5.1.5 Bit labeling optimization

In Sec. 4.4.1 it was explained that in order to examine the BER of the developed auto-encoders, a mapping function between the messages processed by the ANN and the FEC encoder/decoder output/input is needed. In particular, a bit labeling function expressed as

$$\phi : \{1, \ldots, M\} \rightarrow \mathbb{F}_2^B \tag{5.31}$$

is required. This function maps a message m to a binary vector $\mathbf{b} = b_1, \ldots, b_B$, $b_i \in \mathbb{F}_2 = \{0, 1\}$ of length $B = |\log_2(M)|$, with M and B chosen such that $M = 2^B$. In Sec. 4.4.1 the simple method of assigning the binary Gray code to the integer input messages $m \in \{1, \ldots, M\}$ was assumed. However, from the system performance results in Sec. 4.4.4.1 it was observed that, although convenient to implement, such an approach leads to a substantially degraded BER performance compared to the case when only a single bit error arises from an occurred symbol error.

Finding a bit labeling function that minimises the bit error rate is an NP-hard task that is usually solved using combinatorial optimization, e.g., a bit switching algorithm [110]. Here, the use of the *Tabu search* algorithm [111] with a Tabu list of size 256 is proposed. The algorithm is initialised with a random bit labeling and, using the outcome of the validation run, namely the estimated probabilities $\hat{P}(\hat{m}|m)$, the expected error rate is used as cost function. For a given bit labeling, the Tabu search tries all possible combinations of two elements and computes the resulting expected BER. The selected combination is the one that leads to the lowest expected BER and is not in the Tabu list. The Tabu list is then updated with this new assignment in a first-in/first-out fashion. After a pre-defined number of iterations, the overall best assignment is selected. In Sec. 5.2 the resulting BER from the optimized bit mapping is compared with the trivial (ideal) BER lower bound BER$> \hat{P}(\hat{m} \neq m|m)/B$, assuming that each symbol error yields exactly a single bit error.

It is worth mentioning that an alternative approach was suggested in [112]. It consists in modifying the auto-encoder to encode a set of B bits into a set of B decoded bits. In this case, the loss function minimises the average bit error rate. Although a viable alternative, it was found that in many circumstances the training can get stuck in a local minimum, especially when the channel input is heavily constrained. Moreover, the results presented in Sec. 5.2 show that the employed bit labeling optimization already gives a performance close to the lower bound.

Table 5.3: Simulations parameters assumed for the examined systems

Parameter	Value
Transceiver:	
M	64
n	12
Oversampling (s)	4
Receiver estimation window (W)	10
Test sequence length (T)	10^4
messages Number of test sequences	250
$R_{\text{cod.}}$	$\frac{1}{2}$ b/Sa
Channel:	
$R_{\text{samp.}}$	84 GSa/s
Sampling rate ($s \cdot R_{\text{samp.}}$)	336 GSa/s
$R_{\text{inf.}}$	42 Gb/s
LPF bandwidth (B_f)	32 GHz
DAC/ADC ENOB (N_b)	6
Fiber length (L)	varied
Fiber dispersion (D)	17 ps/(nm.km)
Fiber attenuation ($\alpha_{\text{dB/km}}$)	0.2 dB/km
Receiver noise variance ($\sigma_{\text{ref.}}$)	$2.455 \cdot 10^{-4}$

5.2 Numerical investigation of the system performance

5.2.1 Simulation parameters

Table 5.3 lists the simulation parameters assumed for the performance investigation of the proposed SBRNN auto-encoder system. To enable a straightforward comparison with the FFNN-based auto-encoder presented in Chapter 4, input messages from a set of $|M| = M = 64$ total messages (6 bits) were assumed in simulation. These were encoded by the transmitter BRNN into symbols of $n \cdot s = 48$, yielding a coding rate $R_{\text{cod.}} = 1/2$, calculated using Eq. (4.5) in Sec. 4.2.1.3. To increase the simulation resolution, an oversampling by a factor of $s = 4$ was assumed. The corresponding sampling rate thus becomes 336 GSa/s DAC rate, a factor of 4 higher than the DAC/ADC sampling rate of $R_{\text{samp.}} = 84$ GSa/s. The resulting information rate of the systems was $R_{\text{inf.}} = 42$ Gb/s. The listed values of all parameters describing the communication channel are identical to the ones used for investigating the end-to-end FFNN system and their description can be found in Sec. 4.4.3. Note that all ANN-based performance results presented in this section show the testing BER achieved by the best parameter sets found in three independent training runs,

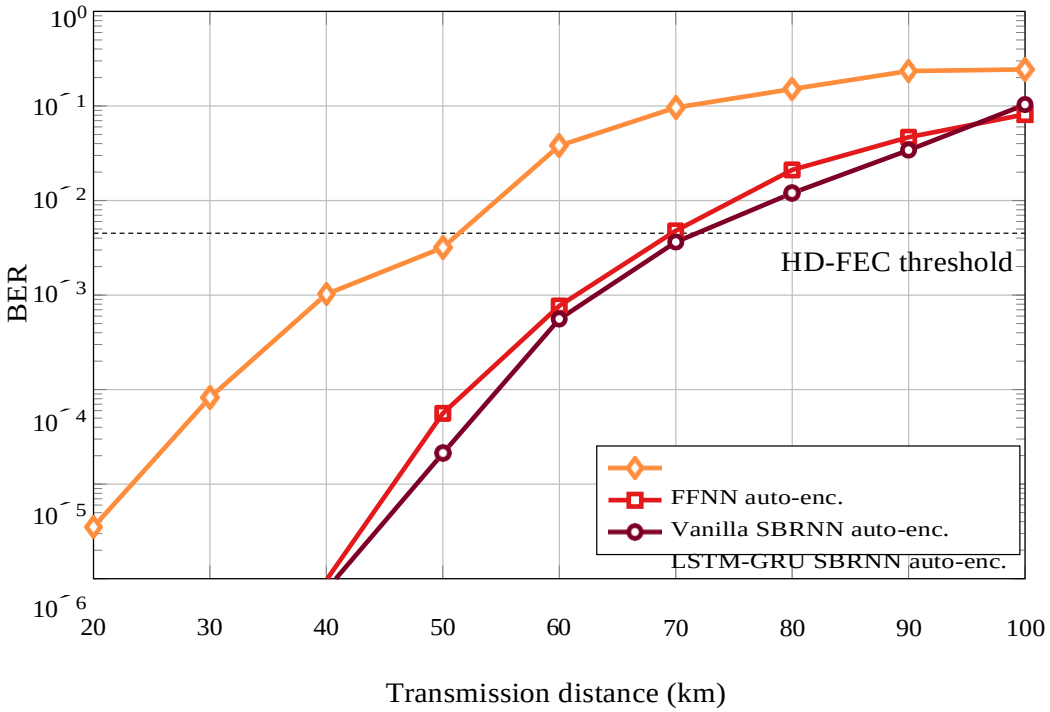

Figure 5.5: Bit error rate as a function of transmission distance for the 42 Gb/s end-to-end vanilla and LSTM-GRU SBRNN auto-encoders compared to the 42 Gb/s endto-end FFNN.

obtained with different initialisation of both the ANN parameters and the random number generators.

5.2.2 System performance

The BER performance of the examined vanilla and LSTM-GRU SBRNN autoencoder systems, computed by assigning the Gray code to the integer input messages, as described in Sec. 4.4.1, is compared to the FFNN design as a function of transmission distance at the information rate of 42 Gb/s. The performance results are shown in Fig. 5.5. Note that for this investigation all systems are trained separately for each distance from 20 to 100 km (in steps of 10 km), i.e. using the *learning at nominal distance* method described in Sec. 4.3.1. Moreover, the fixed window of $W = 10$ received symbol blocks was assumed for the SBRNN systems. In contrast to the FFNN design, the SBRNN handles data sequences, having a structure that allows mitigation of the inter-symbol interference accumulated during transmission. As a consequence, the two SBRNN auto-encoders significantly outperform the FFNN design at each of the examined distances. In particular, the results indicate that the SBRNN designs can enable communication below the $4.5 \cdot 10^{-3}$ HD-FEC threshold at distances up to 70 km, which is a substantial increase over the achievable distance of 50 km below HD-FEC for the FFNN.

In the introduction to this chapter, the main differences in the approaches for

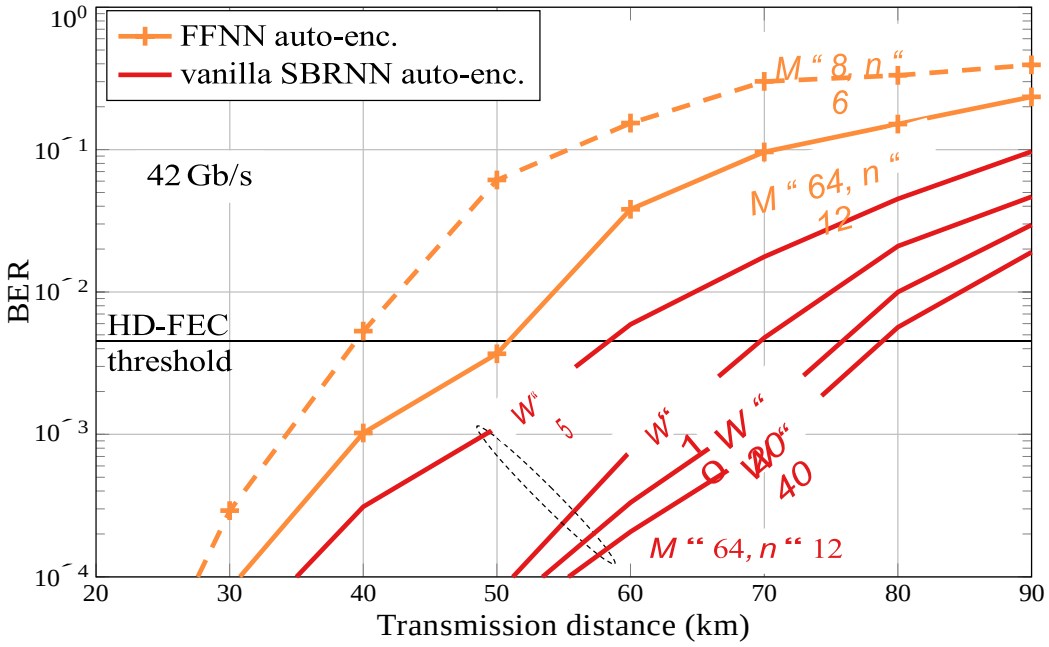

Figure 5.6: BER as a function of transmission distance for the FFNN and vanilla SBRNN auto-encoders for different neural network hyper-parameters M, n (FFNN and SBRNN) and sliding window size W (SBRNN).

expanding the processing memory in the FFNN and the SBRNN auto-encoders were highlighted. In particular, when the coding rate $R_{\text{cod.}}$ of the system is kept fixed, enhancing the resilience to chromatic dispersion in the FFNN system is associated with a rapid increase in the network dimensions (number of nodes), dictated by the hyper-parameters M and n (see Sec. 4.4.3), thus increasing the overall system complexity. On the other hand, the processing memory in the SBRNN receiver depends on the size (number of symbol blocks) of the sliding window W in the sequence estimation algorithm, described in Sec. 5.1.4. The parameter W is external to the BRNN transceiver architecture and thus adjusting it to compensate for more of the interference does not require any increase in the number of transceiver nodes. To investigate the performance of the FFNN and vanilla SBRNN auto-encoders for different M, n and W values, the BER of the systems operating at 42 Gb/s is compared in Fig. 5.6. It can be seen that for the SBRNN system, increasing W effectively reduced the BER, allowing transmission below the 6.7% HD-FEC at distances close to 80 km when $W = 40$ received blocks are processed – further increasing the improvement over the FFNN. On the other hand, if the processing memory of the FFNN auto-encoder is expanded to $W \cdot n$ samples, it would require using neural networks with input and output dimensions $M = 2^{W \cdot n \cdot R_{\text{cod.}}}$ and $W \cdot n$, respectively, resulting in a number of trainable nodes of the order of $\sim 10^9$ even for $W = 5$.

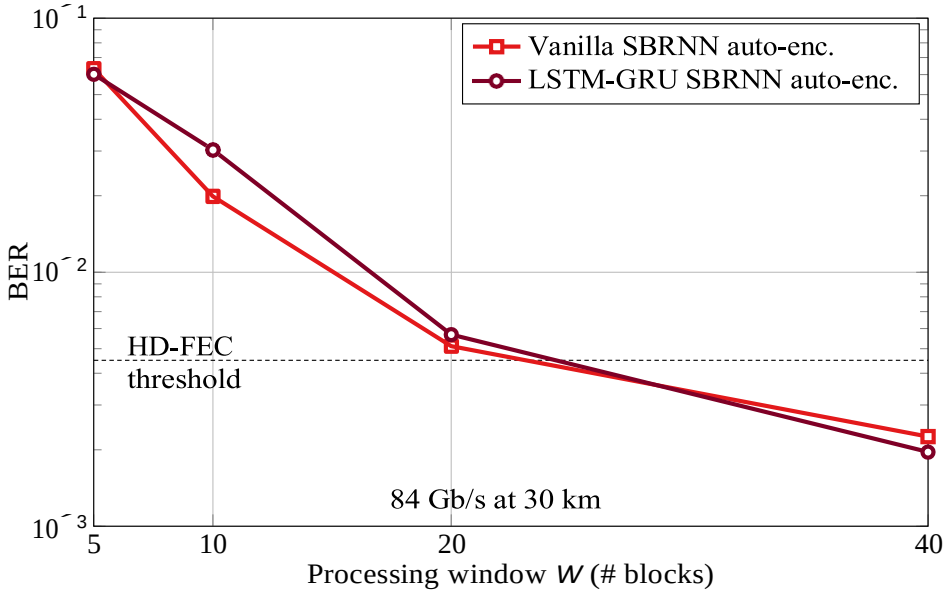

Figure 5.7: Bit error rate as a function of receiver estimation window for the 84 Gb/s vanilla and LSTM-GRU SBRNN at 30 km.

Next, the impact of the sliding window size on the performance of the vanilla and LSTM-GRU auto-encoders is studied at a higher data rate. For this investigation, the BER performance at 30 km transmission distance for the vanilla and the LSTM-GRU operating at the 84 Gb/s (achieved by reducing the block size n from 12 to 6 samples) was examined for varying window size between 5 and 40 symbol blocks (30 to 240 processed samples). The results from the investigation, which are presented in Fig. 5.7, show that for both 84 Gb/s systems the window W can be adjusted such that the BER is reduced. In particular, using W = 40 decreased the BER of the vanilla and LSTM-GRU SBRNN to $2.25 \cdot 10^{-3}$ and $1.9 \cdot 10^{-3}$, respectively, both values below the HD-FEC threshold. Nevertheless, as also previously indicated from the results in Fig. 5.6, diminishing gains for greater W were observed since system nonlinearity and noise start to dominate when ISI is compensated. Such a behaviour was confirmed in the SBRNN experiment, which is described in Chapter 6.

It should be noted that the results from Fig. 5.5 and Fig. 5.7 suggest that the vanilla and the LSTM-GRU SBRNN auto-encoders have a comparable BER performance for the examined scenarios. In the following a direct comparison between the proposed vanilla and LSTM-GRU designs was conducted, examining the training convergence of two representative 42 Gb/s systems at 30 km. The average cross entropy loss, computed as described in Sec. 5.1.3, as a function of optimization step for the vanilla and LSTM-GRU SBRNN systems is plotted in Fig. 5.8 (a) and

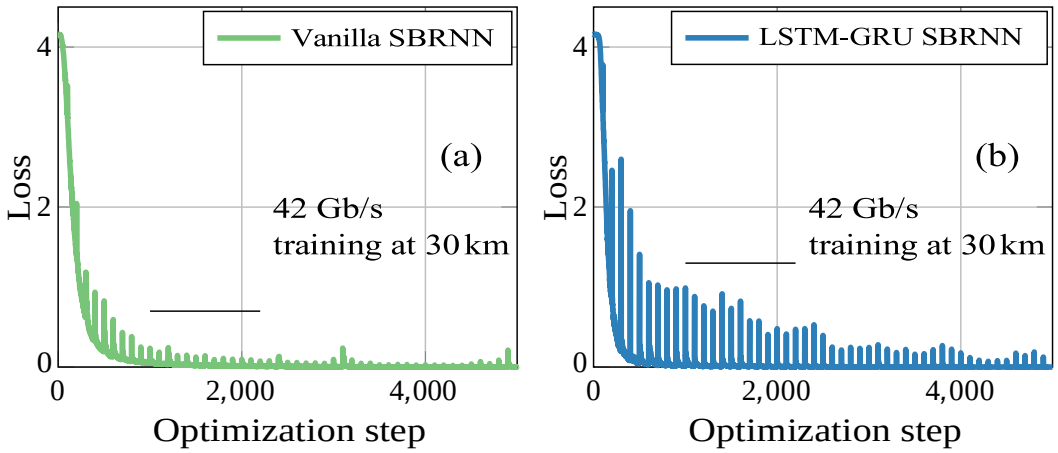

Figure 5.8: Cross entropy loss as a function of optimization step for the 42 Gb/s (a) vanilla and (b) LSTM-GRU SBRNN systems at 30 km.

5.8 (b), respectively. It can be seen that peaks in the loss value occur every 100 optimization steps. This can be attributed to the fact that the outputs in the forward and backward passes of the transmitter and receiver BRNN were re-initialised to $\mathbf{0}$ (see Sec. 5.1.3). Moreover, the LSTM-GRU structure required significantly more training iterations to converge to low values of the loss. This is an outcome of the increased number of parameters in the LSTM-GRU recurrent structure, which entails a slower training process compared to the vanilla SBRNN. In particular, the descriptions of the vanilla and the LSTM-GRU neural network processing, presented in Sections 5.1.1.2 and 5.1.2.2, shows that there are significantly fewer trainable parameters in the vanilla SBRNN system compared to the LSTM-GRU. However, as already mentioned, the performance of the vanilla design is comparable to that of the LSTM-GRU. This may be accounted to the fact that, as a result of fiber dispersion, neighbouring symbols impose stronger interference effects on the current symbol. This means that, applied to optical fiber communications, structures such as the LSTM-GRU which capture long term dependencies in a sequence, do not significantly improve the performance over vanilla RNN-based signal processing.

5.3 Comparisons with state-of-the-art digital signal processing

5.3.1 PAM and sliding window feedforward neural networkbased receiver

In this section the BER performance of the vanilla and LSTM-GRU SBRNN auto-encoders was compared with a reference system based on the conventional pulse amplitude modulation (PAM) and state-of-the-art multi-symbol nonlinear equali-

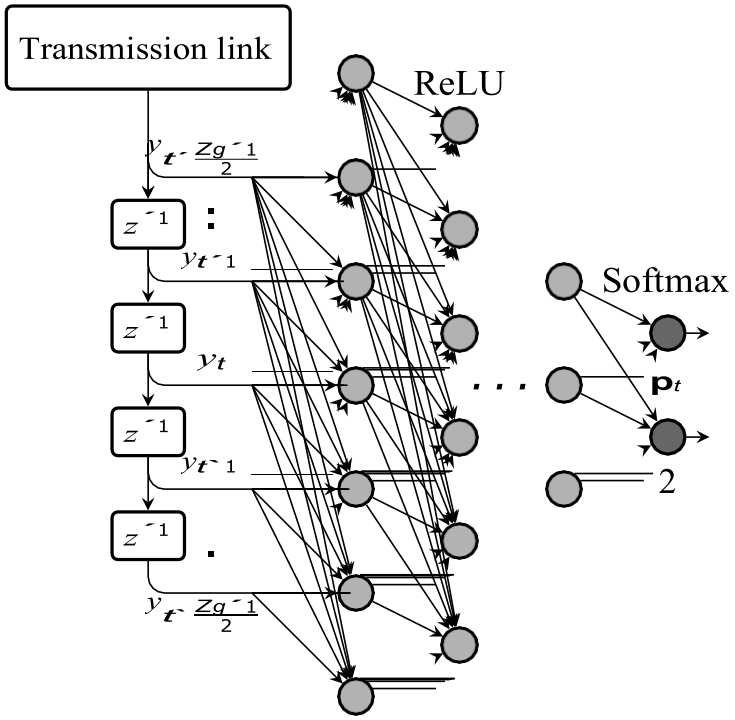

Figure 5.9: Schematic of the sliding window FFNN receiver for PAM symbols.

sation at the receiver using a large sliding window FFNN (SFFNN). It has been shown that such an equaliser performs similarly to classical nonlinear equalisation techniques [113]. This receiver design has been previously considered in low-cost IM/DD links [58, 114] as well as long-haul coherent fiber-optic systems [115]. The section also includes a discussion about the number of trainable nodes in the investigated systems, as an indicator of computational complexity.

5.3.1.1 Design of the reference system

For this investigation, 42 Gb/s two-level PAM (PAM2) transmission was considered. In particular, the system was simulated in the following way: The PAM2 transmitter directly mapped a sequence of bits into a sequence of PAM2 symbols using the levels $\{0; \pi/4\}$. The sequence of PAM symbols was pulse-shaped by a raised-cosine (RC) filter with 0.25 roll-off factor. The shaping was performed at $g = 8$ samples per symbol such that 6 PAM2 symbols (6 bits of information) are transmitted over 48 samples at the simulation sampling rate of 336 Sa/s. This setup was used to ensure that the reference PAM scheme was simulated with the same sampling rate and has coding rate and oversampling factor corresponding to the values assumed in the investigated SBRNN and FFNN auto-encoder setups, which were described in Table 4.3, Sec. 4.4.3 and Table 5.3, Sec. 5.2, respectively. The obtained PAM signal was transmitted over the optical IM/DD communication channel, modeled

with parameters in accordance to Table 5.3. Figure 5.9 shows a schematic of the receiver section in the system denoted as "Tx-PAM2 & Rx-SFFNN". The received sequence of samples, distorted after fiber propagation, was processed by a multilayer FFNN in the following way: samples corresponding to Z neighboring PAM symbols, were applied to the receiver ANN, which estimates the central of these symbols. Thus, the first layer of the neural network had parameters $\mathbf{W} \in \mathrm{R}^{Z\cdot g \times Z\cdot g}$ and $\mathbf{b} \in \mathrm{R}^{Z\cdot g}$, where Z is the number of symbols processed simultaneously by the receiver and $g = 8$, is the assumed oversampling factor. As suggested in [58, 114], the receiver design employed multiple hidden layers which further process the received samples. The number of nodes on each of the hidden layers was given by $Z \cdot g / (2^{l-1})$, where $l = 1$ corresponds to the first hidden layer, $l = 2$ is for the second hidden layer, and so on. ReLU activation functions were applied on all layers, except for the final layer, whose output represents an estimation of the central PAM2 symbol and for which the *softmax* activation was applied. Thus, the receiver output was a probability vector $\mathbf{p}_t \in \mathrm{R}^2$ of length 2 used for recovering the transmitted PAM symbol as $\mathrm{argmax}_{i=1,2}\, \mathbf{p}_{t,i}$. Note that by estimating the central of Z input PAM symbols, this receiver includes preceding and succeeding symbols in the equalisation function. After estimation of the current central symbol, the FFNN equaliser is shifted one time slot ahead to the next symbol position and estimates the following PAM2 symbol from the received sequence. It has been shown that such receivers can approximate high-order nonlinear Volterra equalisers [58, 113, 114]. For this investigation, training of the deep sliding window FFNN was performed by labeling the transmitted PAM2 sequences and using the cross entropy loss between them and the estimated output of the neural network, a standard approach for training FFNN, as described in Sec. 2.1.2. Similar to the optimization procedures implemented for the auto-encoder systems, independent data sequences were generated using the Tausworthe generator [106] in the testing phase. Because of the large size of the receiver FFNN, for the investigation of the Tx-PAM2 & Rx-SFFNN system with a processing window of $Z = 61$ symbols, 400 000 optimization iterations were used to ensure the training convergence.

5.3.1.2 BER performance

Figure 5.10 shows the BER performance of the investigated 42 Gb/s auto-encoder systems at different transmission distances in comparison with the Tx-PAM2 & Rx-SFFNN, for which the processing window Z was varied. Note that all systems were optimized separately for each distance. The Tx-PAM2 & Rx-SFFNN ($Z =$
61) system, whose receiver simultaneously processes relatively the same number of samples as the SBRNN ($Z \cdot g = 488$ to $W \cdot n \cdot s = 480$) was outperformed by

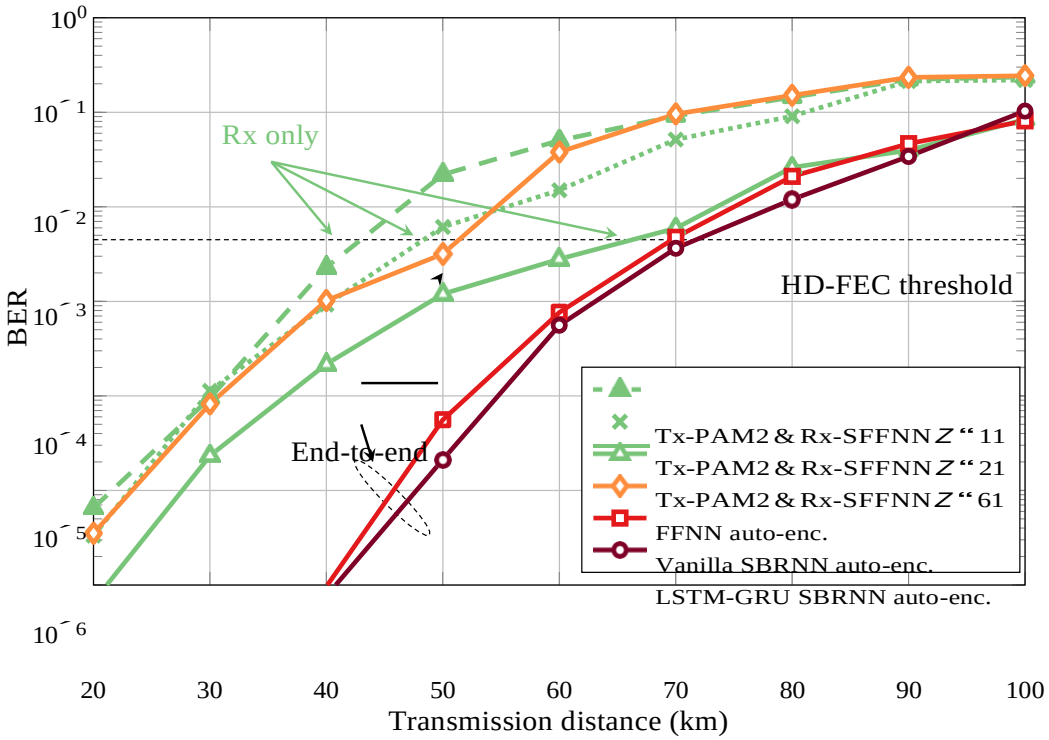

Figure 5.10: BER as a function of transmission distance for the end-to-end vanilla and LSTM-GRU SBRNN systems compared to the end-to-end FFNN as well as the PAM2 system with multi-symbol FFNN receiver. The systems operate at 42 Gb/s.

both the vanilla and the LSTM-GRU auto-encoder designs for all distances up to 80 km. In particular, for distances up to 50 km both SBRNN systems achieved BERs significantly below the value for the Tx-PAM2 & Rx-SFFNN. Moreover, the simulation prediction indicated that the Tx-PAM2 & Rx-SFFNN ($Z = 61$) could not achieve transmission below the HD-FEC beyond 60km. In contrast, the vanilla and LSTM-GRU SBRNN could increase the transmission distance to around 70km. At longer distances these three systems achieved comparable BER performance, which was, however, above 10^{-2}. Moreover, the presented results showed that the FFNN auto-encoder performed similarly to the Tx-PAM2 & Rx-SFFNN when the processing memory of the latter was decreased to $Z = 21$ or $Z = 11$.

5.3.1.3 Number of ANN nodes in the investigated systems

In Sec. 5.2.2 it was discussed that increasing the processing memory in FFNN-based DSP schemes is typically associated with expanding the neural network dimensions. This section provides a concrete example for the cases of FFNN auto-encoder and Tx-PAM2 & Rx-SFFNN systems by investigating their number of nodes, as an indicator of computational complexity, in comparison to the SBRNN auto-encoders. For this purpose, Table 5.4 summarises the neural network hyper-parameters for the aforementioned systems and evaluates them for a reference 42 Gb/s transmis-

Table 5.4: Summary of hyper-parameters for the compared ANN-based systems

Param.	Tx-PAM2 & Rx FFNN	FFNN AE	vanilla/GRUAE
Bits/Sym	1	$\log_2(M) = 6$	$\log_2(M) = 6$
Sa/Sym	$g = 8$	$n \cdot s = 48$	$n \cdot s = 48$
Proc. w (Sym)	$Z = 11/21/61$	N/A	$W = 10$
Proc. w. (Sa)	$Z \cdot g$	$n \cdot s$	$W \cdot n \cdot s$
Layers	$L = 6/7/9$	$3 + 3$	$2 \cdot (1 + 1) + 1$
Nodes	$Z\ g\ +\ \sum_{l=1}^{L-2} \left\lceil \frac{Z \cdot g}{2^{l-1}} \right\rceil + 2$	$10M + 2n\,.\,s$	$15M + 6n\,s$ $35M + 18n\,s$
Nodes at 42 Gb/s	1457 ($Z = 61$)	736	1248 3104

sion. Detailed descriptions of the hyper-parameters for each system can be found in Sections 4.2 and 4.4 (FFNN), Sections 5.1 and 5.2.2 (SBRNN), and Sec. 5.3.1.1 (Tx-PAM2 & Rx-SFFNN). For the vanilla and LSTM-GRU SBRNN, the processing window size was fixed to $W = 10$ symbols, while for the Tx-PAM2 & Rx-SFFNN, the estimation window was increased from $Z = 11$ to $Z = 21$ and then $Z = 61$. The provided summary highlights that the number of layers in the Tx-PAM2 & Rx-SFFNN was increased accordingly as Z increased. In contrast, for the case of SBRNN, only a single layer network at transmitter or receiver was used in either direction of the bidirectional processing (with an additional softmax layer at the receiver output). When comparing the number of ANN nodes for the investigated systems[1], it can be seen that in the recurrent designs there is no dependence on the receiver processing window W, i.e. expanding the processing memory of the BRNN receiver does not require an increase in the number of trainable parameters. As explained in 5.2.2 this is in stark contrast to the case of an FFNN auto-encoder, where expanding the processing capabilities is associated with larger neural networks at both transmitter and receiver. Similarly, increasing Z in Tx-PAM2 & Rx-SFFNN system would result in a substantial increase in the number of nodes in the receiver. As a consequence, when the number of nodes was evaluated for 42 Gb/s systems, the vanilla SBRNN design had fewer trainable parameters compared to the TxPAM2 & Rx-SFFNN, while processing the same amount of received samples. This difference would become even more pronounced when the processing window for each system is increased to capture more of the interference.

[1]An example of how the nodes in the SBRNN system were counted is provided in Appendix B.1.

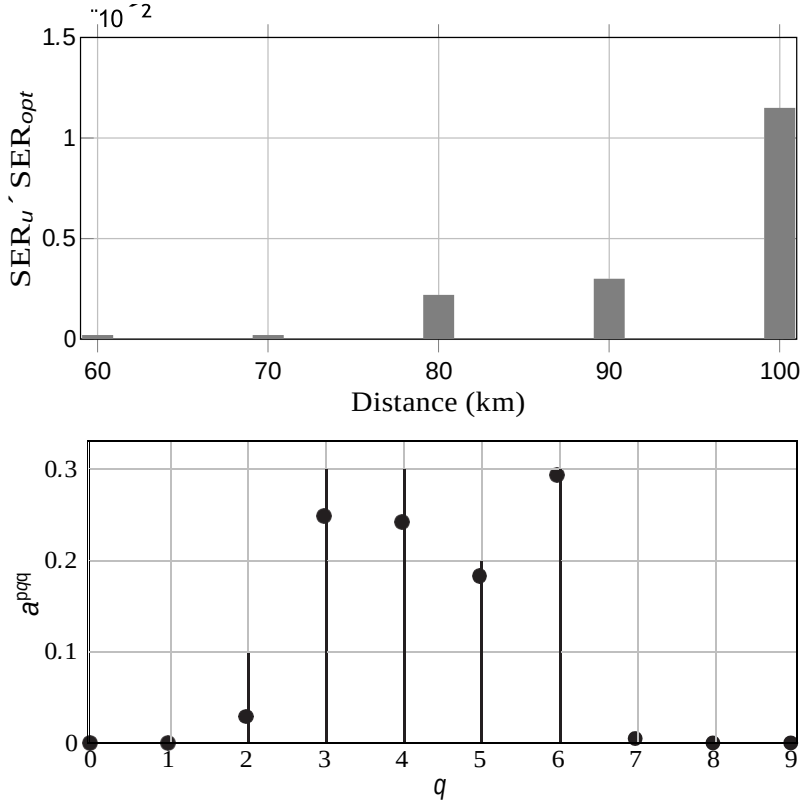

Figure 5.11: Top: SER difference as a function of distance for the two approaches for final probability vector estimation in the sliding window algorithm ($W=10$). SER_u is obtained assuming $a^{(q)} = \frac{1}{W}$, $q = 0, \ldots, W-1$ in Eq. (5.26), while for SER_{opt} the coefficients $a^{(q)}$ are optimized. Bottom: $a^{(q)}$ assignments after optimization for the system at 100 km.

5.3.2 SBRNN auto-encoder enhancement via optimization of the sequence estimation and bit labeling functions

In this section, the results from the optimization of the weight coefficients in the sliding window sequence estimation algorithm, a problem formulated in Sec. 5.1.4.2, are presented. Moreover, the performance gains from the optimization of the bit-tosymbol mapping function, an essential aspect for the auto-encoder systems, using the method proposed in 5.1.5 are examined.

5.3.2.1 Performance after weight optimization

Figure 5.11 (top) shows the improvement in symbol error rate (SER) at different distances for a fixed sliding window size of $W = 10$ after optimization of the coefficients $a^{(q)}$ in Eq. (5.26). The corresponding SER performances when equal weights $a^{(q)} = \frac{1}{W}, q = 0, \ldots, W-1$ are assigned are used as a reference. Note that the set of coefficients is optimized separately for each distance. It can be seen that the benefit from weight optimization becomes more pronounced as the distance increases.

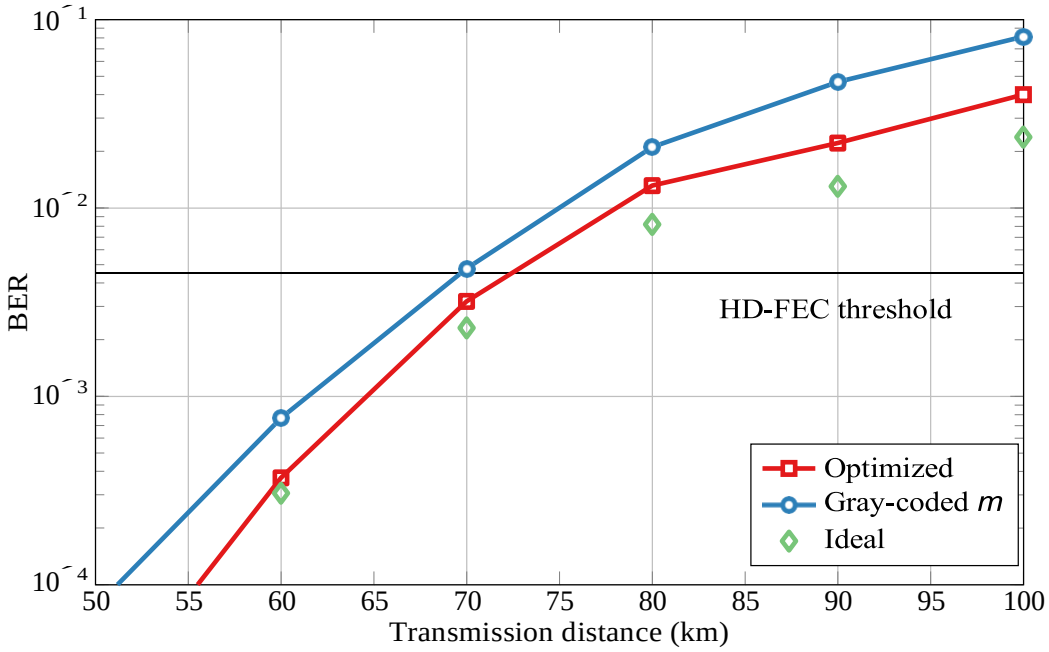

Figure 5.12: Bit error rate as a function of distance for a reference vanilla SBRNN autoencoder, which utilizes two different bit-to-symbol mapping approaches: assigning the Gray code to the transmitted message m, proposed in Sec. 4.4.1, and performing the combinatorial algorithm-based optimization, proposed in Sec. 5.1.5. As an indicator, the ideal bit-to-symbol mapping when a single symbol error gives rise to a single bit error is also displayed.

The $a^{(q)}$ assignments after optimization at 100 km are depicted in Fig. 5.11 (bottom). Greater weights are assigned to the middle indices, indicating that a received symbol is estimated with a higher accuracy by the receiver when relatively equal amount of preand post-cursor interference is captured.

5.3.2.2 Performance after bit-to-symbol mapping optimization

The BER as a function of distance for a reference $W = 10$ vanilla SBRNN auto-encoder, computed using the bit-to-symbol mapping algorithm described in Sec. 5.1.5, is plotted in Fig. 5.12. For a comparison, the figure also shows the BER performance of the system when the ad hoc approach of assigning the Gray code to the input $m \in \{1, \ldots, M\}$, proposed in Sec. 4.4.1, is employed. The trivial case (denoted "ideal") when a single block error results in a single bit error is also shown as a reference. It can be seen that by conducting the bit-to-symbol mapping optimization, the performance of the system is improved, leading to an increase in the achievable distance below the HD-FEC threshold to 70 km. Moreover, the obtained BER after optimization is close to the ideal case for all examined distances.

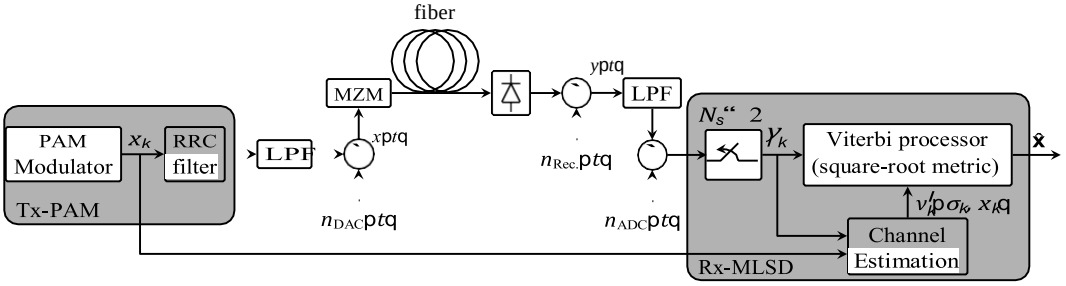

Figure 5.13: Schematic diagram of the system used to evaluate the performance of the MLSD receiver in IM/DD optical links.

5.3.3 PAM and maximum likelihood sequence detection

5.3.3.1 Design of the reference system

To establish a relevant benchmark for the performance of the SBRNN auto-encoder using a classical DSP technique, a reference M-PAM transmission system ($M \in \{2, 4\}$) with a receiver based on maximum likelihood sequence detection (TxPAM&Rx-MLSD) was considered. Receivers based on MLSD have been widely considered for optical IM/DD transmission [68, 69] and represent a valid performance reference scheme. Nevertheless, it is worth noting that the auto-encoder performs an optimization of both the transmitter and receiver designs, while in contrast, the transmitter is assumed fixed in the Tx-PAM&Rx-MLSD scheme. Forthis investigation, the vanilla SBRNN design was used, since the results presented in this chapter clearly showed that in the framework of optical IM/DD communication, the processing capabilities of such an auto-encoder is similar compared to the LSTM-GRU structure, without the associated extra complexity.

The MLSD receiver assumes that the communication channel behaves according to a Markov process, thus facilitating the use of the Viterbi algorithm [67] which selects

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, \hat{p}(\mathbf{y}|\mathbf{x}), \tag{5.32}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_{N_{\text{sym}}})$ is the sequence of transmitted PAM symbols, $\mathbf{y} = (y_1^1, y_1^2, \ldots, y_1^{N_s}, y_2^1, y_2^2, \ldots, y_2^{N_s}, \ldots, y_{N_{\text{sym}}}^{N_s})$ is the sequence of corresponding received samples. Here N_{sym} and N_s denote the number of PAM symbols in the transmitted sequence and number of samples per symbol, respectively. The joint distribution $\hat{p}(\mathbf{y}|\mathbf{x})$ is an approximation of the true channel likelihood $p_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x})$, whose accuracy depends on the number of states and metric used in the Viterbi processor. For optical IM/DD systems, a more convenient option for an MLSD receiver is to operate on the distribution $p(\bar{\mathbf{y}}|\mathbf{x})$, where $\bar{\mathbf{y}} = (\bar{y}_1^1, \bar{y}_1^2, \ldots, \bar{y}_1^{N_s}, \bar{y}_2^1, \bar{y}_2^2, \ldots, \bar{y}_2^{N_s}, \ldots, \bar{y}_{N_{\text{sym}}}^{N_s})$ with $\bar{y}_k^l = y_k^l$ for $l = 1, \ldots N_s$ and $k = 1, 2, \ldots, N_{\text{sym}}$ [69]. The channel state is defined as

$$\sigma_k = (x_{k-\mu}, x_{k-\mu+1}, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{k+\mu}) \tag{5.33}$$

where μ is the number of preand post-cursor PAM symbols determining the channel memory, the distribution of y_k^l conditioned on (σ_k, x_k) was shown to be well-approximated by a Gaussian with equal variances [69]. Under such an approximation, and the assumption of uncorrelated samples y_k^l conditional to (σ_k, x_k), $p(\bar{\mathbf{y}}|\mathbf{x})$ can be suitably factorised via the channel state definition in Eq. (5.33) for the use in a Viterbi processor.

The branch metric of the Viterbi trellis at time k can thus be defined as

$$\lambda_k(\sigma_k, x_k) = \sum_{l=1}^{N_s} \left(\sqrt{y_k^l} - v_k^l(\sigma_k, x_k) \right)^2 \tag{5.34}$$

where y_k^l is the l-th out of N_s samples within the k-th symbol period, and $v_k^l(\sigma_k, x_k)$ is with $y^l(\sigma_k, x_k)$ indicating the value of the received sample y^l when the pair (σ_k, x_k) occurs. The *square-root metric* in Eq. (5.34) was introduced in [68] as a convenient low-complexity alternative to the histogram-based metric with comparable performance [69]. At the end of each given window of T symbols, the Viterbi processor makes a decision on the previous T transmitted symbols by minimising the sequence metric $\Lambda_T = \sum_{k=1}^{T} \lambda_k(\sigma_k, x_k)$ over all surviving state sequences $[\sigma_1, \sigma_2, \ldots, \sigma_T]$.

The schematic diagram of the system model employing MLSD is shown in Fig. 5.13 (with the channel parameters taken from Table 5.3). At the transmitter, a Gray-labelled PAM2 ($\{0;\pi/4\}$) or PAM4 ($\{0;\pi/12;\pi/6;\pi/4\}$) mapper was used to make a transition between bits and symbols. This is followed by pulse-shaping at 2 samples per symbol by a root-raised cosine (RRC) filter with a roll-off factor of 0.25. DAC/ADC rates of 84 GSa/s and 42 GSa/s are assumed for the PAM2 and PAM4 cases, respectively, resulting in a fixed data rate of 42 Gb/s for both systems. A signal corresponding to a sequence of 10^5 symbols was transmitted through the optical IM/DD communication channel described in Chapter 3. At the receiver, the estimation of the $v^l(\sigma_k, x_k)$ was first done computing Eq. (5.35) over a sequence of 10^7 symbols. The received signal, sampled at $N_s = 2$ samples per symbol, is then passed directly to the Viterbi processor which performs the MLSD.

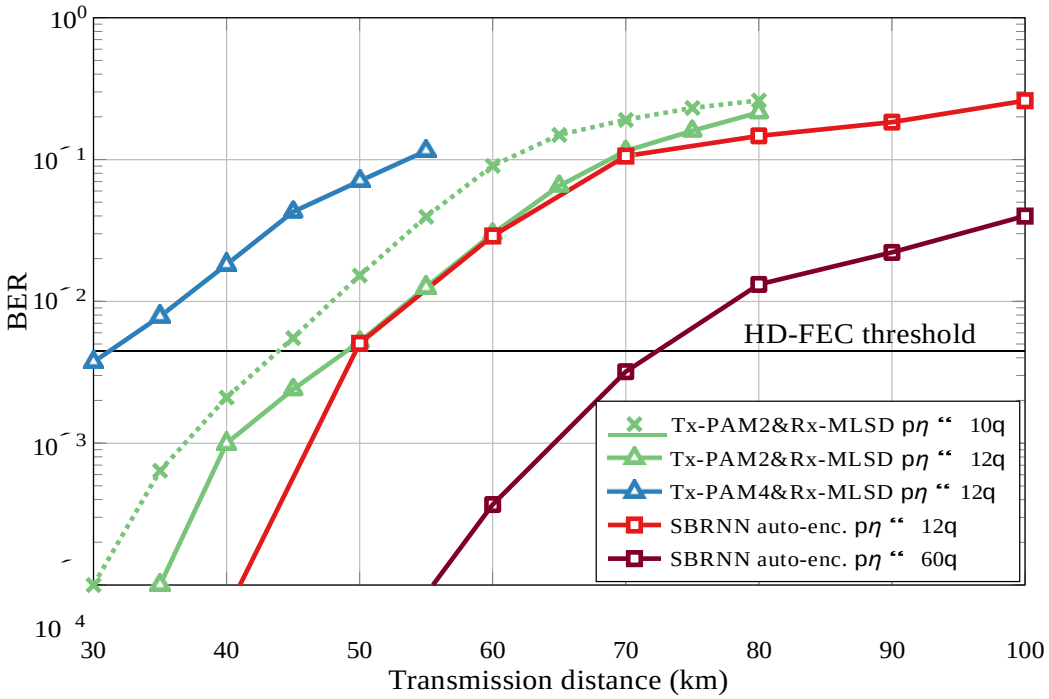

Figure 5.14: Bit error rate as a function of transmission distance for the 42 Gb/s SBRNN auto-encoder and M-PAM & Rx MLSD systems ($M \in \{2,4\}$). In the case of MLSD $\eta = \mu \log_2(M)$, where μ represents the number of pre and post-cursor PAM symbols defining one of M^{μ} channel states. In the case of SBRNN $\eta = W \log_2(M)$ is the number of bits inside the processing window.

5.3.3.2 BER performance

The performance of the SBRNN auto-encoder was numerically evaluated using the design parameters given in Table 5.3. Note that weight optimization in the sliding window estimation was performed as described in Sec. 5.1.4.2. In an attempt to set up a fair comparison between the investigated systems, a parameter η was fixed, which in the case of SBRNN corresponded to the number of information bits processed inside the sliding window $\eta = W \log_2(M)$. For the PAM transmitter with MLSD receiver, this number denoted the amount of bits contained within a channel state in the Viterbi algorithm $\eta = \mu \log_2(M)$, where $M \in \{2, 4\}$ was the PAM order and μ denoted the number of post and pre-cursor PAM symbols which formed a Viterbi state. Note that in Tx-PAM2&Rx-MLSD and Tx-PAM4&Rx-MLSD, a fixed $\eta = 12$ also meant using the same number of Viterbi states (4096). Thus, the Tx-PAM4&Rx-MLSD scheme accounted for a decreased amount of memory compared to the Tx-PAM2&Rx-MLSD.

Figure 5.14 shows the BER performance of the examined systems as a function of transmission distance. It can be seen that for $\eta = 12$, the SBRNN auto-encoder and Tx-PAM2&Rx-MLSD had a comparable performance at all examined distances beyond 50 km. Both systems outperformed the Tx-PAM4&Rx-MLSD, where the

obtained BER was below the HD-FEC threshold up to 30 km. This was due to the decreased sensitivity of the PAM4 signal and Viterbi memory ($\mu = 6$). Moreover, the results indicated that assuming a wider processing window of $\eta = 60$ for the SBRNN leads to a significant BER improvement. Note that the BER of the SBRNN auto-encoder was obtained using the bit-to-symbol mapping algorithm described in Sec. 5.1.5.

5.3.3.3 Complexity

In this section the computational complexity of the two systems was compared using the *floating point operations per decoded bit* ($\text{FLOPS}_{\text{pdb}}$) as a common metric. In the case of the SBRNN auto-encoder, the counted FLOPS occur in matrix multiplications, bias additions and element-wise nonlinear activations in both direction of the recurrent structures as well as multiplications and additions in the final probability vector estimations. Detailed description of the SBRNN auto-encoder hyperparameters can be found in Sections 5.1 and 5.2.2. At the transmitter, the number of FLOPS performed in the encoding of a single bit was given by

$$\text{FLOPS}_{\text{pdb}}^{[\text{SBRNN-TX}]} = \frac{2n \cdot s(2(M + n \cdot s) + 1)}{\log_2(M)}, \tag{5.36}$$

where M, n and s are hyper-parameters of the neural network, i.e. pre-defined design choices which do not change with the processing memory. Correspondingly, at the receiver the number of FLOPS required for decoding were

$$\text{FLOPS}_{\text{pd}}^{[\text{SBRNN-RX}]} = \frac{W(24M^2 + 8Mn_{\cdot}\, s + 5M + 2)}{\log_2(M)}, \tag{5.37}$$

which indicates a *linear* dependence of the floating point operations on the processing window W.

For the Viterbi processor, the amount of floating point operations scales linearly with the number of states in the trellis which is equal to M^{μ} . Assuming a fully populated trellis, the Viterbi processor performs for each trellis section $M^{\mu+1}$ branch metric calculations, $M^{\mu+1}$ additions, and $M^{\mu+1}$ comparisons (other than the storage of M^{μ} sequences). As for the computation of the branch metric, a total of $3N_s$ +(N_s − 1) FLOPS are required (assuming addition, multiplication, and squareroot use 1 FLOP each). Moreover, each comparison is accounted for as 1 FLOP and the additional set of comparisons required for the selection of the most likely sequence at the end of the Viterbi decoding is considered negligible. Thus, the

[2] An example of how the number of $\text{FLOPS}_{\text{pdb}}$ in the SBRNN system were counted is provided in Appendix B.2.

overall number of required FLOPS per decoded bit is approximately given by

$$\text{FLOPS}_{\text{pdb}}^{[\text{MLSD}]} \approx \frac{9M^{\mu+1}}{\log_2(M)}, \tag{5.38}$$

estimated for $N_s = 2$. Importantly, the expression in Eq. (5.38) indicates that the number of $\text{FLOPS}_{\text{pd}}^{[\text{MLSD}]}$ exhibits an *exponential* dependence on the processing memory μ of the detector. In presence of strong intersymbol interference this may result in computationally prohibitive demands or increasingly sub-optimal performance.

5.4 Summary

In this chapter, the design of an optical fiber transceiver using a bidirectional recurrent neural network (BRNN) was proposed and investigated. The BRNN-based auto-encoder was combined with a sliding window sequence estimation algorithm (SBRNN), efficiently handling the memory in the communication channel. The detailed numerical investigation showed that such a design has an improved resilience to intersymbol interference compared the feedforward neural network-based autoencoder, which was investigated in Chapter 4. Importantly, comparisons with conventional PAM transmission based on state-of-the-art nonlinear receivers as well as classical maximum likelihood sequence detection (MLSD) were carried out. The conducted research revealed that the SBRNN auto-encoder can increase reach or enhance the data rate for shorter distances compared to the benchmark PAM systems without the associated computational complexity of such schemes. The work was published in [116–118]. The SBRNN auto-encoder system was successfully verified on a transmission test-bed as part of the experimental work on end-to-end deep learning-based optical fiber systems described in Chapter 6. The investigation in Chapter 6 covered detailed comparisons with PAM systems using sliding window FFNN receivers as well as nonlinear Volterra equalisers.

Chapter 6

Experimental demonstrations, comparisons with state-of-the-art DSP and optimization using measured data

This chapter describes the setup and presents the results from three different experiments, which were conducted throughout the Ph.D. programme to demonstrate the concept of end-to-end deep learning in optical fiber communications. The diagram presented in Fig. 6.1 summarises the scope of each experiment. The first two investigations covered the implementation and performance of the feedforward (FFNN) and sliding window bidirectional recurrent neural network (SBRNN) transceiver designs proposed in Chapter 4 and Chapter 5, respectively. The third investigation involved the development and experimental verification of a novel method for endto-end system optimization using a generative model of the optical transmission link, a technique which is described in Sec. 6.6.

The chapter is organized as follows: First, the optical IM/DD transmission test-bed, common for each of the experiments, is described in Sec. 6.1. Next, Section 6.2 presents the different strategies which were considered when implementing

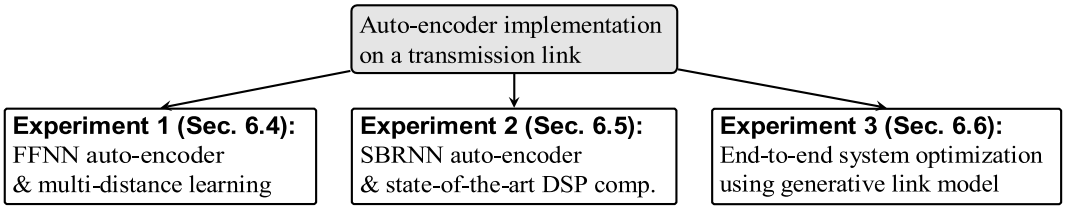

Figure 6.1: Diagram showing the three experiments, which were conducted for the demonstration of the end-to-end deep learning concept in optical fiber communication systems.

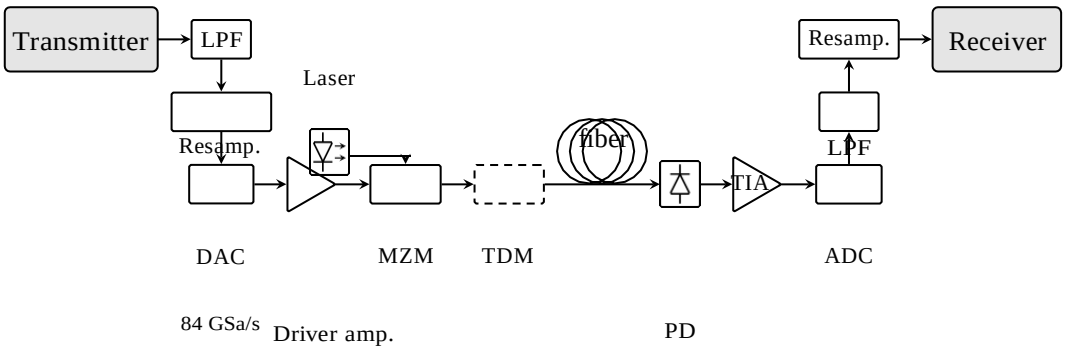

Figure 6.2: Schematic of the experimental optical IM/DD transmission test-bed used for investigation of the digital signal processing schemes in this chapter. LPF: lowpass filter, DAC: digital-to-analogue converter, MZM: Mach-Zehnder modulator, TDM: tunable dispersion module, PD: photodiode, TIA: trans-impedance amplifier, ADC: analogue-to-digital converter.

the auto-encoder systems on an actual transmission link. These include optimization based on transmission in simulation using the explicit link model (Chapter 3) as well as learning of the transceiver parameters from experimentally collected data in an end-to-end or receiver-only mode. The demonstration of the FFNNbased auto-encoder is described in Sec. 6.4 with comparisons to PAM transmission with conventional linear equalisation. The section also includes the experimental verification of the multi-distance learning method from, which was developed in Sec. 4.3.2. Section 6.5 details the experimental implementation of the end-to-end SBRNN system. It includes investigation of the SBRNN performance compared to state-of-the-art receiver DSP based on classical as well as deep learning approaches. The chapter concludes with the description and experimental demonstration of the end-to-end system optimization algorithm using a generative model.

6.1 Optical transmission test-bed

This section introduces the experimental transmission test-bed used to investigate the performance of the auto-encoders as well as the reference DSP schemes considered in the chapter. Figure 6.2 shows a schematic of the test-bed. An intensity modulation/direct detection fiber transmission link without in-line optical amplification was considered. Note that in the case of auto-encoders, the transmitter and receiver are artificial neural networks, while for the systems used for performance comparisons, the transmitter is based on PAM, for which different conventional and ANN-based receivers were considered. For all systems, the electrical signal output of the transmitter was filtered in the digital domain by a brick-wall low-pass filter (LPF) which had a bandwidth of 32 GHz. The signal was then re-sampled (module *Resamp.*) and applied to the digital-to-analogue converter (DAC) which was operated at 84 GSa/s. The obtained electrical waveform was pre-amplified by a driver amplifier and used to externally modulate the 1550 nm laser via a Mach-Zehnder

modulator (MZM). For the auto-encoder system, the MZM was carefully biased to match the simulation output, while for the reference PAM experiments, the bias was set at the quadrature point. The optical power which was input to the fiber for all systems is set to 1 dBm. The modulated optical signal was propagated over a fixed length of standard single mode fiber (SSMF) and the received waveform was direct-detected by a positive-intrinsic-negative (PIN) photo-diode (PD) combined with a trans-impedance amplifier (TIA). The signal was real-time sampled by an analogue-to-digital converter (ADC). After synchronisation, scaling, offset and resampling, the digitised photocurrent was stored for digital signal processing at the receiver. Note that re-scaling and offset were necessary because the employed receiver was AC-coupled and also has a non-ideal photoresponsivity. It should also be noted that in the experimental setup for the investigation of the FFNN auto-encoder (Experiment 1), a tunable dispersion module (TDM) was deployed to enable sweeping the dispersion around a given value set by the SSMF length. The reason was the verification of the multi-distance learning method. Thus, for Experiment 1 only, the modulated optical signal first entered the TDM allowing to vary the dispersion from −170 to +170 ps/nm in steps of 10 ps/nm, before being launched at a power of 1 dBm into the SSMF span.

6.2 System implementation strategies

The different strategies for implementing the auto-encoder systems on an actual transmission link are summarised with the schematic shown in Fig. 6.3. Within the auto-encoder framework, which was introduced in Sec. 4.1, the channel is considered as a segment of the end-to-end computational graph representing the complete communication system. Thus, the end-to-end learning concept can be readily applied in scenarios where the channel model is known and differentiable. In this case, the transceiver is optimized via simulation transmission and applied “as is” to the real system, as shown in Fig. 6.3 i). Such systems present a viable perspective for low-cost optical fiber communications since no optimization process is required after deployment. However, as first reported in [61] and also seen from the presented results in the chapter, the performance of transceivers learned on a specific model assumption often deteriorates when applied to an actual transmission link. In such cases, the ANN parameters could be optimized to the specific link properties using experimentally collected data.

The optimization is easier to implement at the receiver (Fig. 6.3 ii)), because the backpropagation [12, 13] for computing gradients extends only to the link output. For receiver-only optimization, the set S of training elements is formed by the pairings of input messages m_t and the corresponding received blocks of samples

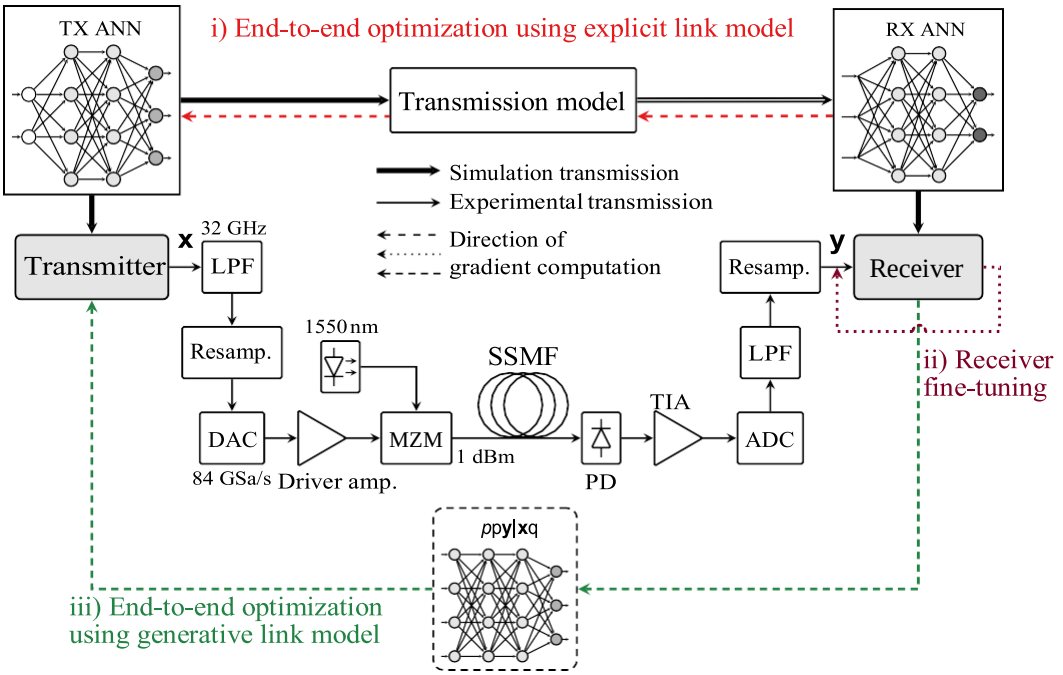

Figure 6.3: Schematic of the experimental optical IM/DD transmission test-bed, showing the methods for system optimization using i) numerical simulation of the link;
ii) & iii) experimental traces.

(symbols) $\mathbf{y}_t$ from the experimental traces (see Sec. 6.3). The receiver ANN parameters ϑ_{Rx} are adjusted via the Adam optimizer, described in Sec. 2.2.1.2, aimed at minimising the average loss $\underline{L}$, computed at the receiver output, over a mini-batch $\underline{S}$ from the training set, given by

$$\underline{\mathscr{L}}(\boldsymbol{\theta}_{\text{Rx}}) = \frac{1}{|S|} \sum_{(m_t, \mathbf{y}_t) \in S} \ell\left(m_t, f_{\text{rec}}(\ldots, \mathbf{y}_t, \ldots)\right), \tag{6.1}$$

where f_{rec} is used to denote a generic receiver ANN function. Note that, in the case of an SBRNN auto-encoder the receiver will take multiple received symbols as an input. In this work, the cross entropy loss function, defined in Sec. 2.1.2.2, Eq. (2.9) was used in the optimization of receiver parameters using measured data.

However, the transmission link can be considered a *black box* for which only inputs and outputs are observed, precluding backpropagation to the transmitter parameters. As a consequence, optimizing the transmitter without the explicit knowledge of the model for the underlying physical channel remains an open problem. In this thesis, a framework for transmitter optimization using a generative model of the optical link was developed and experimentally demonstrated. The method, which is described in detail in Sec. 6.6 uses a generative adversarial network for approximating the channel conditional distribution $p(\mathbf{y}/\mathbf{x})$. As shown in Fig. 6.3 iii),

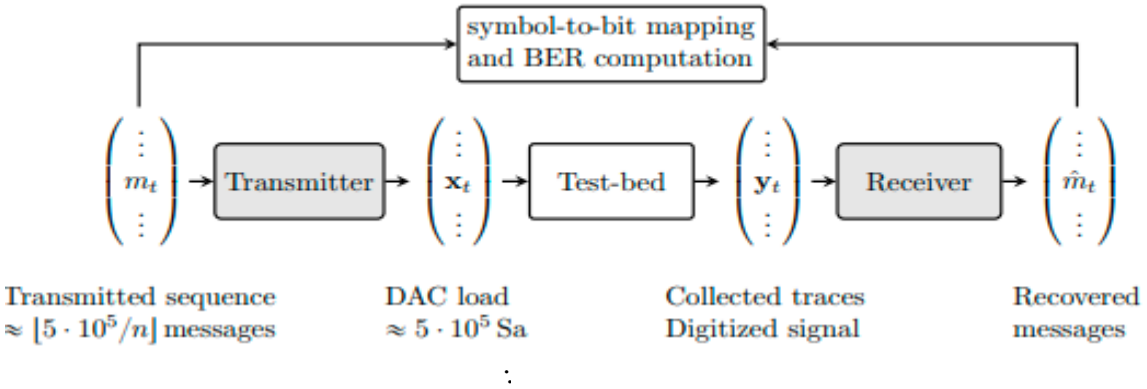

Figure 6.4: Basic schematic showing the data generation, collection and error counting. Each of the experiments involved multiple DAC loads with different random sequences.

the obtained ANN-based model is applied *in lieu* of the transmission link during optimization, enabling backpropagation and thus the computation of the transmitter gradients.

Note that an alternative mechanism for transmitter optimization which does not rely on backpropagation through the channel was proposed in [119]. It consists in separating the transmitter and receiver training in two iteratively executed processes. The receiver training is performed in the aforementioned manner, which, as already mentioned, does not require the presence of a channel model. The method circumvents the use of a model in the transmitter ANN training by viewing its optimization as a reinforcement learning task. However, the approach requires a dedicated feedback link to communicate to the transmitter the loss computed at the receiver output. For this reason, it was not considered in the investigation.

6.3 Collection of representative experimental data for optimization

It was explained in Sec. 4.4.2.1 that during the transceiver optimization in simulation, new random input messages can be continuously generated using state-ofthe-art random number generators to avoid learning representations of a sequence. In the experimental validation, long random sequences (*not* PRBS, as suggested in [104]) are generated and then processed by the transmitter ANN to obtain a waveform, loaded (after filtering and resampling) into the DAC. Figure 6.4 shows a basic schematic of the procedure for data generation, collection and error counting, forming an experimental measurement. During each experiment, multiple DAC loads with different random sequences were performed, forming a large database of experimental traces from different measurements.

For the process of receiver ANN optimization using measured data, the mini-

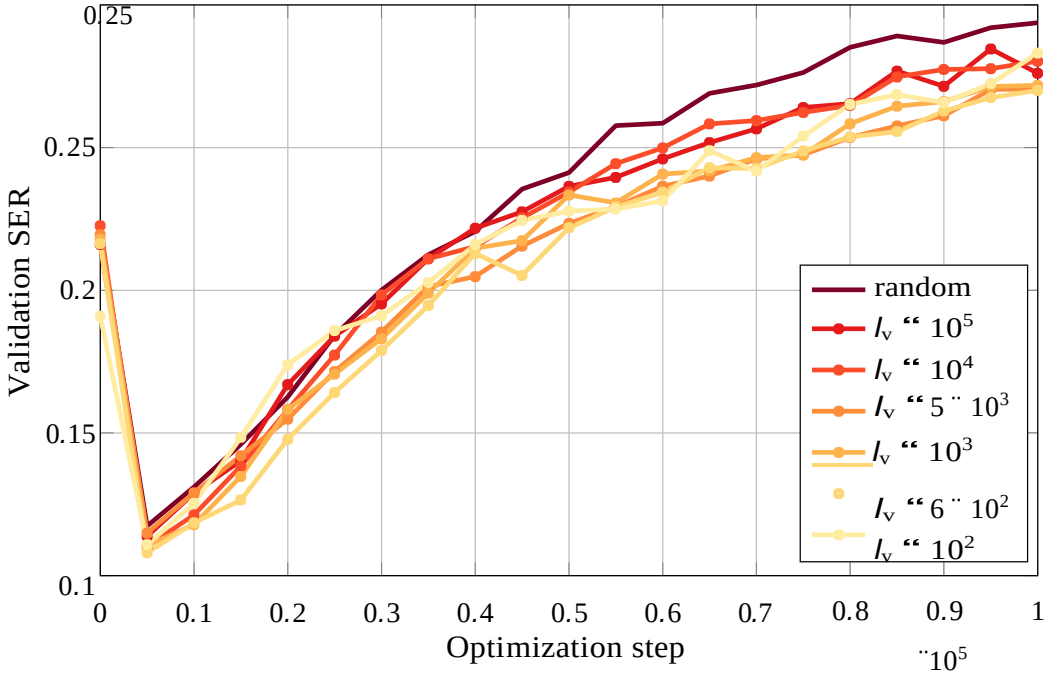

Figure 6.5: Validation symbol error rate as a function of optimization step for different lengths l_v of the validation pattern. The receiver ANN of the system is optimized using traces from the transmission of training sequences with a short pattern $l_{tr} = 6 \cdot 10^2$.

batch of training examples is formed by randomly picking messages and corresponding received blocks of symbols from a subset of the database (combining multiple measurements). To investigate system performance, the trained and stored models are used to perform testing on a disjoint subset of the database of experimental traces, having no overlap with the subset used for training. This procedure ensures that the presented experimental results are achieved with independent data. However, it is worth noting that, due to the long memory of the fiber, it is not possible to capture the interference effects of all possible sequences of symbols preceding and succeeding the symbol under consideration in the experiment. Hence, a special care should be taken when re-training the receiver ANN, such that it does not learn to adapt to the interference pattern of the sequence. The results in this section investigate the effects of re-training based on repeated sequences to verify that a sufficiently large set of different experimental traces was captured.

The impact of repeated sequences on the optimization performance was studied in the following way: In simulation, an auto-encoder system is trained on a simplified channel model, including only a few of the optical IM/DD transmission effects described in Chapter 3. Then, transmission of sequences with repeated pattern is simulated over the full channel model, collecting the data at the output of

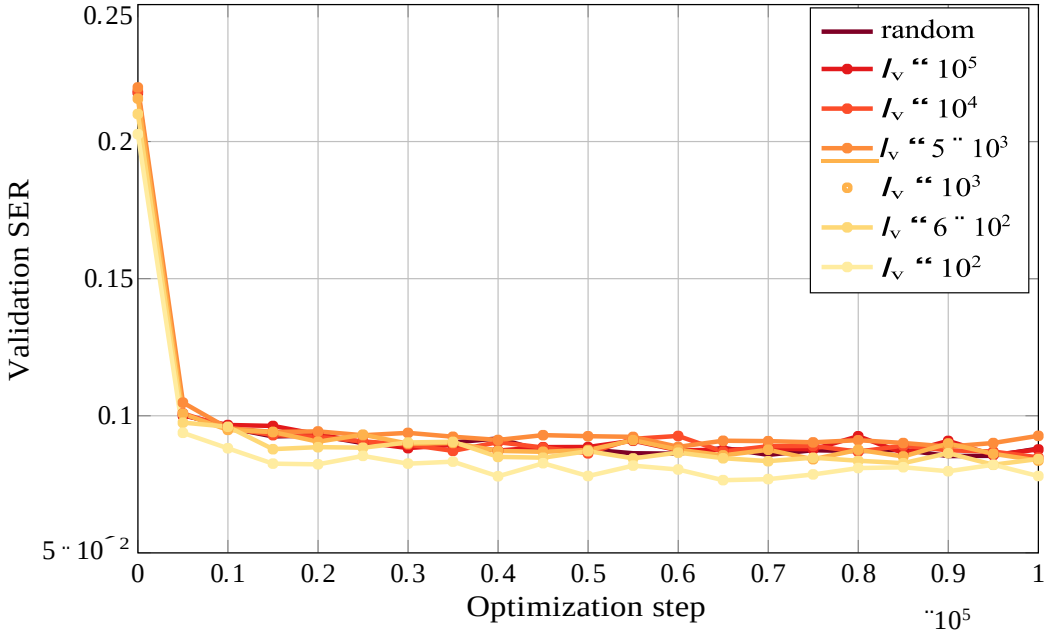

Figure 6.6: Validation symbol error rate as a function of optimization step for different lengths l_v of the validation pattern. The receiver ANN of the system is retrained using data from the transmission of training sequences with a long repetition pattern $l_{tr} = 10^4$.

the channel. Two sets of sequences are used – for re-optimization of the receiver ANN parameters as well as for performance validation. In particular, the FFNN auto-encoder described in Chapter 4 was used as a reference system. It was trained, following the procedure in Sec. 4.4.2.2, on a model which included only transmitter LPF, fiber dispersion and square-law detection. The optimized transceiver was then deployed for test transmission, simulating the full optical IM/DD channel model with low-pass filtering at transmitter and receiver, noise from the DAC, the ADC and the electrical amplification circuit, nonlinearity from the MZM and the squarelaw detection and fiber dispersion.

Figure 6.5 shows the scenario when the length of the training pattern l_{tr} is set to 600 messages, which are repeated in a long transmitted sequence, e.g. 10^6 messages. The figure shows the symbol error rate as a function of the receiver optimization step for validation sets with different pattern lengths l_v. Note that the case where new validation messages are continuously generated is referred to as *random*. After an initial reduction, the SER in all validation scenarios starts to increase as optimization progresses. Such a behaviour can be attributed to the fact that the receiver is optimized using a very short repeated pattern to which the ANN parameters are overfitted.

In order to investigate this effect further, Fig. 6.6 shows the obtained SER

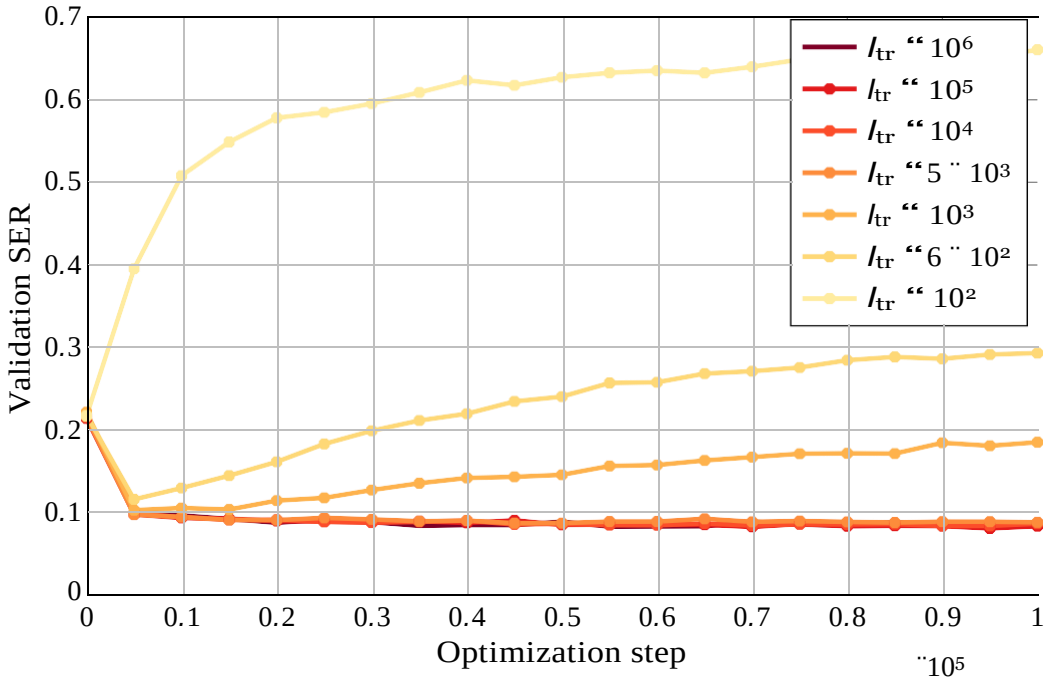

Figure 6.7: Validation symbol error rate as a function of optimization step for continuously generated random validation messages and different lengths l_{tr} of the message pattern used for receiver optimization.

from validations with different pattern lengths versus optimization step for the case where the receiver was re-trained using data from sequences with the significantly longer pattern length of $l_{tr} = 10^4$. In such a scenario, the SER for all validations decreases during optimization. Moreover, apart from the case of $l_v = 10^2$ (short validation pattern), a convergence of the obtained SERs can be observed at each step. The results indicate that the pattern of $l_{tr} = 10^4$, utilized in the training process is sufficiently, allowing for generalization of the ANN parameters, evidenced by the validation convergence.

Finally, Figure 6.7 investigated the effects of the training pattern length on the optimization performance when the validation is random. Convergence in the validation symbol error rate is observed for $l_{tr} \geq 5 \cdot 10^3$. The presented results were used as general guidelines for transmission experiments and in particular for the collection of measured data used for optimization of the ANN parameters.

6.4 Experimental demonstration of the FFNN-based auto-encoder

Following the simulations presented in Chapter 4, the performance of the optical transmission system based on end-to-end FFNN was validated using the experimen-

tal IM/DD test-bed. For this experiment, the data rate of the system was 42 Gb/s. The transmitted waveforms were obtained by feeding a random sequence of messages m_t (from an alphabet of M = 64) to the transmitter FFNN, which transforms it into the sequence of encoded symbols $\mathbf{x}_t$, with $\mathbf{x}_t \in \mathbb{R}^{n \cdot s}$. The symbol sequence was digitally filtered by the LPF, which had a bandwidth of 32 GHz, down-sampled and applied to the DAC. In the experiment, downsampling by a factor of s = 4 was performed on the resulting filtered concatenated series of symbols, each now containing n = 12 samples. Because of the LPF, there is no loss of information, since the original series of symbols, at $n \cdot s$ = 48 samples each and running at 336 GSa/s, can be exactly regenerated from this downsampled series of symbols, n = 12 samples per symbol at 84 GSa/s. The obtained electrical waveform was used to modulate the MZM and applied to the rest of the setup, described in Sec. 6.1. The digitised received waveforms were fed symbol-by-symbol to the receiver FFNN.

In this experiment, $40 \cdot 10^6$ symbols (blocks of samples) were transmitted and received for each distance. This was achieved by transmitting 1000 sequences of $40 \cdot 10^3$ messages. The performance of the system was investigated in two scenarios: Firstly, the error rate of the system, calculated between the experimentally transmitted and received messages (see Fig. 6.4), was studied for transceiver optimized in simulation-only and applied "as is" to the actual transmission link. The optimization procedure, the simulation setup, and the computation of the BER were performed as described in Sec. 4.3 and Sec. 4.4.

Secondly, the performance enhancement of the auto-encoder, achieved by optimizing the receiver ANN parameters with data from the measurements was also investigated. In particular, for each of the considered distances, a set of $|S| = 30 \cdot 10^6$ (75% of all traces) received symbols was used for optimization, while validation during the process was performed with a set of $|S_\text{v}| = 5 \cdot 10^6$ (12.5% of all traces) different received blocks of samples (from different measurements). To obtain the BER performance of the system after receiver re-optimization, testing was performed using the remaining $|S_\text{test}| = 5 \cdot 10^6$ (12.5% of all traces) data, which was *not* used for training and validation.

6.4.1 Experimental performance results

The bit error rate performance of the 42Gb/s FFNN auto-encoder was investigated for the distances of 20, 40, 60, 80 km and the obtained results are presented in Fig. 6.8. For this scenario the transceiver was optimized in simulation using the IM/DD transmission model from Chapter 3. At distance of 20 km, the BER of the system was $1.1 \cdot 10^{-3}$, which is below the 6.7% HD-FEC threshold used as a performance indicator. The error rate was $9.8 \cdot 10^{-2}$ at 40 km and increased further

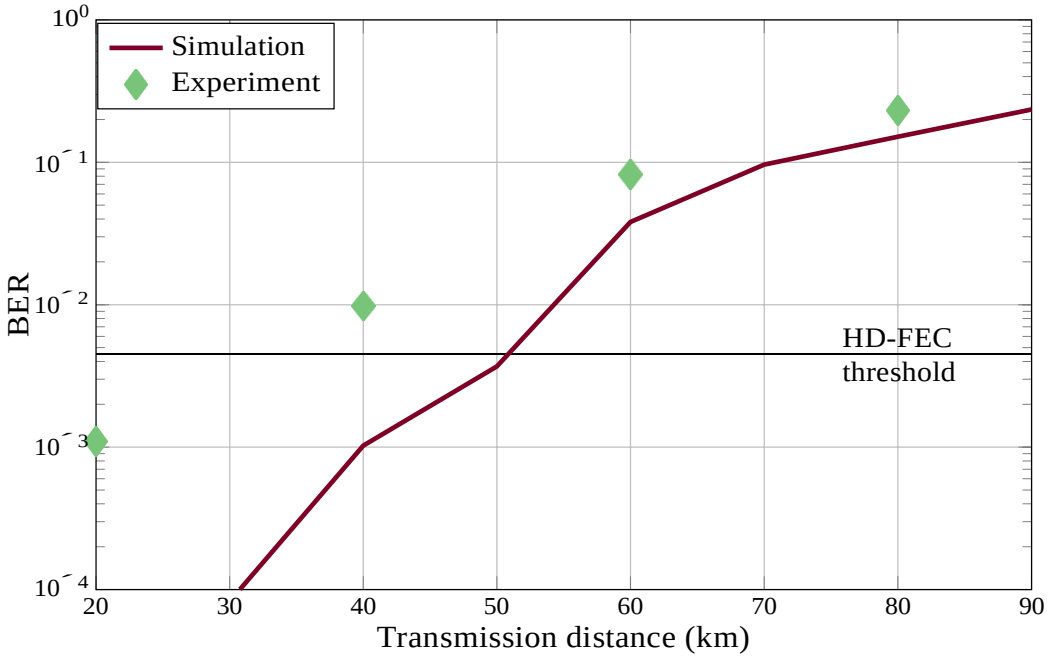

Figure 6.8: Experimental BER performance at 20, 40, 60, 80 km of the FFNN-based autoencoder trained on an explicit transmission model and applied "as is" to the optical test-bed.

at 60 km and 80 km. The results showed that the FFNN auto-encoder optimized only in simulation cannot achieve reliable communication below the HD-FEC at distances longer than 20 km. Moreover, the figure indicates a substantial penalty in BER compared to simulation. It was explained in Sec. 6.2 of this chapter, that the performance deteriorates as a consequence of any discrepancies between the channel model used in simulation and the actual experimental link. For example, these can be stemming from the reduced accuracy of the waveform generation process due to limited DAC resolution or because of distortions, which were not accounted for in simulation, e.g. from the TDM module.

For improving the performance, receiver training using the measured data was considered. To study the effects of re-optimization at different distances, the FFNN auto-encoder was trained using the multi-distance learning method (see Sec. 4.3.2) for this investigation. More specifically, receiver optimization was carried out in two different ways, the results of which are shown in Fig. 6.9. In the first approach, denoted "fine-tuning", the receiver ANN parameters were initialised with the values previously obtained in simulation and then the re-training is carried out using the labeled experimental samples. In the second approach, denoted "randomization", the receiver ANN is assigned with randomly initialised parameters, for example sampled from a truncated normal distribution (similar to the initialisation stage in

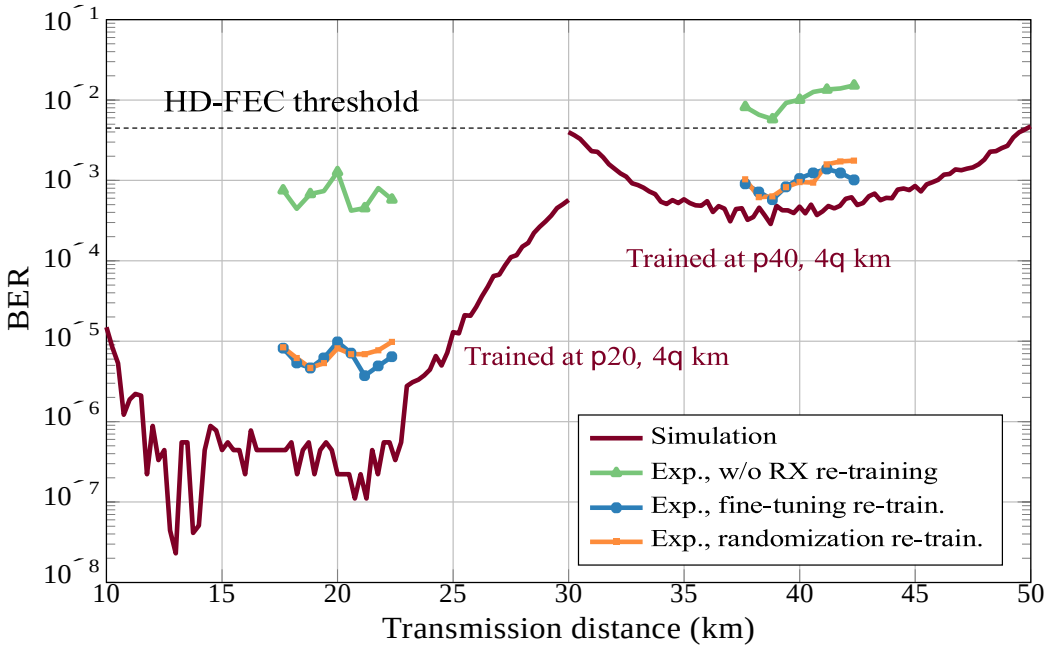

Figure 6.9: Comparison of the experimental BER performance for systems trained at (20, 4) km and (40, 4) km (i) without re-training of the receiver FFNN, (ii) retraining the receiver FFNN by fine-tuning, (iii) training the receiver FFNN by randomization.

simulation). The experimental BER curves at 20 and 40 km for the two re-training approaches are compared with the case of no-retraining. Note that the different link lengths were achieved by sweeping the disperion in the TDM module. It can be observed that accounting for the difference between the real experimental environment and the assumed channel model by re-training the receiver improves performance at both 20 km and 40 km distances. Moreover, the results confirmed that both receiver re-optimization strategies converge to approximately the same BER values for all investigated distances. It is worth mentioning that, although the number of training iterations for the two approaches was kept equal, initialising the receiver ANN parameters with pre-trained values had the advantage of requiring less iterations to converge.

It is worth noting that the obtained results without re-training indicate that, due to the implemented multi-distance optimization, the BER of the system does not increase rapidly when the distance is varied. Further verification of the method applied to the receiver re-training is presented in Sec. 6.4.3. Before this, the performance of the auto-encoder with receiver optimized via “fine-tuning” is investigated in a comparison with conventional, ubiquitously deployed IM/DD systems.

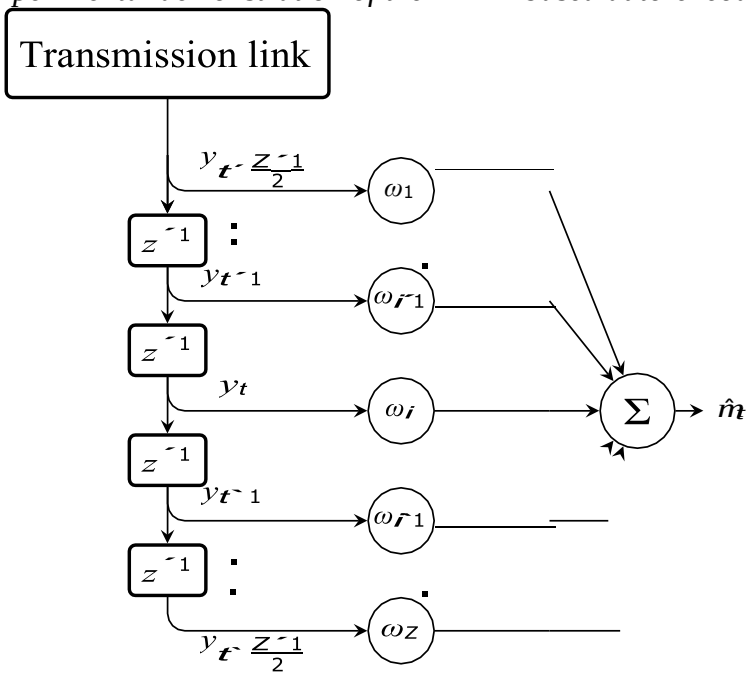

Figure 6.10: Schematic of a conventional feedforward equaliser, used in the reference PAM transmission for experimental performance comparison.

6.4.2 Performance comparison with conventional systems

In this section, the BER performance of the end-to-end FFNN-based system was compared to conventional schemes based on PAM transmission with linear equalisation at the receiver. The data rate of the considered PAM systems was also set to 42 Gb/s. In particular, the first system employed 42 Gbaud PAM2 transmission and raised cosine pulses (roll-off of 0.99). The second system was operating at 21 Gbaud with PAM4 and raised cosine pulses (roll-off of 0.4). The receiver DSP for both schemes employed the conventional feedforward equaliser (FFE), whose schematic is shown in Fig. 6.10. It processes $(Z-1)/2$ pre and post-cursor symbols for the equalisation of the central PAM symbol. The implementation was based on $Z=13$ taps, at $n=2$ taps per PAM symbol, ensuring that the receiver covers a temporal window similar to the auto-encoder. It is important to mention that the noise power of 0.245 mW obtained from a measurement at the receiver was used for the noise model in Chapter 4 as well as in the simulations for the systems described in Chapter 4 and Chapter 5.

Figure 6.11 shows the experimental BER results for the compared systems for a fiber of length 20 km and 40 km, respectively. The TDM dispersion value was swept between −40 ps and +40 ps, resulting in effective link distances in the ranges of 17.65 – 22.35 km and 37.65 – 42.35 km, respectively. For the system around 20 km, BERs below 10^{-5} were achieved experimentally at all distances. In

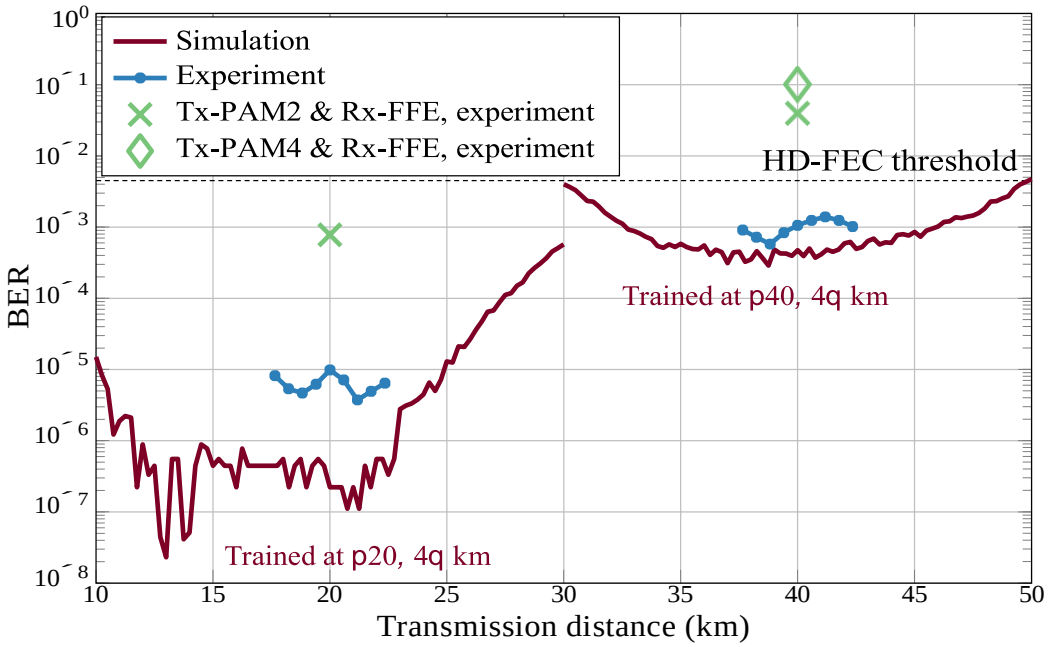

Figure 6.11: Experimental BER performance for systems trained at (20, 4) km and (40, 4) km. The systems are compared to PAM2 and PAM4 systems with receivers based on conventional feedforward equalisation.

particular, the lowest BER of $3.73 \cdot 10^{-6}$ was obtained at 21.18 km. For comparison, the PAM2 system experimentally achieves $7.77 \cdot 10^{-4}$ BER at 20 km. The endto-end deep learning-based optical system significantly outperforms the reference PAM scheme. At 40 km, the proposed system outperforms both the 42 Gbaud PAM2 and the 21 Gbaud PAM4 schemes, as neither of these can achieve BERs below the HD-FEC threshold. On the other hand, the FFNN-based system achieved BERs below $1.4 \cdot 10^{-3}$ at all distances in the examined range. In particular, BERs of $1.05 \cdot 10^{-3}$ at 40 km and a lowest BER of $5.75 \cdot 10^{-4}$ at 38.82 km have been obtained.

Figure 6.12 shows the experimental results at 60 km and 80 km fiber length and TDM dispersion varied between −40 ps and +40 ps, yielding effective link distances in the ranges 57.65 – 62.35 km and 77.65 – 82.35 km, respectively. For both systems BERs below the HD-FEC threshold cannot be achieved by the end-toend deep learning approach, as predicted by the simulation. Nevertheless, at 60 km the system outperforms both the PAM2 and PAM4 links. However, for the 80 km link, the noise at the receiver becomes more significant due to the low signal power levels without optical amplification. This is combined with a significant amount of accumulated dispersion, whose effects at 80 km extend across multiple blocks and cannot be compensated by the block-by-block processing, accommodated in the FFNN design. The results is an operation close to the sensitivity limits of the

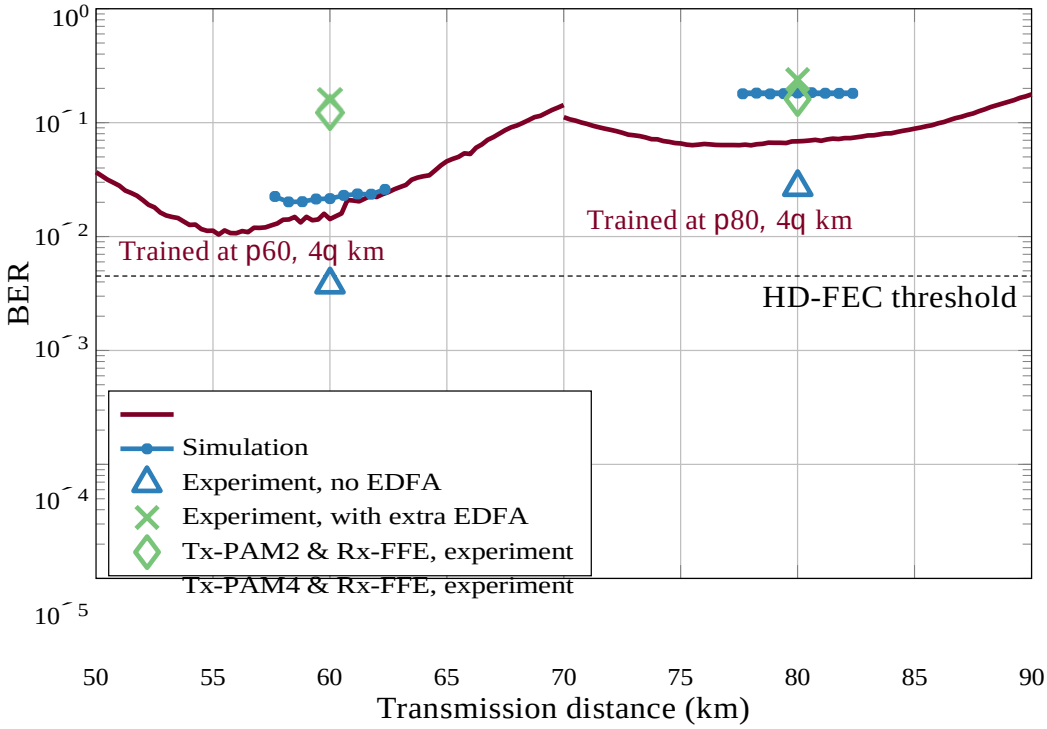

Figure 6.12: Experimental BER performance for systems trained at (60, 4) km and (80, 4) km. The systems are compared to PAM2 and PAM4 systems with receivers based on conventional feedforward equalisation.

receiver which ultimately restricts the achievable BERs.

To further investigate the impact of received signal power on the performance of the system, an erbium-doped fiber amplifier (EDFA) was included in the test-bed for pre-amplification at the receiver. Thereby, the received power is increased from
-11 and -15 dBm at 60 km and 80 km, respectively to -7 dBm. The obtained BERs at these distances are also shown in Fig. 6.12. It can be seen that by changing the link to include an extra EDFA, the end-to-end deep learning system achieves significantly improved performance. In particular, at 60 km, a BER of $3.8 \cdot 10^{-3}$, below the HD-FEC threshold, can be achieved. Nevertheless, the performance of the system at 80 km is still limited and the measured BER is $2.8 \cdot 10^{-2}$. The last set of results highlight the potential for performance improvement by including different link configurations inside the end-to-end learning process.

6.4.3 Distance-agnostic transceiver

This section describes the experimental verification of the *multi-distance learning* method developed in Sec. 4.3.2. The investigation covers the optimization of receiver parameters using experimentally collected received traces with deviating dispersion, set by the tunable dispersion module (TDM). More specifically, the data was acquired by employing fixed fiber spools of 20 km and 40 km and using the TDM to appropriately sweep the dispersion around these mean values. As previ-

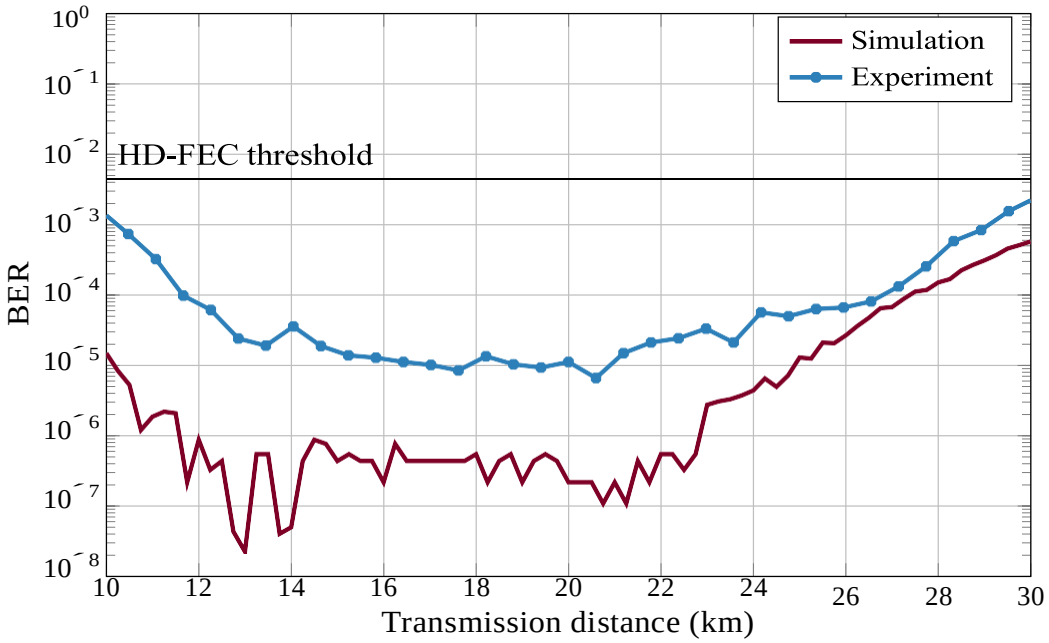

Figure 6.13: Experimental BER performance as a function of transmission distance between 10 and 30 km for a (20, 4) km-trained system, whose receiver ANN parameters are tuned using measured data and the multi-distance learning method proposed in Sec. 4.3.2.

ously explained (see Sec. 6.1), the TDM allowed to adjust the link dispersion from −170 to +170 ps/nm in steps of 10 ps/nm. The digitized received photocurrent is collected for each effective distance. The amount of data collected at each point follows a Gaussian distribution with a mean $\mu = 20$ km or $\mu = 40$ km and a standard deviation of 67 ps/nm, which effectively corresponds to around 4 km of SSMF (matching the simulation assumption). The system with optimized receiver ANN parameters was tested using a disjoint collection of traces at each distance.

Figure 6.13 shows the achieved BER performance at 42 Gb/s around the mean distance of 20 km. In agreement with simulation, the experimental system reached BER below the 6.7% HD-FEC threshold for all distances between 10 and 30 km. Moreover, it can be observed that the lowest BER was achieved around 20 km as a result of the employed strategy of training dispersion values with a Gaussian distribution, i.e. the number of training examples around the mean value was greater. In Fig. 6.14 the BER performance of the system trained around 40 km is examined at distances ranging from 30 to 50 km. Moreover, the results are compared to the performance obtained for the conventional PAM2/PAM4 42 Gb/s transmission based on a FFE receiver, which was described in Sec. 6.4.1. Note that, unlike the examined system based on multi-distance learning, the FFE receiver for PAM2/PAM4

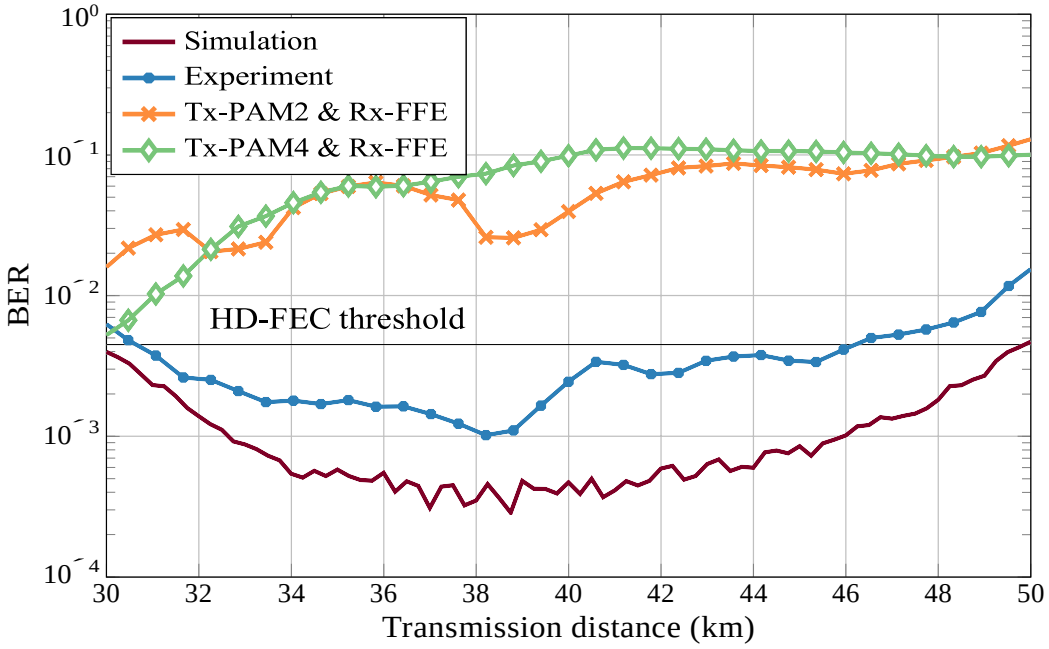

Figure 6.14: Experimental BER performance as a function of transmission distance between 30 and 50 km for a 42 Gb/s system trained on (40, 4) km, whose receiver ANN parameters are tuned using measured data and the multi-distance learning method proposed in Sec. 4.3.2. The BER of PAM2 and PAM4 schemes with an FFE receiver is shown as a performance indicator. Note that the parameters of the FFE receiver are optimized separately for each distance.

needs to be optimized separately for each fiber length. The auto-encoder system enabled BERs below the HD-FEC for all distances between 31 and 46 km. Furthermore, it significantly outperformed the reference PAM setups, neither of which could achieve BER below the threshold at the examined distances. It can be seen that the performance of the PAM2 system oscillates as a function of distance, caused by the power fading profile of the signal experiencing different dispersions [120]. In contrast, the ANN-based system, specifically trained to tolerate dispersion variations, does not present such a strong dependence on the latter.

The results from the investigation successfully demonstrate the greater robustness to link parameter variations achieved for the deep learning-based transceivers optimized using the proposed multi-distance method.

6.5 Experimental demonstration of the SBRNN autoencoder

This section describes the implementation and investigates the performance of the SBRNN auto-encoder proposed in Chapter 5 on the experimental IM/DD test-bed.

The performance of the system is investigated when the transceiver is optimized on a transmission model with parameters applied "as is" to the test-bed as well as in the case where receiver optimization using the measured data is performed. The investigation covers comparisons with PAM systems based on receivers optimized for handling nonlinear ISI using state-of-the-art deep learning techniques as well as the classical nonlinear Volterra equalisation.

For the SBRNN auto-encoder experiment, $Z = 800$ different sequences of $N = 5152$ random input messages $m_{i,\,j}$ (from an alphabet of $M = 64$ different messages), with $i \in \{1,\ldots, Z\}$ and $j \in \{1,\ldots, N\}$, are transmitted. Each of the sequences is first encoded by the transmitter BRNN into series of symbols of $n \cdot s$ samples before being applied to the experimental test-bed, starting with the LPF. Note that, similarly to the FFNN experiment, an up-sampling factor of $s = 4$ is used in both transmitter and receiver. Thus the series of encoded blocks are downsampled by s in experiment after the LPF at the transmitter and then up-sampled after the receiver LPF. The value of n dictates the information rate of the system and for 42 Gb/s or 84 Gb/s transmission, $n = 12$ or $n = 6$ was selected, respectively. A full load of the DAC required the concatenation of 8 encoded sequences of symbols and this is fed to the link for a single transmission iteration. To collect a sufficient amount of data, the experiment involved a total of 100 DAC loads. Similarly to the investigation of the FFNN auto-encoder, the performance of the SBRNN system was investigated in two scenarios:

First, the error rate between the experimentally transmitted and received messages (see Fig. 6.4), was investigated for transceiver optimized in simulation-only and applied "as is" to the actual transmission link. The optimization procedure, the simulation setup, and the BER computation were performed as described in Sec. 5.1 and Sec. 5.2. All examined systems in this section were trained separately for each distance.

Then, the system performance was enhanced by re-training the receiver parameters with the experimental data. For this optimization and its testing stage, the sequences of digitised received symbols $\mathbf{y}_{i,j}$, with $i \in \{1,\ldots, Z\}$ and $j \in \{1,\ldots, N\}$, were used together with their corresponding labels $m_{i,j}$. The sequences with indices $i \in \{1,\ldots, 720\}$ were used for training. For convenience, these were organized in containers of data elements $\mathbf{D}^{\mathrm{AE}}_{\mathrm{train}}$ and corresponding label elements $\mathbf{L}^{\mathrm{AE}}_{\mathrm{train}}$. The remaining sequences with indices $i \in \{721,\ldots, Z\}$, which are independent from the training set, were used for testing, organized in $\mathbf{D}^{\mathrm{AE}}_{\mathrm{test}}$ and $\mathbf{L}^{\mathrm{AE}}_{\mathrm{test}}$. The construction and utilization procedure of $\mathbf{D}^{\mathrm{AE}}_{\mathrm{train}}$, $\mathbf{L}^{\mathrm{AE}}_{\mathrm{train}}$, $\mathbf{D}^{\mathrm{AE}}_{\mathrm{test}}$, and $\mathbf{L}^{\mathrm{AE}}_{\mathrm{test}}$ is described in Appendix C.1.

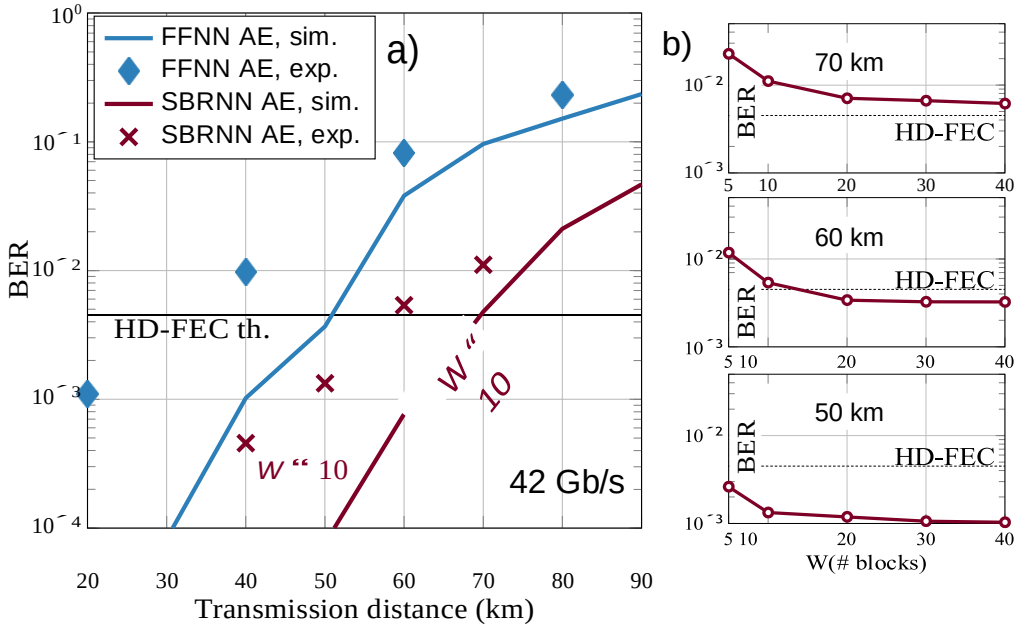

Figure 6.15: a) Experimental BER performance at 20, 40, 60, 80 km for the 42 Gb/s SBRNN-based auto-encoder trained on a channel model and applied "as is" to the optical fiber transmission test-bed. The BER of the end-to-end FFNN system is also shown as a performance indicator. b) BERs at 50, 60 and 70 km as a function of the sliding window size W for the sequence estimation algorithm at the SBRNN receiver.

6.5.1 Experimental performance results

The conducted numerical investigation in Chapter 5 clearly showed that in the framework of optical IM/DD communication, the processing capabilities of the vanilla BRNN auto-encoder are comparable to the long short-term memory structure, without the associated extra complexity. For this reason, the vanilla SBRNN auto-encoder was implemented on the experimental test-bed.

First, the transceiver optimized on the optical IM/DD channel model was applied "as is" to the experimental test-bed. Initially, the fixed sliding window size of $W = 10$ was employed. The achieved BER performance at 42 Gb/s as a function of distance is shown in Fig. 6.15 a), where it was compared to the end-to-end FFNN system, investigated in Sec. 6.4. It can be seen that, in good agreement with simulation, the SBRNN system significantly outperformed the FFNN auto-enocoder. In particular, it allowed transmission below the 6.7% HD-FEC at distances beyond 50 km in experiment, a 30 km increase compared to the FFNN.

The conducted investigation in Sec. 5.2.2 showed that the performance of the SBRNN auto-encoder can be improved by increasing the window W, which dictates the amount of received symbols used for the sequence estimation algorithm at the receiver. Greater W allows to effectively capture intersymbol interference effects at

longer distances. The presented results in Fig. 6.15 b) investigate the experimental BER of the SBRNN auto-encoder at 50, 60 and 70 km as a function of the sliding window W. It can be seen that, as a result of the increased window size, the BER of the system was reduced below the HD-FEC threshold at the distance of 60 km. More specifically, the error rate was below $4 \cdot 10^{-3}$ for $W \geq 20$. It is important to mention that, in an excellent agreement with the simulation prediction in Sec. 5.2.2, the experiment showed diminishing gains from further increase of W since noise and system nonlinearities become more dominant when the ISI is compensated.

The approach of enhancing the SBRNN auto-encoder system by using the collected experimental data for re-optimization of the receiver parameters was considered next. The performance of the SBRNN auto-encoder, whose receiver was optimized with the measured data, was compared to multi-level PAM systems employing state-of-the-art DSP schemes at the receiver for nonlinear ISI compensation. The processing and optimization procedures for these reference systems are described in the following section, while Sec. 6.5.3 presents the outcome of the performance comparison with the auto-encoder.

6.5.2 Reference PAM systems with state-of-the-art receiver DSP

The conventional twoand four-level PAM transmission was considered and the performance of deep learning approaches for PAM detection based on recurrent and feed-forward neural networks as well as the classical nonlinear Volterra equalisation were examined. For all three schemes, the experimental data was used for optimization of the receiver parameters. The PAM experiment was conducted in the following way: At the transmitter, PAM2 or PAM4 symbols were generated and accordingly scaled to sets $M = \{0; \pi/4\}$ or $M = \{0; \pi/12; \pi/6; \pi/4\}$, respectively. Note that, similarly to the auto-encoder systems where a clipping layer was specifically designed, scaling of the PAM symbols was necessary to enable operation in the linear region of the MZM. Also note that Gray-coded bit-to-symbol mapping was considered in the case of PAM4 in order to investigate the BER metric as an indicator of the system performance. The sequence of PAM symbols was pulse-shaped at $n = 2$ Sa/sym by a 0.25 roll-off raised cosine filter. It was then applied to the experimental test-bed. The experiment involved $Z = 100$ loads of the DAC, each of them equivalent to a sequence of $N = 249750$ PAM symbols $m_{i,j}$, $i \in \{1,\ldots, Z\}$ and $j \in \{1,\ldots, N\}$. The sequences of received symbols $\mathbf{y}_{i,j} \in \mathrm{R}^n$ after propagation through the test-bed link were utilized together with the labels $m_{i,j}$ for optimization of the DSP parameters and testing. Sequences $i \in \{1,\ldots,90\}$ were used for training, organized in $\mathbf{D}^{\mathrm{PAM}}$ and $\mathbf{L}^{\mathrm{PAM}}$ for traces and labels, respectively.

The testing was carried out using the disjoint set of remaining sequences with in-

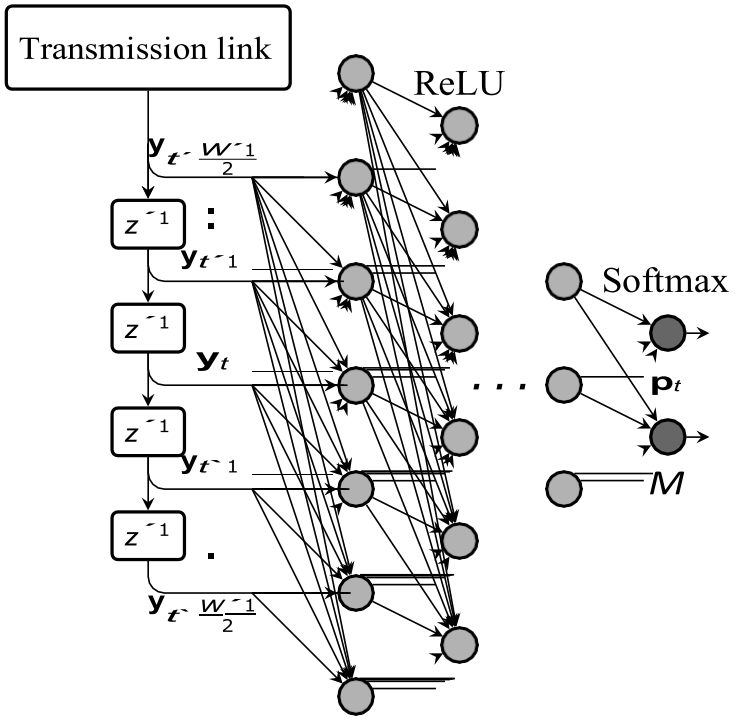

Figure 6.16: Schematic of the reference sliding window FFNN receiver for PAM symbols, utilized in the experiment.

dices $i \in \{91, \ldots, 100\}$, organized in $\mathbf{D}_{\text{tes}}^{\text{PAM}}$ and $\mathbf{L}_{\text{tes}}^{\text{PAM}}$. The database structuring is described in Appendix C.2, where the details regarding the data utilization in the optimization of the different PAM receiver schemes are also provided. The three considered schemes are described in the following.

6.5.2.1 Sliding Window FFNN receiver

In this scheme, already introduced and examined in simulation in Sec. 5.3.1, a PAM sequence is transmitted through the link and the samples of an appropriately chosen sub-sequence of received symbols are fed into a multi-layer feed-forward ANN, as shown in Fig. 6.16. It was discussed in Sec. 5.3.1 that a similar approach for the receiver design has been considered for example in [58, 113–115]. The neural network estimates the central symbol of the sub-sequence. More specifically, the transmitted message m_t at time t is recovered as $\hat{m}_t$ by processing the train of W

received symbols of $n = 2$ samples $(\mathbf{y}_{t-\frac{W-1}{2}}, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_{t+\frac{W-1}{2}})$. Note that for this receiver, the processing memory dictates the size of the ANN layers, the first one having parameters $\mathbf{W} \in \mathbb{R}^{4W \cdot n \times W \cdot n}$ and $\mathbf{b} \in \mathbb{R}^{4W \cdot n}$. Subsequently, 6 hidden layers are employed before $\hat{m}_t$ is estimated. The number of nodes on each of these is given by $4W \cdot n / (2^{i-1})$, where i is the hidden layer index. The ReLU activation is applied on all of the layers except for the final one, where a softmax activation allows to compute a probability vector $\mathbf{p}_t$ for the transmitted PAM2/4 symbol with

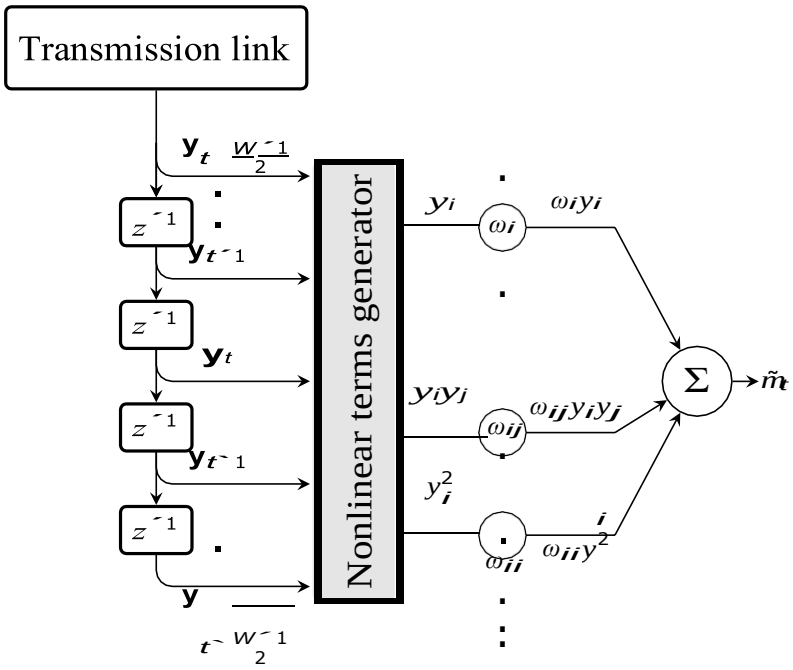

Figure 6.17: Schematic of the second-order nonlinear Volterra equaliser for PAM symbols, used as a reference system in the experiment.

$\mathbf{p}_t \in \mathrm{R}^{2/4}$. The processing window slides by one received symbol position ahead to estimate the next symbol in the sequence. Training was performed as described in Appendix C.2.1 using $\mathbf{D}^{\mathrm{PAM}}_{\mathrm{train}}$ and $\mathbf{L}^{\mathrm{PAM}}_{\mathrm{train}}$

6.5.2.2 Nonlinear Volterra Equaliser

Volterra equalisers, shown schematically in Fig. 6.17, have been widely considered for IM/DD links based on PAM [66]. In particular, for the experimental investigation in this chapter, a second order Volterra filter was used, allowing the compensation of linear and nonlinear ISI. The output of the equaliser can be expressed as

$$\hat{m}_t = \omega_{dc} + \sum_{q_1=-\frac{W-1}{2}\cdot n}^{\frac{W-1}{2}\cdot n} \omega_{q_1} \cdot y_{t+q_1} + \sum_{q_2=-\frac{W_1-1}{2}\cdot n}^{\frac{W_1-1}{2}\cdot n} \sum_{q_3=k_2}^{\frac{W_1-1}{2}\cdot n} \omega_{q_2,q_3} \cdot y_{t+q_2} \cdot y_{t+q_3}, \qquad (6.2)$$

where W and W_1 denote the symbol memory in the first and second order series, respectively, and ω_i and $\omega_{i,j}$ are the first and second order coefficients. Similar to the other PAM schemes, the algorithm is processing the neighbourhood of W received symbols of $n = 2$ samples $\mathbf{y}_{t-\frac{W-1}{2}}, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_{t+\frac{W-1}{2}}$, thus including the ISI from pre- and post-cursor samples in the detection of the current input. To capture an identical amount of memory compared to the auto-encoder and the SFFNN receiver, the first order memory of the Volterra equaliser was fixed to $W = 61$, while $W_1 = 21$ was chosen such that the complexity of the algorithm was kept manageable. The

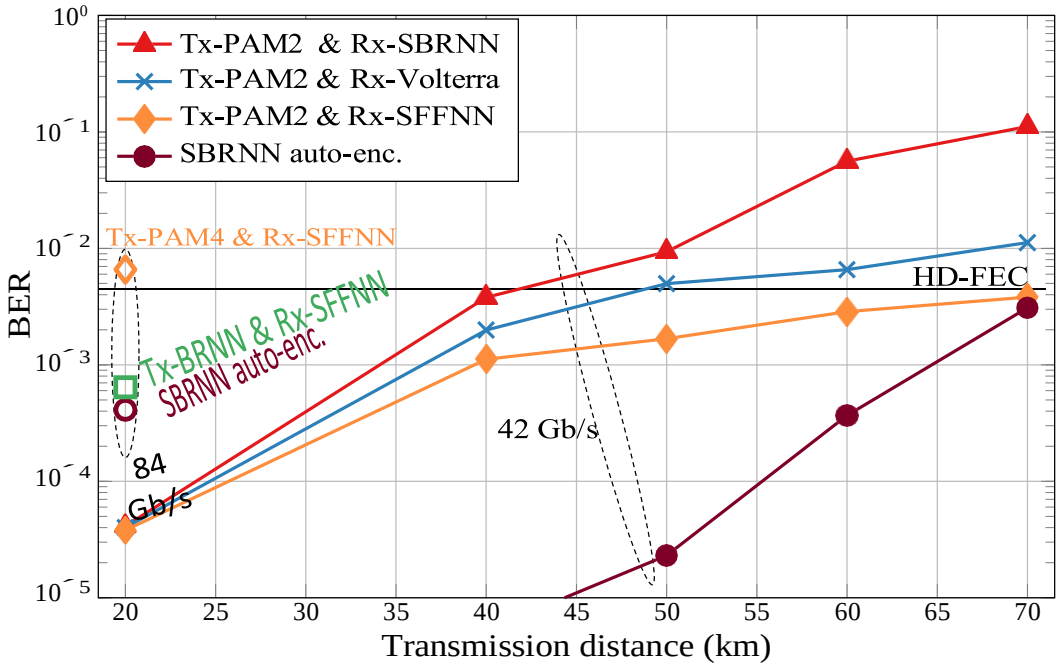

Figure 6.18: BER as a function of transmission distance for 42 Gb/s and 84 Gb/s systems employing deep learning-based and classical DSP schemes optimized using experimental data.

optimization of the filter coefficients using is described in Appendix C.2.2.

6.5.2.3 Sliding Window BRNN receiver

A system that combines the PAM transmitter with the SBRNN receiver was also investigated. The receiver was implemented in a way which was identical to the receiver section of the SBRNN auto-encoder. As opposed to the auto-encoder, for the optimization of the SBRNN receiver for PAM parameters the databases $\mathbf{D}^{\text{PAM}}$ and $\mathbf{L}^{\text{PAM}}$ were used. Details on the training procedure are provided in Appendix C.2.3.

6.5.3 Experimental performance comparison between the SBRNN auto-encoder and the reference PAM systems

For this investigation, the number of bits simultaneously processed by each receiver algorithm was fixed. The setting was aimed at carrying out a fair comparison between the three different PAM DSP schemes with the SBRNN auto-encoder. Thus, for the auto-encoder, the size of the sliding window was set to $W = 10$ (60 bits) to match the simulation investigation, while for the PAM2 systems $W = 61$ (61 bits). To allow for a better compensation in the case of PAM4, the window of $W = 61$ PAM symbols was kept, which was the equivalent of processing 122 bits.

Figure 6.18 shows the achieved BER for all experimentally examined systems at different transmission distances. As previously mentioned, the DSP for these

systems was optimized separately for each distance. The obtained results show that for shorter distances and 42 Gb/s, the SBRNN auto-encoder achieved a BER much lower than the PAM schemes. In terms of system reach, it allowed transmission up to 70 km below the HD-FEC threshold, similar to the PAM2&SFFNN. Both schemes outperformed the classical Volterra equaliser, whose experimental BER was below the HD-FEC up the transmission distance of 40 km. Hence, the results from this experimental investigation indicate that the SFFNN can be considered as a viable receiver DSP solution for the conventional PAM transmission.

For further investigation, this receiver was used together with PAM4 in a 84 Gb/s system. Moreover, the SFFNN receiver was combined in an auto-encoder setting with the BRNN transmitter. The results show that the PAM4&SFFNN could not transmit information with error rates below the HD-FEC at 20 km, while the TxBRNN&Rx-SFFNN auto-encoder performance was significantly superior at this distance, achieving a BER close to the SBRNN auto-encoder system. Nevertheless, it is worth mentioning that, unlike the SBRNN, the number of parameters in the SFFNN receiver increases rapidly with the processing memory. Expanding the SFFNN processing memory W translates in increasing the dimension of the neural network input layer and subsequently – the following hidden layers, which would result in a rapid increase in the number of trainable parameters. In contrast, for the BRNN auto-encoder, the window W is external to the end-to-end network architecture, whose number of parameters can be fixed. A more detailed computational complexity comparison between the SBRNN and SFFNN designs was carried out and described in Sec. 5.3.1.3.

6.6 End-to-end system optimization using a generative model

This section describes the concept and experimental demonstration of the developed end-to-end optical fiber system optimization method using a GAN-acquired model for the transmission link. The proposed algorithm, which was briefly introduced in 6.2, consists in performing iterative steps of training the generative model with experimental data and using it to optimize the transceiver. As a proof of concept, a simple FFNN auto-encoder is considered. Table 6.1 lists the hyper-parameters for both transmitter and receiver ANNs. As described for FFNN auto-encoders (see Sec.4), the transmitter maps the input messages $m_t \in \{1,\ldots, M\}$, each of which carrying $\log_2(M)$ bits, into blocks (symbols) of n samples, which are denoted as
$\mathbf{x}_t \in \mathrm{R}^n$. After propagation through the channel, the symbols are fed to the receiver as $\mathbf{y}_t \in \mathrm{R}^n$. Generative adversarial networks (GAN) were introduced in Sec. 2.1.4.

Table 6.1: Definitions of the transmitter and receiver FFNN used for the experimental verification of the end-to-end system optimization method using a generative model

	Layer	Activation	Output dimension
Transmitter:			
	Input	one-hot	M
	Hidden 1	enc.	$8M$
	Hidden 2	ReLU	$8M$
	Final	ReLU	n
		Clipping	
Receiver:			
	Input	N/A	n
	Hidden 1	ReLU	$8M$
	Hidden 2	ReLU	$8M$
	Final	Softmax	M

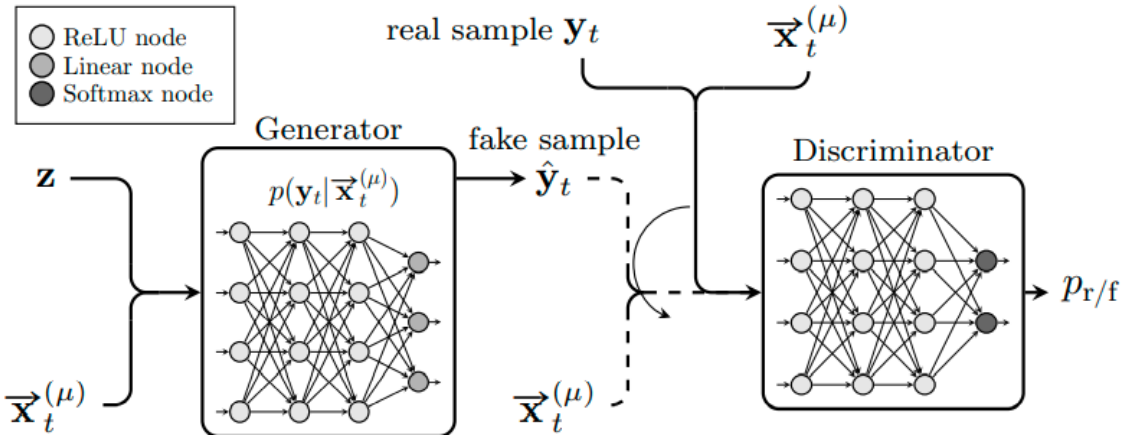

Figure 6.19: Schematic of the utilized conditional GAN for approximating the function of the IM/DD transmission link.

In the following, the specific design of the GAN that was used in the experimental investigation is introduced. Then, the transmission and optimization regimes in the proposed method are described and the investigation of the achieved performance improvement is presented.

6.6.1 Generative adversarial network design

As explained in Sec. 2.1.4, a GAN typically employs two ANNs – a generator and a discriminator. They have competing objective functions and are trained iteratively. The generator ANN aims at translating its input into a (possibly high-dimensional) output sample, mimicking the ground truth distribution of the data, which is specific to the problem. The discriminator acts as a binary classifier between real (ground truth) and fake (generated) samples. Related to communication systems, GANs can

Table 6.2: Definitions of the generator and discriminator ANNs in the conditional GAN

	Layer	Activation	Output dimension
Generator:			
	Input	concat.	$2\mu \cdot n$
	Hidden 1	ReLU	$30n$
	Hidden 2	ReLU	$20n$
	Hidden 3	ReLU	$13n$
	Hidden 4	ReLU	$8n$
	Hidden 5	ReLU	$5n$
	Final	Linear	n
Discriminator:			
	Input	concat.	$(\mu + 1) \cdot n$
	Hidden 1	ReLU	$16n$
	Hidden 2	ReLU	$10n$
	Hidden 3	ReLU	$6n$
	Final	Sigmoid	1

be used to train a generative model (generator) which mimics the function of the channel by approximating its conditional probability distribution. In the framework of auto-encoder design, the model can be used in the end-to-end system optimization to enable to computation of gradients via backpropagation.

Since ISI in the optical IM/DD stems from both preceding and succeeding symbols, the goal of the generative model is to approximate $p(\mathbf{y}_t / \dot{\mathbf{x}}^{(\mu)})$, where μ, an
odd integer, is the modeled symbol memory and

$$\dot{\mathbf{x}}_t^{(\mu)} := \begin{bmatrix} \mathbf{x}^T_{t-(\mu-1)/2} & \cdots & \mathbf{x}^T_{t+(\mu-1)/2} \end{bmatrix}^T \in \mathbb{R}^{\mu \cdot n}, \tag{6.3}$$

is used to denote the concatenation of the preand post-cursor neighbourhood of μ symbols around the symbol $\mathbf{x}_t$ transmitted at time t. The schematic of the utilized conditional GAN for approximating $p(\mathbf{y}_t / \dot{\mathbf{x}}_t^{(\mu)})$ is shown in Fig. 6.19. To mimic the probability distribution, the generator takes as an input the concatenation of a vector of uniformly-distributed random samples $\mathbf{z} \sim U^{(0,1)}$ with $\dot{\mathbf{x}}^{(\mu)}$. It transforms it into the fake symbol:-

$$\hat{\mathbf{y}}_t = G_{\boldsymbol{\theta}_g}\left(\left(\mathbf{z}^T \quad \dot{\mathbf{x}}_t^{(\mu)T}\right)^T\right) \in \mathbb{R}^n, \tag{6.4}$$

where G_{ϑ_g} denotes the function of the generator, a feedforward ANN defined by the hyper-parameters listed in Table 6.2.

The discriminator is first fed with the real symbol $\mathbf{y}_t$, i.e. obtained in the ex-

periment, conditioned (concatenated) on $\vec{\mathbf{x}}_t^{(\mu)}$, producing the output probability

$$p_{\mathrm{r}} = D_{\theta_{\mathrm{d}}}\left(\begin{pmatrix}\mathbf{y}_t^T & \vec{\mathbf{x}}_t^{(\mu)T}\end{pmatrix}^T\right) \in \mathbb{R}, \tag{6.5}$$

where $D_{\vartheta_{\mathrm{d}}}$ describes the function of the discriminator – a feedforward ANN defined by the hyper-parameters listed in Table 6.2, which has a sigmoid activation in the final layer. The discriminator is then fed with the fake (output of the generator) symbol $\hat{\mathbf{y}}_t$, also concatenated with $\vec{\mathbf{x}}_t^{(\mu)}$, thus producing

$$p_{\mathrm{f}} = D_{\theta_{\mathrm{d}}}\left(\begin{pmatrix}\hat{\mathbf{y}}_t^T & \vec{\mathbf{x}}_t^{(\mu)T}\end{pmatrix}^T\right) \in \mathbb{R}. \tag{6.6}$$

Following the GAN theory (Sec. 2.1.4), the two sets of discriminator (ϑ_{d}) and generator (ϑ_{g}) ANN parameters are iteratively optimized via stochastic gradient descent (using the Adam algorithm). The procedure is aimed at minimising the average losses

$$\mathscr{L}_D(\boldsymbol{\theta}_{\mathrm{d}}) = \frac{1}{S}\sum_{i=1}^{S}[\ell(l_{\mathrm{r}}, p_{\mathrm{r},i}) + \ell(l_{\mathrm{f}}, p_{\mathrm{f},i})], \qquad \mathscr{L}_G(\boldsymbol{\theta}_{\mathrm{g}}) = \frac{1}{S}\sum_{i=1}^{S}\ell(l_{\mathrm{r}}, p_{\mathrm{f},i}), \tag{6.7}$$

over a mini-batch S of elements from the training set, where $l_{\mathrm{r}} := 1$ and $l_{\mathrm{f}} := 0$ are the labels for real and fake symbols, respectively. The cross entropy loss function is utilized. Note that the objective of the discriminator is to classify $\mathbf{y}_t$ and $\hat{\mathbf{y}}_t$ correctly. On the other hand, by minimising $\mathscr{L}_G$, the generator learns representations $\hat{\mathbf{y}}_t$ that are statistically similar to $\mathbf{y}_t$. The conditional GAN is trained over 10^4 steps used as a stopping criterion. Each of the steps consists in 4 consecutive discriminator updates with the learning rate of 10^{-3}, followed by a generator update with the rate gradually reduced every 200 steps from $5 \cdot 10^{-4}$ to 10^{-5}. After training, the generator ANN model is employed for end-to-end system optimization.

6.6.2 End-to-end optimization algorithm

The schematic of the experiment is shown in Fig. 6.20. The flow diagram in Fig. 6.21 shows the proposed algorithm for end-to-end system optimization. At an algorithm iteration $k = 0$, the transmitter and receiver ANNs are initialised with parameters trained offline, following Chapter 4, using the complete IM/DD channel model from Chapter 3. To collect a sufficient amount of experimental data, $Z = 500$ random sequences (full DAC loads) of $w = 8 \cdot 10^4$ messages $\ldots m_{t-1}, m_t, m_{t+1} \ldots \in \Sigma$, from an alphabet of $M = 8$ (3 bits), were generated and encoded by the transmitter into the sequences $\ldots \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \ldots$ of symbols of $n = 6$ samples. The symbol sequences were filtered by the 32 GHz LPF and applied to the 84 GSa/s DAC, result-

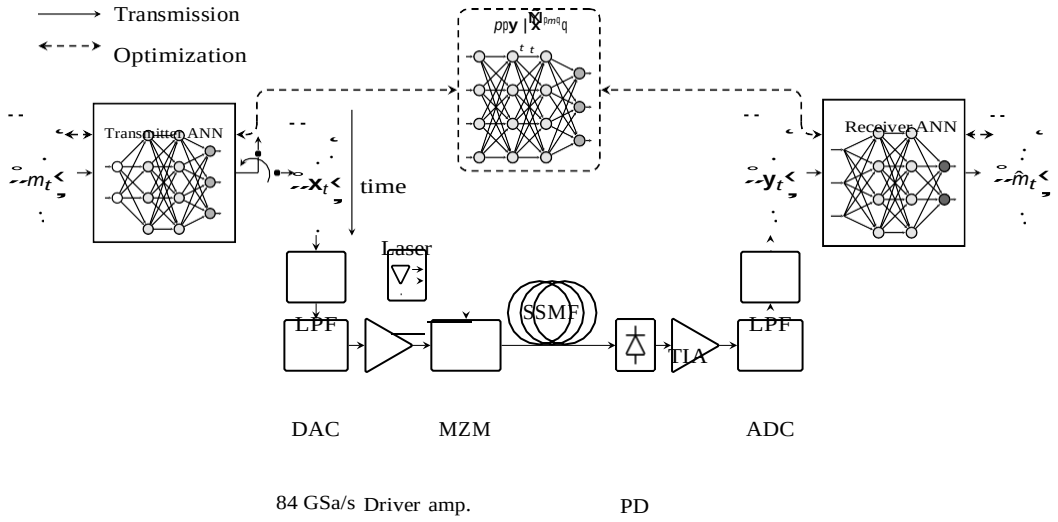

Figure 6.20: Schematic of the experimental setup for end-to-end system optimization, showing the forward propagation through the IM/DD link and the optimization flow through the generative model, used in lieu of the actual link.

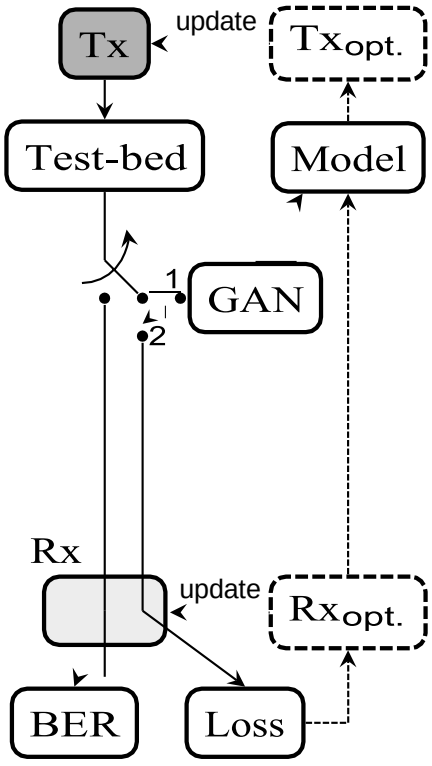

Figure 6.21: Flow chart of the proposed iterative algorithm for end-to-end deep learning using measured data. The algorithm includes transmission and optimization regimes.

ing in a 42 Gb/s data rate of the system. The electrical waveforms were fed to optical fiber transmission link, described in Sec. 6.1, which for this experiment consisted of a fixed 20 km span of SSMF. The digital signal $\ldots, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1} \ldots$, obtained at the receiver after ADC sampling, was applied in two ways: i) to the receiver ANN for processing, symbol decision and BER calculation using the optimized

bit-to-symbol mapping, described in Sec. 5.1.5. ii) together with the transmitted digital sequences $\ldots, \mathbf{x}_{t-1}, \mathbf{x}_t, \mathbf{x}_{t+1} \ldots$, it was used for GAN training, followed by transceiver optimization.

Part of the data ($q = 10^3$ elements), grouped as

$$\mathbf{A} := \begin{bmatrix} m_1 \\ \vdots \\ m_q \end{bmatrix} \in \mathbb{R}^q, \qquad \mathbf{B} := \begin{bmatrix} \mathbf{y}_1^T \\ \vdots \\ \mathbf{y}_q^T \end{bmatrix} \in \mathbb{R}^{q \times n}, \tag{6.8}$$

was used for the single step of transceiver learning within this algorithm iteration. The remainder, setting $\mu = 3$ for modeling the channel, was structured as (transmitted symbols)

$$\mathbf{C}^{(\mu=3)} := \begin{bmatrix} \vec{\mathbf{x}}_{q+2}^{(3)T} \\ \vdots \\ \vec{\mathbf{x}}_{Zw-1}^{(3)T} \end{bmatrix} \in \mathbb{R}^{(Zw \;\; q \;\; 2) \times 3n}, \tag{6.9}$$

First, the generative model was trained via the procedure described in Sec. 6.6.1. Each row i from $\mathbf{C}^{(\mu=3)}$ and $\mathbf{D}$ is used to condition the GAN and provide the ground truth symbol, respectively (see Eqs. (6.4) – (6.6)). Next, the transceiver learning was performed. The goal was to minimise the average cross entropy loss of the system over the mini-batch of size $|\underline{S}| = q$, computed as

$$\mathscr{L}_{\text{syst.}}(\boldsymbol{\theta}_{\text{Tx}}, \boldsymbol{\theta}_{\text{Rx}}) = \frac{1}{|\underline{S}|} \sum_{i=1}^{|\underline{S}|} \ell\left(m_i, f_{\text{Rx}}(\mathbf{y}_i)\right), \tag{6.11}$$

with m_i and $\mathbf{y}_i$ being rows from $\mathbf{A}$ and $\mathbf{B}$, respectively, and f_{Rx} denoting the function of the receiver. At this point, the generative model was applied *in lieu* of the transmission link. This enables the computation of the gradient of the loss with respect to both the transmitter (ϑ_{Tx}) and receiver (ϑ_{Rx}) parameter sets using backpropagation. Afterwards, these parameters are updated in a single process via SGD using the Adam algorithm with a learning rate of 10^{-3}. This completed an iteration of the optimization algorithm. The transmitter and receiver representations were

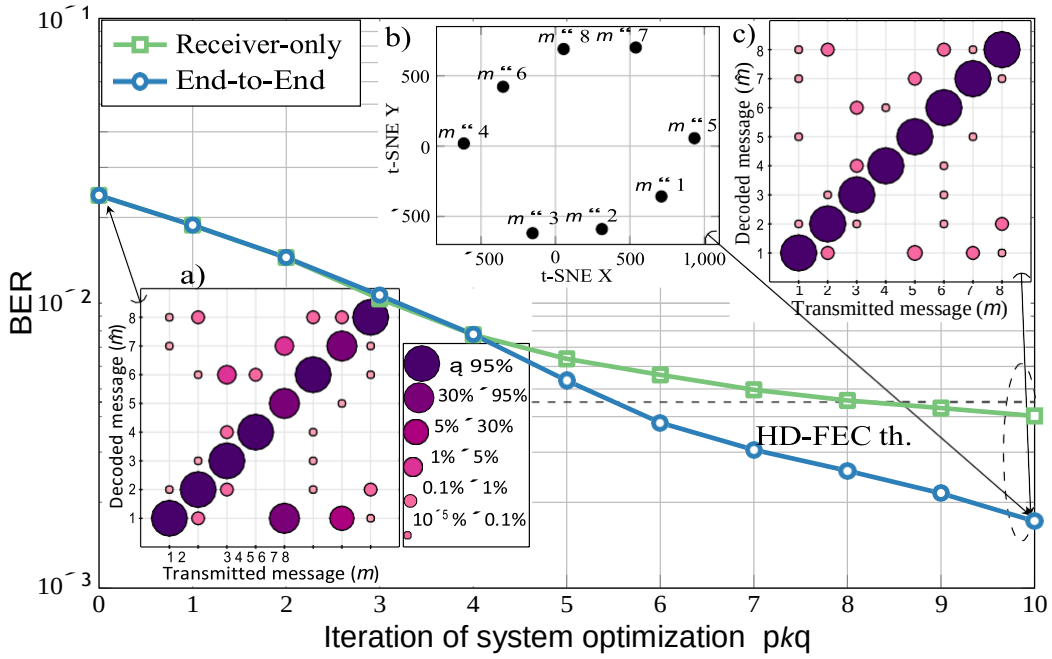

Figure 6.22: Experimental bit error rate as a function of optimization iteration. Inset figures show: a) error probabilities at $k = 0$; b) 2D t-SNE representation of the waveforms output of the transmitter ANN at $k = 10$; c) error probabilities at $k = 10$.

updated and the processes of transmission, GAN training and system optimization were sequentially repeated.

6.6.3 Performance results

The bit error rate of the system as a function of the iteration of the proposed optimization algorithm was investigated and the results are shown in Fig. 6.22. It can be observed that a monotonic BER reduction was achieved on each iteration. In particular, the BER of the system improved from $2.4 \cdot 10^{-2}$ at step $k = 0$, i.e. the case without any transceiver optimization based on the measured data, to $1.7 \cdot 10^{-3}$ after only 10 steps of the algorithm. This improvement is further illustrated via inset figures a) and c), which show the error probabilities at $k = 0$ and $k = 10$, respectively. As an illustrative example showcasing the improvement, one can observe that no errors exceeding the 1% threshold occurred at $k = 10$, which was in stark contrast compared to the initial stage. For the case of $k = 10$, the inset figure b) depicts the modulation constellation in two dimensions using the t-distributed stochastic neighbor embedding (t-SNE) dimensionality reduction technique [121], used purely for visualisation purpose. It can be seen that the 8 transmitted constellation points are well-separated in the t-SNE space, a factor that contributes to the observed message recovery enhancement.

Furthermore, Fig. 6.22 also provides a direct comparison between the performance achieved using the proposed algorithm for end-to-end optimization and the method of optimizing only the receiver using measured data. Note that in the latter case, only the set of receiver ANN parameters (ϑ_{Rx}) was updated for minimising the average cross entropy loss of the system over the mini-batch of size $|\underline{S}| = q$, i.e.

$$\underline{\mathscr{L}}_{\mathrm{syst.}}(\boldsymbol{\theta}_{\mathrm{Rx}}) = \frac{1}{|\underline{S}|}\sum_{i=1}^{|\underline{S}|} \ell\left(m_i, f_{\mathrm{Rx}}(\mathbf{y}_i)\right). \tag{6.12}$$

It can be observed that after only 4 iterations of optimization, the proposed method for end-to-end learning started to outperform the simple receiver-only strategy. The difference in achieved performance increased further as the optimization progressed. Importantly, the results indicate that the gains from receiver re-training start to diminish. At step $k = 10$ this method achieved BER of $4 \cdot 10^{-3}$, significantly higher than the $1.7 \cdot 10^{-3}$ enabled by optimizing the complete transceiver.

It is worth mentioning that the training of the GAN required 500 transmissions of different data sequences, which was the time-limiting step in such an experimental setup. It also required stable link parameters to ensure the validity of the obtained generative model. Thus, as a proof of concept, 10 optimization iterations were performed. The presented results clearly show that this already produced substantial performance gains.

6.7 Summary

This chapter described the implementation and performance of the developed FFNN and SBRNN auto-encoder designs on an experimental IM/DD transmission testbed. The conducted experiments for the first time demonstrated the end-to-end deep learning concept on an optical transmission link. The research was published in [62, 63, 122–124]. The investigation showed that for the simple approach of optimizing the auto-encoders only in simulation, the FFNN auto-encoder can transmit 42 Gb/s at 20 km below the HD-FEC threshold, while for the SBRNN auto-encoder this reach can be increased to 60 km.

Strategies for system optimization using the experimentally collected data were developed and successfully implemented. It allowed to further enhance the performance of the auto-encoders. The proposed systems showed a superior performance for a fixed processing memory in comparisons with state-of-the-art DSP algorithms also optimized with the measured data. In particular, compared to PAM system with classical nonlinear Volterra equalisation, the SBRNN auto-encoder allowed to increase the transmission reach below the HD-FEC with more than 20 km. More-

over, the SBRNN enabled the transmission of 84 Gb/s below the HD-FEC at 20 km, outperforming the advanced DSP schemes for PAM used as references.

The chapter also includes the experimental verification of the proposed method (Sec. 4.3.2) for multi-distance learning. An auto-encoder system, agnostic to distance variations was successfully demonstrated. More specifically, it allowed transmission below the HD-FEC threshold at distances ranging from 31 km to 46 km, without any reconfiguration.

The first end-to-end optimization of an optical communication system via deep learning based on measured data was also described an important step towards practical end-to-end optimized transmission. Optimization was achieved by developing an algorithm for transceiver learning using a GAN-based model of the transmission link to allow gradient computation at the transmitter via backpropagation.

Chapter 7

Conclusions and future work

7.1 Conclusions

The research described in the thesis proposes and experimentally demonstrates a fundamentally new outlook on fiber communication systems which allows to unlock the potential of data transmission over the nonlinear, dispersive channel. More specifically, the implementation of the complete fiber-optic system as an end-to-end computational graph is investigated for the first time. This approach enables the optimization of the transceiver, implemented as a deep artificial neural network, in a single deep learning process over the channel constraints. The benefits of optical fiber auto-encoder were illustrated by applying it to short reach optical fiber links based on the intensity modulation/direct detection (IM/DD) technology. These systems are characterised by a highly nonlinear detection process as well as chromatic dispersion-induced intersymbol interference (ISI). Currently, there is a lack of optimal, computationally feasible, digital signal processing (DSP) algorithms which address the data rate and system reach limitations imposed by the IM/DD impairments.

The implementation of a transceiver based on a feedforward neural network (FFNN), which processes each input independently, was investigated first. Through extensive numerical simulations it was shown that such systems can achieve bit error rates (BER) below the 6.7% hard-decision forward error correction (HD-FEC) threshold at a range of distances that are suitable for short reach applications. These results were successfully verified in a first-in-field end-to-end deep learning-based experiment. In particular, information rates of 42 Gb/s below the HD-FEC threshold at distances beyond 40 km were achieved in experiment. For a variety of link distances, the proposed FFNN auto-encoder outperformed transmission based on twoand four-level pulse amplitude modulation (PAM2/PAM4) with feedforward equalisation (FFE) – a ubiquitously deployed solution for IM/DD.

Moreover, a novel method for transceiver optimization based on multi-distance

learning was developed and verified both in simulations and experiment. It yields robust and flexible transceivers that allow—without reconfiguration—reliable transmission over a large range of link lengths, offering a significant level of flexibility. In particular, the ANN-based system was trained to tolerate significant deviations in the link's dispersion, obtaining transmitter and receiver parameters generalized for communication over varied distance. The experimental demonstration of the approach showed that, without reconfiguration, the optimized transceiver can allow operation at 42 Gb/s below the HD-FEC at a 15 km range of fiber lengths around 40 km. This is in stark contrast with conventional DSP, which is typically optimized separately for each distance.

Nevertheless, because the FFNN auto-encoder did not include connections between neighboring symbols, the design was inherently unable to compensate for the intersymbol interference associated with transmission at longer distances. It introduced a limitation on the achievable performance of such systems in terms of transmission distance. To address this, a transceiver design tailored for communication over the dispersive nonlinear channel was developed. It is based on the processing of data sequences by an end-to-end bidirectional recurrent neural network. At the receiver, a sliding window technique for an efficient sequence estimation was implemented. A comprehensive numerical study of the performance of the sliding window bidirectional recurrent neural network (SBRNN) auto-encoder was carried out. First, it was shown that the SBRNN system can achieve a significant bit-errorrate reduction for all examined distances in comparison to the previous FFNN-based design, leading to an increased transmission reach or an enhanced information rate at shorter distances. The numerical investigation showed that the SBRNN autoencoder can enable the transmission of 42 Gb/s and 84 Gb/s below the HD-FEC threshold for link lengths beyond 70 km and 30km, respectively. The performance was further enhanced by carrying out essential optimizations of the bit-to-symbol mapping and coefficient assignments in the sliding window scheme.

The SBRNN auto-encoder was compared to state-of-the-art digital signal processing solutions for nonlinear channels with ISI. In a numerical study of BER performance and computational complexity covering schemes based on PAM modulation and maximum likelihood sequence detection (MLSD), the obtained results indicated that for a fixed memory in the receiver algorithms, the SBRNN autoencoder can achieve BER close to the scheme based on PAM2. It outperformed the PAM4 scheme, which was associated with higher sensitivity to noise in the system. Importantly, in terms of complexity, evaluated in floating point operations per decoded bit, the SBRNN auto-encoder exhibits linear dependence on the assumed memory in the processing algorithm. In contrast, for the benchmark MLSD

scheme this dependence is exponential. The end-to-end SBRNN system was also compared to PAM transmission based on the simultaneous processing of multiple received symbols by a sliding window FFNN (SFFNN), a receiver scheme widely investigated for optical communications. It was shown that the auto-encoder has an improved BER performance, while training significantly fewer ANN parameters.

To complement the numerical investigation, the SBRNN auto-encoder was successfully demonstrated in an experiment. Using the simplest system implementation method based on optimizing the transceiver in simulation and applying it "as is" to the transmission test-bed, communication at information rates of 42 Gb/s below the HD-FEC was achieved at fiber lengths up to 60 km. The system was enhanced by re-optimization of the receiver using data from measurements, increasing the achievable distance below HD-FEC to 70 km. This performance was compared to other DSP schemes optimized on measured data. In particular, the PAM transmission with a sliding window FFNN (SFFNN) receiver was also investigated in experiment. Furthermore, the comparison included a PAM system with a nonlinear Volterra receiver, a classical DSP solution for nonlinear ISI equalisation. Both deep learning-based systems outperformed the conventional nonlinear Volterra equalisation. Compared to the SBRNN auto-encoder, the 42 Gb/s transmission based on PAM and SFFNN resulted in a degraded BER performance for short distances. Interestingly, in terms of system reach below the HD-FEC, the performance of the two systems was comparable. Nevertheless, when the data rates were increased to 84 Gb/s, only the SBRNN design allowed transmission below the HD-FEC at the distance of 20 km.

The experimental investigation of the optical fiber auto-encoders prompted the development and successful demonstration in an experiment of a novel method for end-to-end system optimization based on measured data. The proposed technique allows to tailor the deep learning process to the properties of the actual transmission link as opposed to using a specific, often simplified, model of the link components. The experimental demonstration of the method forms an important step towards practical end-to-end optimized optical fiber transmission. The scheme utilized a generative adversarial network (GAN) for dynamically obtaining an ANN-based model of the transmission link. The generative model was used in lieu of the link during the optimization stage, enabling the computation of gradients at the transmitter. The GAN-based method allowed a monotonic reduction in the BER of the system on each optimization step. The outcome was the first end-to-end optimization of an optical communication system via deep learning based on measured data.

In this thesis, the end-to-end optimization of optical fiber systems using ar-

tificial neural networks and deep learning is extensively investigated for the first time. The comprehensive numerical and experimental investigations as well as the detailed comparisons of performance and complexity with state-of-the-art signal processing benchmarks, highlight the developed deep learning-based transceivers as superior, lower complexity, solutions for fiber links impaired by nonlinearity and intersymbol interference. The proposed system designs and optimization methods are general and can be applied to different models and systems.

7.2 Future work

There is a number of future directions that naturally follow on from the research presented in this thesis. They can be generally classified as being related to the design of the deep learning-based system, its optimization procedure as well as the application to different types of optical fiber systems and models. This section presents a specific example for each of these directions.

7.2.1 Advanced optimization methods for distance-agnostic transceiver enhancement

In Section 4.3.2, a method for ensuring tolerance to dispersion variations was proposed within the process of transceiver optimization. The concept, that consists in forming a mini-batch of elements experiencing different fiber propagation lengths. was successfully verified in experiment (see Sec. 6.4.3). It allowed to achieve BERs below the HD-FEC for a large range of link distances using only a single set of transmitter and receiver parameters. Especially for the transmitter-based DSP such a method is of great importance as it does not require any feedback for tuning to the changing link conditions. Thus, a more detailed investigation on the trade-off between system performance and robustness for the distance-agnostic transceiver is an interesting direction for future work.

More specifically, different distributions could be considered when assigning the distances over which the sequences, elements in a mini-batch, propagate through the optical channel. This could potentially lead to a change in the error rate profile as a function of distance for the examined transceiver. For example, in contrast to the presented investigation which assumes a Gaussian distribution, employing a uniform distribution could allow for a relatively constant BER values at different distances in a specified region. Moreover, the generalization capabilities of the multi-distance learning method should be investigated in a scheme where the distance is kept fixed within all sequence of a mini-batch. It can then be changed for different batches, i.e. at each optimization step. Such an implementation would greatly facilitate the application of the approach for recurrent neural network-based

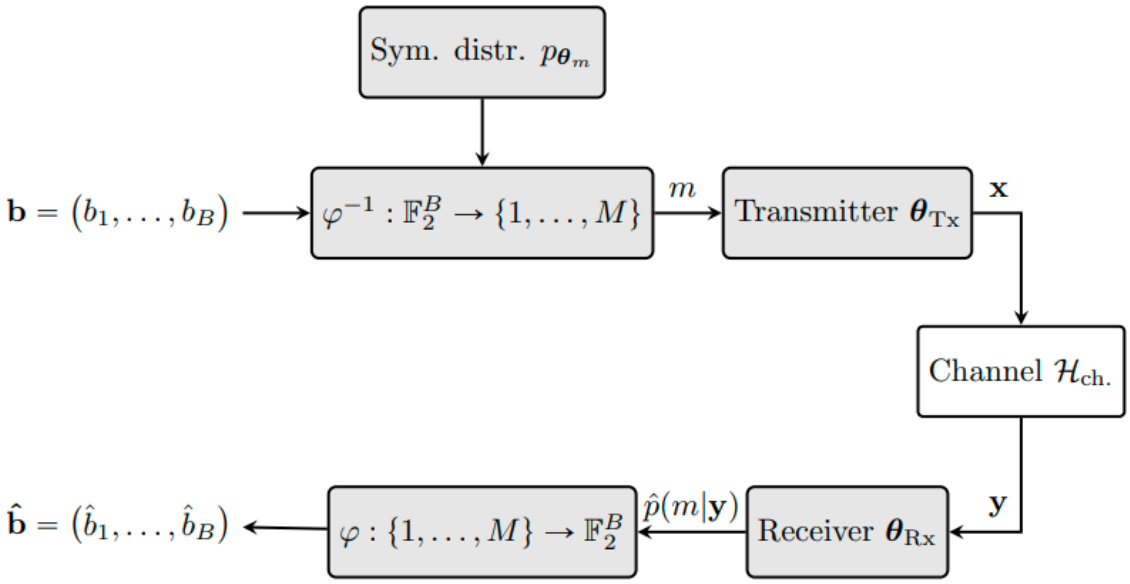

Figure 7.1: Schematic representation of an auto-encoder system which parametrises and includes optimization of the distribution of transmitted symbols (p_{ϑ_m}) together with the transmitter (ϑ_{Tx}) and receiver (ϑ_{Rx}) ANN parameters.

auto-encoders, which are associated with the encoding of long sequences.

Finally, it would be interesting to explore if performance close to the transceiver optimized at a fixed nominal distance can be achieved by the application of state-of-the-art techniques from the field of transfer learning [125].

7.2.2 Extending the auto-encoder framework

The research described in this thesis utilizes deep learning techniques to design an optical fiber transmitter whose output can be viewed as a novel multi-dimensional modulation format, optimized for resilience to the channel impairments and thus enhancing the system performance. In addition to optimizing the modulation's geometry for increasing the data rate of the system, a technique known as *probabilistic shaping* [126], which consists in controlling the occurrence of the constellation points, has been widely considered in recent years. Probabilistic shaping (PS) has been proven as an effective method for the enhancement and the seamless adaptation of the data rates as well as reach increase in optical communication systems based on conventional modulation formats [127]. However, finding the optimal input distribution for the fiber channel is an open problem, especially for multidimensional modulation formats [128], and a deep learning-based solution could be considered. The implementation of PS within the auto-encoder framework was recently proposed and for wireless communications [129, 130]. Within the FFNN optical fiber auto-encoder framework, the learning of set of robust waveforms which are transmitted through the channel can thus be extended to include the optimization of their probability of occurrence. Figure 7.1 shows a schematic of such a system

implementation. As previously explained (see Sec. 5.1.5), the sequences of bits $\mathbf{b} = b_1,\ldots,b_B$ need to be mapped into sequences of messages m. In the presence of PS this procedure is performed such that the different messages from the alphabet of size M appear with frequencies corresponding to a distribution p_{ϑ_m}. This distribution is parameterised to facilitate deep learning. Its parameters are optimized jointly with the transmitter and receiver ANNs. The application of probabilistic shaping within the auto-encoders for optical fiber communication as well as the extension of the method to multi-dimensional modulation formats is both challenging and important.

7.2.3 Auto-encoders for long-haul coherent optical fiber communications

As proof of concept, the investigation of end-to-end deep learning for optical fiber systems concentrated on short reach IM/DD communications. Nevertheless, it should be pointed out that the developed methods are general and can be extended to other, eventually more complex models and systems. In particular, an extension towards coherent optical fiber communications for long-haul transmission is a natural continuation of the work. Deep learning can be used to address the Kerr nonlinearity-induced limitations in such links. These systems are typically associated with the transmission of multiple wavelength channels over many spans of fiber and optical amplification. The higher powers at the input to each span give rise to nonlinear Kerr effects, which, combined with chromatic dispersion, produce a more complicated evolution of the transmitted signal (see Sec 3.2). As a result, both the linear and the nonlinear part of the NLSE need to be considered in the channel model. Such an equation does not have an explicit analytical solution and needs to be solved numerically.

The most commonly used numerical algorithm for solving the NLSE is the split step Fourier method (SSFM) [131]. In general, it consists in representing the dispersion term and the nonlinearity by the operators

$$\mathscr{D} = -j\frac{\beta_2}{2}\frac{\partial^2}{\partial t^2}, \tag{7.1}$$

$$\mathscr{N} = j\gamma|q(z,t)|^2, \tag{7.2}$$

respectively, and separating their effects for sufficiently small distance Δ along the fiber. The SSFM method is then applied as

$$q(z+\Delta,t) = \exp\left(\mathscr{D}\frac{\Delta}{2}\right)\exp(\mathscr{N}\Delta)\exp\left(\mathscr{D}\frac{\Delta}{2}\right)q(z,t), \tag{7.3}$$

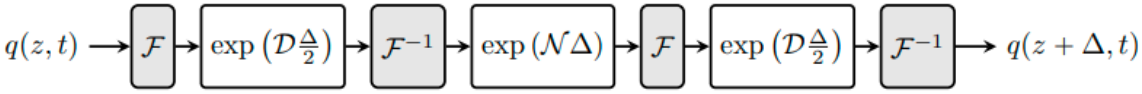

Figure 7.2: Schematic diagram of the split step Fourier method used to simulate a small propagation step Δ along the optical fiber.

where the symmetrically applied dispersion half-step reduces the approximation error [24]. As discussed in Sec. 3.2 dispersion is applied in the frequency domain, while nonlinearity is applied in the time domain.

Figure 7.2 shows a schematic diagram of the SSFM. The number of required steps for accurate simulation of fiber propagation increases with input power, bandwidth and transmission distance. Importantly, in the framework of auto-encoders such repeated split-step segments are part of the end-to-end computational graph representing the system. Thus, the optimization process via backpropagation and SGD becomes more challenging. Nevertheless, deep learning over the SSFM has been reported in [132], where the authors optimize the shape of a long pulse transmitted over the fiber. However, the application of auto-encoders to high data rate wide bandwidth coherent optical fiber communication is currently unexplored. Demonstrating the experimental implementation of such systems would be potentially of great interest to the field.

Appendix A

FFNN auto-encoder transmitted signal characteristics

During end-to-end optimization of the transceiver, the transmitter learns waveform representations which are robust to the optical channel impairments. During actual transmission on the experimental test-bed (reported in Chapter 6) different long sequences of random messages are applied to the transmitter FFNN, followed by 32 GHz LPF to generate the transmit waveforms. This section examines the temporal and spectral representations of such transmit signals.

Figure A.1 (top) shows the filtered output of the neural network, trained at (40,4) km, for the representative 10-symbol message sequence $(m_t)_{t=1}^{10}$ = (2, 36, 64, 40, 21, 53, 42, 41, 34, 13), with $m_t \in \{1, \ldots, 64\}$ denoting the input message to the ANN at time/block t. Each symbol carries 6 bits of information, consists of $n \cdot s = 48$ samples at 336 GSa/s, and is thus transmitted at 7 GSym/s, yielding a symbol duration of $T \approx 143$ ps. It can be observed that, as an effect of the implemented clipping layer in the transmitter FFNN, the waveform amplitude is limited in the linear region of operation of the Mach-Zehnder modulator with small departure from the range $[0; \pi/4]$ due to the filtering effects. Figure A.1 (bottom) also shows
the un-filtered 48 samples for each symbol in the sub-sequence ($m_{t_{t=6}}$ = (53, 42).)[7]

These blocks of samples represent the direct output of the transmitter FFNN.

The trained FFNN transmitter can be viewed as a *look-up table* which simply maps the input message to one of $M = 64$ optimized blocks. Figure A.2 illustrates the 48 amplitude levels in each of these blocks. Interestingly, it can be seen that the extremal values of 0 and $\pi/4$ are the prevailing levels. It appears that the FFNN tries to find a set of binary sequences optimized for end-to-end transmission. Nevertheless, some intermediate values are also used. To bring more clarity, the constellation of this multi-dimensional modulation format is visualised by using a state-of-the-art dimensionality reduction machine learning technique such as t-Distributed Stochas-

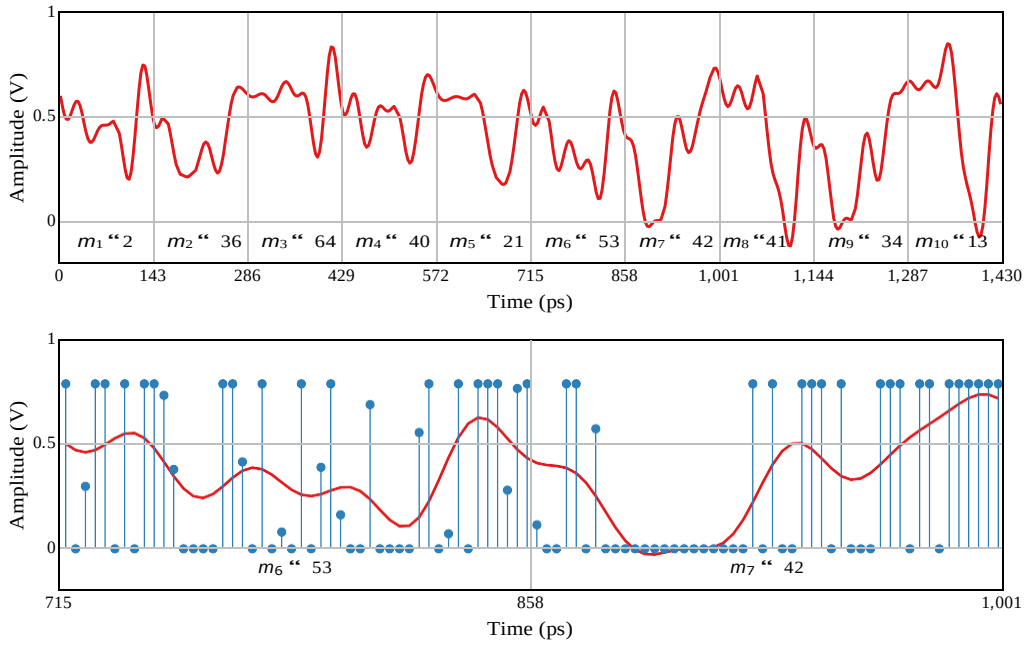

Figure A.1: Top: Output of the transmitter ANN, trained at (40,4) km, after filtering with 32 GHz brick-wall LPF for the representative random sequence of 10 symbols $(m_t)_{t=1}^{10} = (2, 36, 64, 40, 21, 53, 42, 41, 34, 13)$ transmitted at 7 GSym/s, i.e. $T \approx$ 143 ps. Bottom: Un-filtered ANN output samples, 48 per symbol, for the sub-sequence $(m_t)_{t=6}^{7} = (53, 42)$.

tic Neighbor Embedding (t-SNE) [121]. Figure A.3 shows the two-dimensional tSNE representation of the un-filtered FFNN outputs of Fig. A.2. It can be seen that the 64 different waveforms are well-separated in the t-SNE space which is an indication that they can hence be discriminated well enough at the receiver.

Figure A.4 shows the spectrum of the real-valued electrical signal at the transmitter after LPF. Because of the low-pass filtering the spectral content is confined within 32 GHz. The LPFs at both transmitter and receiver ensure that the signal bandwidth does not exceed the finite bandwidth of transmitter and receiver hardware. It can be further observed that, as a result of the block-based transmission, the signal spectrum consists of strong harmonics at frequencies that are multiples of the symbol rate.

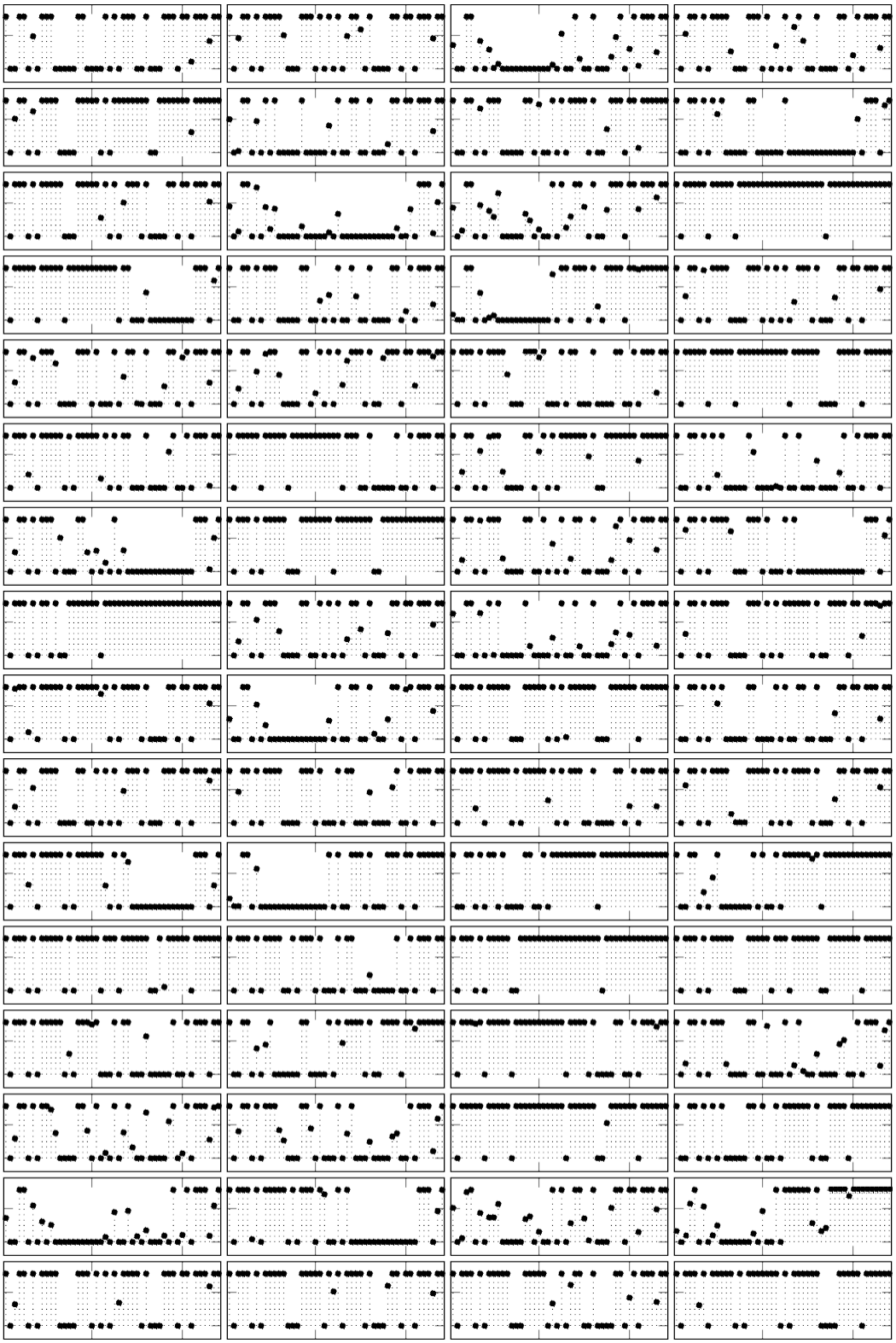

Figure A.2: All 64 possible outputs ($m = 1$ to $m = 64$, upper left to bottom right) of the transmitter FFNN before low-pass filtering.

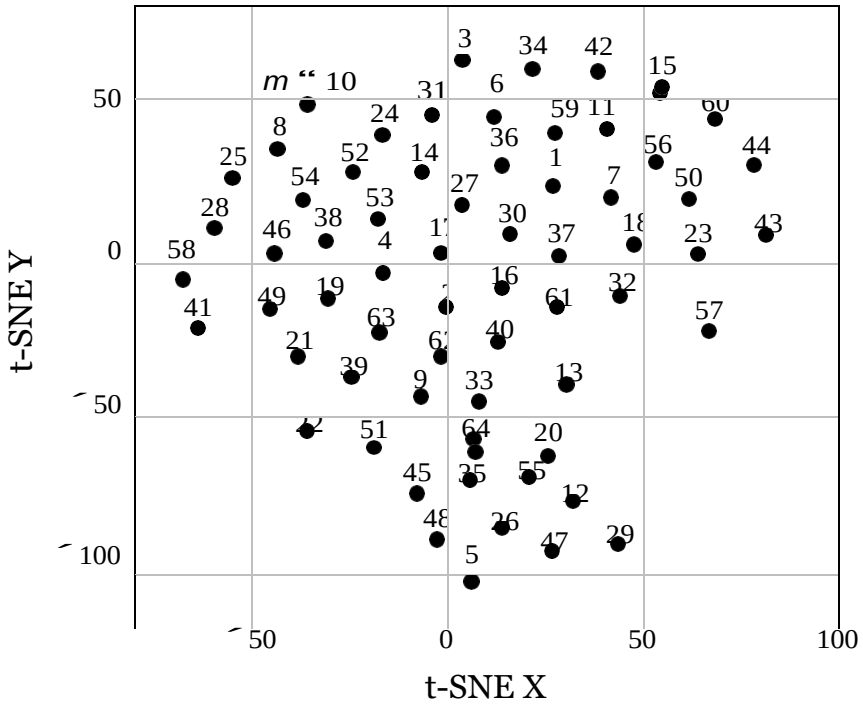

Figure A.3: t-SNE representation of the multi-dimensional waveforms output of the transmitter FFNN on the two-dimensional plane. The points are labeled with their respective message number *m*.

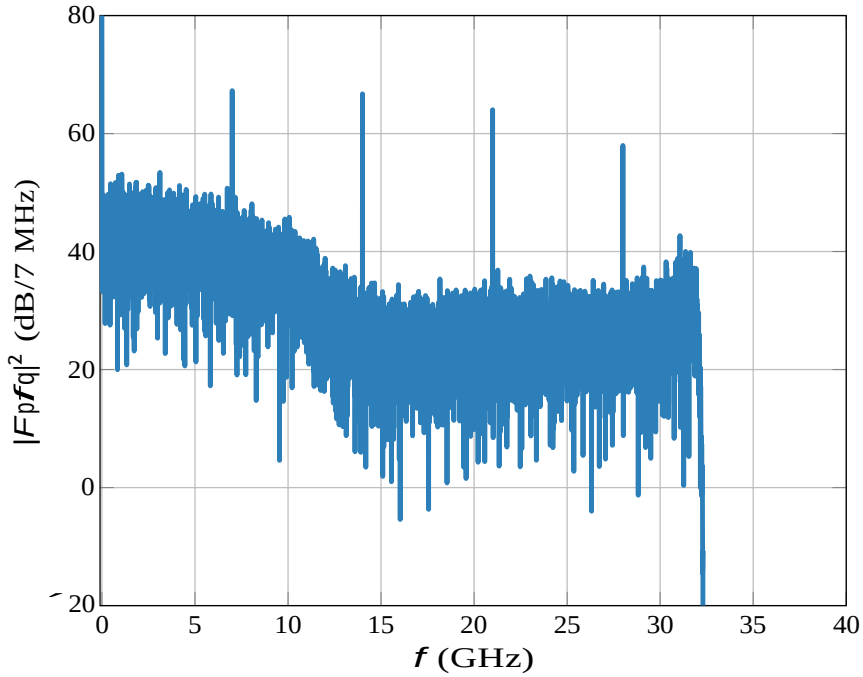

Figure A.4: Spectrum of the 32 GHz brick-wall low-pass filtered waveform at the output of the transmitter FFNN, trained at (40,4) km. The figure was generated using 7 MHz spectral bins.

Appendix B

Number of nodes and floating point operations per decoded bit in the SBRNN auto-encoder

B.1 Counting the number of nodes

In the following, an example for counting the number of nodes in the SBRNN autoencoder is provided using the vanilla variant of the system as a reference. The hyper-parameters of the vanilla SBRNN auto-encoder at the transmitter and the receiver were described in Sections 5.1.1.2 and 5.1.2.2, respectively.

According to Table 5.1, Sec. 5.1.1.2, the input to the transmitter neural network has the dimension $M + n \cdot s$, while the output ANN dimension is $n \cdot s$. The architecture assumes a single layer in both the forward and backward recurrent passes. Therefore, the number of nodes at the transmitter per single direction is given by

$$M + n \cdot s + n \cdot s = M + 2n \cdot s.$$

The network architectures in both directions are identical and thus the number of nodes in the bidirectional RNN at the transmitter becomes

$$M + 2n \cdot s + M + 2n \cdot s = 2M + 4n \cdot s. \tag{B.1}$$

At the receiver, the input dimension of the neural network, specified in Table 5.2, Sec. 5.1.2.2, is $n \cdot s + 2M$, while the input to the hidden layer is $2M$. Accordingly, the number of nodes (in a single direction) is thus

$$n \cdot s + 2M + 2M = 4M + n \cdot s.$$

Accounting for the backward direction at the receiver, which used identical hyper-

parameters, the total number of nodes at the receiver BRNN accumulates to

$$4M + n \cdot s + 4M + n \cdot s = 8M + 2n \cdot s. \tag{B.2}$$

As explained in Sec. 5.1.2.2, a final softmax layer is applied at the output of the receiver BRNN to obtain probability vectors of size M used for the optimization and error counting procedures. This layer's input is the concatenation of the outputs in the hidden layers (each of dimension $2M$) for the forward and backward processing, the size of the concatenated input vector thus becomes $4M$ and the number of nodes for the softmax stage correspondingly become

$$4M + M = 5M. \tag{B.3}$$

The total number of nodes in the vanilla SBRNN design of the auto-encoder can then be obtained by adding up Equations (B.1), (B.2), and (B.3), resulting in

$$2M + 4n \cdot s + 8M + 2n \cdot s + 5M = 15M + 6n \cdot s, \tag{B.4}$$

which is the expression used in the summary provided in Table 5.4, Sec. 5.3.1.3.

B.2 Counting the floating point operations

For the investigation in Sec. 5.3.3.3, the number of floating point operations required per encoded/decoded bit were considered as a metric which indicates the computational complexity of the SBRNN auto-encoder. In particular, as explained in Sec. 5.3.3.1, a vanilla SBRNN auto-encoder was used for this investigation. In the following, the expression for the number of floating point operations per encoded bit at the transmitter, given by Eq. (5.36), Sec. 5.3.3.3, is derived step-by-step. For this calculation, the number of floating points operations (FLOPS) in matrix multiplication was used, which for matrices $\mathbf{A} \in \mathrm{R}^{a \times b}$ and $\mathbf{B} \in \mathrm{R}^{b \times c}$ is given by [133]

$$O(bac) = (2b - 1)ac. \tag{B.5}$$

As described in Sec.5.1.1.2, the input vector to the neural network has the size $\begin{bmatrix} \mathbf{1}_{m,t}^{T} \\ \vec{\mathbf{x}}_{t-1}^{T} \end{bmatrix}^{\Sigma_T} \in \mathrm{R}^{M+n \cdot s}$, as it is a concatenation between the current one-hot vector of size M and the previous transmitter output of size $n \cdot s$. In the forward BRNN direction, it is multiplied as $\mathbf{W}_{\mathrm{fw}} \begin{bmatrix} \mathbf{1}_{m,t}^{T} \\ \vec{\mathbf{x}}_{t-1}^{T} \end{bmatrix}^{\Sigma_T}$, by a weight matrix with dimensions $\mathbf{W}_{\mathrm{fw}} \in \mathrm{R}^{n \cdot s \times (M+n \cdot s)}$. Using Eq. (B.5), the number of FLOPS for this opera-

tions is calculated as

$$(2(M + n \cdot s) - 1)n \cdot s \cdot 1 = 2n \cdot s(M + n \cdot s) - n \cdot s.$$

Identical multiplication is performed in the backward direction of the BRNN, increasing the number of FLOPS to

$$2n \cdot s(M + n \cdot s) - n \cdot s + 2n \cdot s(M + n \cdot s) - n \cdot s = 4n \cdot s(M + n \cdot s) - 2n \cdot s. \quad \text{(B.6)}$$

As explained in Sec. 5.3.3.3, in addition to the FLOPS associated with matrix multiplication, the complexity investigation also considered the floating point operations performed for the bias addition and the application of the activation function. In particular a bias vector $\mathbf{b}_{\text{fw}} \in \mathrm{R}^{n \cdot s}$ is added to the product $\mathbf{W}_{\text{fw}} \in \mathrm{R}^{n \cdot s \times (M + n \cdot s)}$ in the forward direction. The operation is identically performed in the backward pass. As a result, the FLOPS performed by the transmitter become

$$4n \cdot s(M + n \cdot s) - 2n \cdot s + n \cdot s + n \cdot s = 4n \cdot s(M + n \cdot s). \quad \text{(B.7)}$$

For the activation function (applied element-wise) at the output neural network layer of size $n \cdot s$, it was assumed that a single FLOP per element is performed. As a consequence, the total number of FLOPS in the BRNN transmitter, including the FLOPS ($2n \cdot s$) from the activation functions in both the forward and backward passes, accumulates to

$$\text{FLOPS}^{[\text{SBRNN-TX}]} = 4n \cdot s(M + n \cdot s) + 2n \cdot s = 2n \cdot s(2(M + n \cdot s) + 1). \quad \text{(B.8)}$$

The BRNN transmitter encodes $\log_2(M)$ bits of information into $n \cdot s$ samples and thus the number of FLOPS per processed bit becomes

$$\text{FLOPS}_{\text{pdb}}^{[\text{SBRNN-TX}]} = \frac{2n \cdot s(2(M + n \cdot s) + 1)}{\log_2(M)}, \quad \text{(B.9)}$$

which concludes the derivation of the expression from Eq. (5.36), Sec. 5.3.3.3.

Appendix C

Data collection in the experiments

C.1 SBRNN auto-encoder

The *training* data was formed by the sequences with indices $i \in \{1,\ldots,720\}$ and was organized as follows

$$\mathbf{D}_{\text{test}}^{\text{AE}} := \begin{bmatrix} \mathbf{y}_{1,1\ldots N}^T & \mathbf{y}_{2,1\ldots N}^T & \mathbf{y}_{3,1\ldots N}^T & \mathbf{y}_{4,1\ldots N}^T \\ \mathbf{y}_{5,1\ldots N}^T & \mathbf{y}_{6,1\ldots N}^T & \mathbf{y}_{7,1\ldots N}^T & \mathbf{y}_{8,1\ldots N}^T \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{y}_{717,1\ldots N}^T & \mathbf{y}_{718,1\ldots N}^T & \mathbf{y}_{719,1\ldots N}^T & \mathbf{y}_{720,1\ldots N}^T \end{bmatrix}, \tag{C.1}$$

$$\mathbf{L}_{\text{test}}^{\text{AE}} := \begin{bmatrix} m_{1,1\ldots N} & m_{2,1\ldots N} & m_{3,1\ldots N} & m_{4,1\ldots N} \\ m_{5,1\ldots N} & m_{6,1\ldots N} & m_{7,1\ldots N} & m_{8,1\ldots N} \\ \vdots & \vdots & \vdots & \vdots \\ m_{717,1\ldots N} & m_{718,1\ldots N} & m_{719,1\ldots N} & m_{720,1\ldots N} \end{bmatrix}, \tag{C.2}$$

test

where $\mathbf{D}_{\text{train}}^{\text{AE}} \in \mathsf{R}^{180\times 4\cdot N\cdot n\cdot s}$ consists of the data elements $\mathbf{y}_{i,j}$ and $\mathbf{L}_{\text{train}}^{\text{AE}} \in \mathsf{R}^{180\times 4\cdot N}$ contains the label elements $m_{i,j}$. During the procedure of training the receiver on measured data (see Sec. 6.5.1) a window of V columns is picked to form the minibatch on each stochastic gradient descent optimization step. More specifically, at an optimization step s, the mini-batch of elements[1] $\mathbf{D}^{\text{AE}}[:, s:(s+V)]$ was processed by the receiver to obtain corresponding output probability vectors, where V is the training window which was fixed to $V = 10$. The cross entropy loss between the input messages $\mathbf{L}^{\text{AE}}[:, s:(s+V)]$ and the probability outputs was computed and averaged before completing a single iteration of the optimization algorithm.

To ensure a sufficiently large mini-batch of independent examples, when forming the data-sets, one full DAC load was split into two parts, treated indepen-

[1]Note that the MATLAB notation $\mathbf{A}[:, i:j]$ is used to denote extracting the $j - i + 1$ columns of column indices $i, i+1, \ldots, j$ from $\mathbf{A}$.

It was verified that the converg ence of the loss is achieved within one epoch of the training data, which was used as a stopping criterion. The remaining sequences with indices $i \in \{721,\ldots, Z\}$, which are independent from the training data, were used to form the *testing* data, identically organized in $\mathbf{D}^{\text{AE}}_{\text{test}}$ and $\mathbf{L}^{\text{AE}}_{\text{test}}$.

C.2 Reference PAM systems

Sequences $i \in \{1,\ldots, 90\}$ were used for *training*, organized as

$$\mathbf{D}^{\text{PAM}}_{\text{train}} := \begin{bmatrix} \mathbf{y}^T_{1,1} & \cdots & \mathbf{y}^T_{1,N/2} \\ \mathbf{y}^T_{1,N/2+1} & \cdots & \mathbf{y}^T_{1,N} \\ \vdots & \ddots & \vdots \\ \mathbf{y}^T_{90,1} & \cdots & \mathbf{y}^T_{90,N/2} \\ \mathbf{y}^T_{90,N/2+1} & \cdots & \mathbf{y}^T_{90,N} \end{bmatrix} \tag{C.3}$$

$$\mathbf{L}^{\text{PAM}}_{\text{train}} := \begin{bmatrix} m_{1,1} & \cdots & m_{1,N/2} \\ m_{1,N/2+1} & \cdots & m_{1,N} \\ \vdots & \ddots & \vdots \\ m_{90,1} & \cdots & m_{90,N/2} \\ m_{90,N/2+1} & \cdots & m_{90,N} \end{bmatrix} \tag{C.4}$$

where $\mathbf{D}^{\text{PAM}}_{\text{train}} \in \mathbb{R}^{180\times N\cdot n/2}$ consists of the received elements $\mathbf{y}_{i,j}$ and $\mathbf{L}^{\text{PAM}}_{\text{train}} \in \mathbb{R}^{180\times N/2}$ has the elements $m_{i,\,j}$. During the DSP optimization for the deep learningbased PAM systems (see Appendices C.2.1 and C.2.3), a window of W from the data was picked column-wise to form a mini-batch. Similar to the auto-encoder training, one full DAC load was split into two parts that are independently treated by the DSP, e.g. $\begin{bmatrix} \mathbf{y}^T_{1,1} & \cdots & \mathbf{y}^T_{1,N/2} \\ \mathbf{y}^T_{1,N/2+1} & \cdots & \mathbf{y}^T_{1,N} \end{bmatrix}$, ensuring an identical mini-batch size. Again, one epoch over the training data was performed during optimization, while the *testing* was carried out using the disjoint set of remaining sequences with indices $i \in \{91,\ldots, 100\}$, identically formed as $\mathbf{D}^{\text{PAM}}_{\text{test}}$ and $\mathbf{L}^{\text{PAM}}_{\text{test}}$.

C.2.1 Optimization of the SFFNN receiver

The receiver ANN parameters in this scheme were optimized using the experimental data as follows: At an iteration s, the mini-batch of elements $\mathbf{D}^{\text{PAM}}_{\text{train}}[:, s:(s+W)]$ was processed by the FFNN to obtain the corresponding output probability vectors. The average cross entropy loss between these outputs and the input messages $\mathbf{L}^{\text{PAM}}_{\text{train}}[:, s+\frac{W-1}{2}]$ was computed to complete an optimization step of the FFNN parameters

via backpropagation and SGD using the Adam algorithm (see Chapter 2.

C.2.2 Optimization of the Volterra receiver

The first and second order Volterra filter coefficients were optimized using the MATLAB `fitlm` function for linear and polynomial regression. The optimization of the coefficients was performed on a randomly chosen pair of transmitted sequence $\mathbf{L}^{\text{PAM}}_{\text{train}}[i,:]$ and received samples $\mathbf{D}^{\text{PAM}}_{\text{train}}[i,:]$.

C.2.3 Optimization of the SBRNN receiver

The elements of the collected PAM datasets $\mathbf{D}^{\text{PAM}}_{\text{train}}$ and $\mathbf{L}^{\text{PAM}}_{\text{train}}$ were used in the training algorithm. At an iteration s of the optimization algorithm, the mini-batch of elements $\mathbf{D}^{\text{PAM}}_{\text{train}}[:,s:(s+V)]$ was processed by the BRNN to obtain the corresponding output probability vectors. The average cross entropy loss was computed between the input messages $\mathbf{L}^{\text{PAM}}_{\text{train}}[:,s:(s+V)]$ and the probability outputs. Then the optimization step of the BRNN was completed via backpropagation and SGD using the Adam algorithm (see Chapter 2. The training window was fixed to $V=61$, such that the scheme was trained with a processing memory identical to the auto-encoder.

Appendix D

Acronyms

ADC Analogue-to-digital converter

ANN Artificial neural network

AWGN Additive white Gaussian noise

BER Bit error rate

BRNN Bidirectional recurrent artificial neural network

DAC Digital-to-analogue converter

DBP Digital backpropagation

DD Direct detection

DL Deep learning

DSP Digital signal processing **EDFA**

Erbium-doped fiber amplifier **FEC**

Forward error correction

FFE Feedforward equaliser

FFNN Feedforward artificial neural network

FFT Fast Fourier transform

FTTH Fiber-to-the-home

FLOPS Floating point

operations

GAN Generative adversarial network

GD Gradient descent

GRU Gated recurrent unit

HDD Hard decision decoding

HD-FEC Hard-decision forward error correction

IFFT Inverse fast Fourier transform

IM Intensity modulation

IM/DD Intensity modulation/direct detection

ISI Intersymbol interference

LDBP Learned DBP

LPF Low-pass filter

LSTM Long short-term memory

LSTM-GRU Long short-term gated recurrent unit

MAP Maximum a-posteriori probability

ML Machine learning

MLSD Maximum likelihood sequence detection

MZM Mach-Zehnder modulator

NLSE Nonlinear Schrödinger

equation **PAM** Pulse amplitude

modulation

PD Photodiode

PDE Partial differential equation

PIN Positive-intrinsic-negative

PON Passive optical network

PRBS Pseudo-random binary sequence

PS Probabilistic shaping

ReLU Rectified linear unit

RNN Recurrent artificial neural network

SBRNN Sliding window bidirectional recurrent artificial neural network

SER Symbol error rate

SFFNN Sliding window feedforward artificial neural network

SGD Stochastic gradient descent

SNR Signal-to-noise ratio

SSFM Split-step Fourier

method

SSMF Standard single mode fiber

TDM Tunable dispersion module

TIA Trans-impedance amplifier

t-SNE t-distributed stochastic neighbor embedding

Bibliography

[1]T. Mitchell. *Machine learning*. McCraw Hill, Inc., 1997.

[2]C. Bishop. *Pattern recognition and machine learning*. Springer, 2006. [3]M. de Pardo. *Advances in financial machine learning*. John Wiley & Sons, 2018.

[4]J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng. Large scale distributed deep networks. In *Proc. of Advances in Neural Information Processing Systems (NIPS) 25*, pages 1–9, 2012.

[5]Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(6):436–444, 2015.

[6]A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Proc. of Advances in Neural Information Processing Systems (NIPS) 25*, pages 1090–1098, 2012.

[7]G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[8]R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[9]M. Ibnkahla. Applications of neural networks to digital communications—a survey. *Signal Processing*, 80(7):1185–1215, 2000.

[10]O. Simeone. A very brief introduction to machine learning with applications to communication systems. *IEEE Transactions on Cognitive Communications and Networking*, 4(4):648–664, 2018.

[11]F. N. Khan, Q. Fan, C. Lu, and A. P. T. Lau. An optical communication's perspective on machine learning and its applications. *Journal of Lightwave Technology*, 37(2):493–516, 2019.

[12]I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT press, 2016.

[13]D. Rumelhart, G. Hinton, and R. Williams. Learning representations by backpropagating errors. *Nature*, 323:533–536,1986.

[14]A. Cauchy. Méthode générale pour la résolution des syst'emes d'équations simultanées. *C.R. Acad. Sci. Paris*, 25:536–538, 1847.

[15]H. B. Curry. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2(3):258–261, 1944.

[16]D. Kingma and J. Ba. Adam: a method for stochastic optimization. In *Proc. of International Conference for Learning Representations*, pages 1–15, 2015.

[17]A. Karpathy. A peek at trends in machine learning. *Medium. Accessed on: Aug. 4, 2020. [Online]. Available: https://medium.com/@karpathy/a-peek-at-trends-in-machine-learning-ab8a1085a106*, Apr. 7, 2017.

[18]S. Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

[19]S. Cammerer. Deep learning-based polar code design. *presented at the 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, September 25, 2019.*

[20]O. Simeone. A brief introduction to machine learning for engineers. *Foundations and Trends® in Signal Processing*, 12(3-4):200–431, 2017.

[21]E. Brynjolfsson and T. Mitchell. What can machine learning do? Workforce implications. *Science*, 358(6370):1530–1534, 2017.

[22]T. S. Rappaport. *Wireless communications: Principles and practice, 2nd ed.* Prentice Hall, 2002.

[23]G. Agrawal. *Fiber-Optic Communication systems, 4th ed*. Wiley, 2010.

[24]G. Agrawal. *Nonlinear fiber optics, 5th ed.* Academic Press, 2013.

[25]J. Proakis and M. Salehi. *Digital Communications, 5th ed.* McGraw-Hill Education, 2007.

[26]Cisco Systems Inc. Cisco global cloud index: Forecast and methodology, 2016 2021. *White Paper*, 2018.

[27]H. Kim and N. Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, 2013.

[28]Huawei Technologies Co. Ltd. White paper on technological developments of optical networks. *White Paper*, 2016.

[29]D. Cote. Using machine learning in communication networks. *IEEE/OSA Journal of Optical Communications and Networking*, 10(10):D100–D109, 2018.

[30]M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang. Machine learning for networking: workflow, advances and opportunities. *IEEE Network*, 32(2):92– 99, 2018.

[31]A. Deylamsalehi, D. A. P. Davis, P. Afsharlar, M. Bahrami, W. Chen, and V. M. Vokkarane. Using machine learning to balance energy cost and emissions in optical networks. *IEEE/OSA Journal of Optical Communications and Networking*, 10(10):D72–D83, 2018.

[32]F. Musumeci, C. Rottondi, A. Nag, I. Macaluso, D. Zibar, M. Ruffini, and M. Tornatore. An overview on application of machine learning techniques in optical networks. *IEEE Communications Surveys & Tutorials*, 21(2):1383–1408, 2019.

[33]R. Boutaba, M. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. EstradaSolano, and O. M. Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(16):1–99, 2018.

[34]G. Choudhury, G. Thakur, and S. Tse. Joint optimization of packet and optical layers of a core network using SDN controller, CD ROADMs and machine-learning-based traffic prediction. In *Proc. of Optical Fiber Communications Conference*, pages 1–3, 2019.

[35]Z. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani. State-of-the-art deep learning: evolving machine intelligence toward

tomorrow's intelligent network traffic control systems. *IEEE Communications surveys & tutorials*, 19(4):2432–2455, 2017.

[36]F. Morales, M. Ruiz, L. Gifre, L. M. Contreras, V. Lopez, and L. Velasco. Virtual network topology adaptability based on data analytics for traffic prediction. *IEEE/OSA Journal of Optical Communications and Networking*, 9(1):A35–A45, 2017.

[37]D. Rafique and L. Velasco. Machine learning for network automation: overview, architecture, and applications. *IEEE/OSA Journal of Optical Communications and Networking*, 10(10):D126–D143, 2018.

[38]Y. Okada, S. Ata, N. Nakamura, Y. Nakahira, and I. Oka. Comparisons of machine learning algorithms for application identification of encrypted traffic. In *Proc. of 10th International Conference on Machine Learning and Applications*, pages 358–361, 2011.

[39]B. Dong and X. Wang. Comparison of deep learning method to traditional methods used for network intrusion detection. In *8th IEEE International Conference on Communication Software and Networks*, pages 581– 585, 2016.

[40]T. P. Oliveira, J. S. Barbar, and A. S. Soares. *Multilayer Perceptron and Stacked Autoencoder for Internet Traffic Prediction*. Springer Berlin Heidelberg, 2014.

[41]J. Proakis and D. Manolakis. *Digital signal processing: principles algorithms and applications*, *3rd ed.* Prentice Hall, 2001.

[42]E. Agrell, M. Karlsson, A. R. Chraplyvy, D. J. Richardson, P. M. Krummrich, P. Winzer, K. Roberts, J. K. Fischer, S. J. Savory, B. J. Eggleton, M. Secondini, F. R. Kschischang, A. Lord, J. Prat, I. Tomkos, J. E. Bowers, S. Srinivasan, M. Brandt-Pearce, and N. Gisin. Roadmap of optical communications. *Journal of Optics*, 18(6):063002, May 2016.

[43]O. Shental and J. Hoydis. Machine LLRning: learning to softly demodulate. In *IEEE Globecom Workshops (GC Wkshps)*, pages 1–7, 2019.

[44]E. Nachmani, Y. Be'ery, and D. Burshtein. Learning to decode linear codes using deep learning. In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 341–346, 2016.

[45]T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink. On deep learning-based channel decoding. In *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2017.

[46]S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink. Scaling deep learningbased decoding of polar codes via partitioning. In *GLOBECOM 2017 2017 IEEE Global Communications Conference*, pages 1–6, 2017.

[47]D. Tandler, S. Dörner, S. Cammerer, and S. ten Brink. On recurrent neural networks for sequence-based processing in communications. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 537–543, 2019.

[48]A. Buchberger, C. Häger, H. D. Pfister, L. Schmalen, and A. Graell i Amat. Pruning neural belief propagation decoders. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 1–5, 2020.

[49]E. Ip and J. M. Kahn. Compensation of dispersion and nonlinear impairments using digital backpropagation. *Journal of Lightwave Technology*, 26(20):3416–3425, 2008.

[50]C. Häger and H. D. Pfister. Nonlinear interference mitigation via deep neural networks. In *Proc. of Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3, 2018.

[51]C. Fougstedt, C. Häger, L. Svensson, H. D. Pfister, and P. Larsson-Edefors. ASIC implementation of time-domain digital backpropagation with deeplearned chromatic dispersion filters. In *Proc. of European Conference on Optical Communication (ECOC)*, pages 1–3, 2018.

[52]V. Oliari, S. Goossens, C. Häger, G. Liga, R. M. Bütler, M. v. d. Hout, S. v. d. Heide, H. D. Pfister, C. Okonkwo, and A. Alvarado. Revisiting efficient multi-step nonlinearity compensation with machine learning: an experimental demonstration. *Journal of Lightwave Technology*, 38(12):3114–3124, 2020.

[53]S. Chen, G. J. Gibson, C. F. N. Cowan, and P. M. Grant. Adaptive equalization of finite non-linear channels using multilayer perceptrons. *Signal Processing*, 20:107–119, 1990.

[54]N. Benvenuto, M. Marchesi, F. Piazza, and A. Uncini. Non linear satellite radio links equalized using blind neural networks. In *[Proceedings] ICASSP*

91: 1991 International Conference on Acoustics, Speech, and Signal Processing, pages 1521–1524 vol.3, 1991.

[55]T. Koike-Akino, Y. Wang, D. S. Millar, K. Kojima, and K. Parsons. Neural turbo equalization: deep learning for fiber-optic nonlinearity compensation. *Journal of Lightwave Technology*, 38(11):3059–3066, 2020.

[56]V. Kamalov, L. Jovanovski, V. Vusirikala, S. Zhang, F. Yaman, K. Nakamura, T. Inoue, E. Mateo, and Y. Inada. Evolution from 8QAM live traffic to PS 64QAM with neural-network based nonlinearity compensation on 11000 km open subsea cable. In *Proc. of Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3, 2018.

[57]J. Estaran, R. Rios-Mueller, M. A. Mestre, F. Jorge, H. Mardoyan, A. Konczykowska, J. . Dupuy, and S. Bigo. Artificial neural networks for linear and non-linear impairment mitigation in high-baudrate IM/DD systems. In *Proc. of European Conference on Optical Communication (ECOC)*, pages 1–3, 2016.

[58]V. Houtsma, E. Chou, and D. van Veen. 92 and 50 gbps TDM-PON using neural network enabled receiver equalization specialized for PON. In *Proc. of Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3, 2019.

[59]T. J. O'Shea, K. Karra, and T. C. Clancy. Learning to communicate: channel auto-encoders, domain specific regularizers, and attention. In *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 223–228, 2016.

[60]T. J. O'Shea and J. Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

[61]S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):132–143, 2018.

[62]B. Karanov, M. Chagnon, F. Thouin, T. A. Eriksson, H. Bülow, D. Lavery, P. Bayvel, and L. Schmalen. End-to-end deep learning of optical fiber communications. *Journal of Lightwave Technology*, 36(20):4843–4855, 2018.

[63]M. Chagnon, B. Karanov, and L. Schmalen. Experimental demonstration of a dispersion tolerant end-to-end deep learning-based IM-DD transmission system. In *Proc. of European Conference on Optical Communication (ECOC)*, pages 1–3, 2018.

[64]R. T. Jones, T. A. Eriksson, M. P. Yankov, and D. Zibar. Deep learning of geometric constellation shaping including fiber nonlinearities. In *Proc. of European Conference on Optical Communication (ECOC)*, pages 1–3, 2018.

[65]S. Li, C. Häger, N. Garcia, and H. Wymeersch. Achievable information rates for nonlinear fiber communication via end-to-end autoencoder learning. In *Proc. of European Conference on Optical Communication (ECOC)*, pages 1–3, 2018.

[66]N. Stojanovic, F. Karinou, Z. Qiang, and C. Prodaniuc. Volterra and wiener equalizers for short-reach 100G PAM-4 applications. *Journal of Lightwave Technology*, 35(21):4583–4594, 2017.

[67]G. D. Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[68]P. Poggiolini, G. Bosco, J. Prat, R. Killey, and S. Savory. Branch metrics for effective long-haul MLSE IMDD receivers. In *2006 European Conference on Optical Communications*, pages 1–2, 2006.

[69]G. Bosco, P. Poggiolini, and M. Visintin. Performance analysis of MLSE receivers based on the square-root metric. *Journal of Lightwave Technology*, 26(14):2098–2109, 2008.

[70]M. Chagnon. Optical communications for short reach. *Journal of Lightwave Technology*, 37(8):1779–1797, 2019.

[71]W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[72]K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[73]G. Gybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control*, *Signals and Systems*, 2:303–314, 1989.

[74]K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[75]V. Nair and G. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. of Int. Conf. Mach. Learn.*, pages 807–814, 2010.

[76]P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1:339–356, 1988.

[77]Y. Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3, 1993.

[78]S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[79]I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS) 27*, pages 3104–3112, 2014.

[80]R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 2342–2350, 2015.

[81]K. Cho, B. van Merriënboer, c. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[82]M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[83]A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[84]I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS) 27*, pages 2672–2680, 2014.

[85]L. Devroye. *Non-uniform random variate generation*. Springer-Verlag, 1986.

[86]W. H. Press, W. Vetterling, S. Teukolsky, and B. P. Flannery. *Numerical recipes, 3rd ed.* Cambridge University Press, 2007.

[87]H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[88]L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of 19th International Conference on Computational Statistics*, pages 177–186, 2010.

[89]F.-F. Li, R. Krishna, and D. Xu. Convolutional neural networks for visual recognition. *CS231n Class, Module 1: Neural Networks, Lecture: Learning and Evaluation, Stanford University. Accessed on: Aug. 4, 2020. [Online]. Available: https://cs231n.github.io/neural-networks-3/*, Spring 2020.

[90]G. Strang. *Calculus, 3rd ed.* Wellesley-Cambridge Press, 2017.

[91]D. Plabst, J. G. Gómez, T. Wiegart, and N. Hanik. Wiener filter for shortreach fiber-optic links. *IEEE Communications Letters*, pages 1–1, 2020.

[92]Q. Hu, M. Chagnon, K. Schuh, F. Buchali, and H. Bülow. High data rate tomlinson-harashima-precoding-based pam transmission. In *45th European Conference on Optical Communication (ECOC 2019)*, pages 1–4, 2019.

[93]Application Note. High-speed DACs. *Tektronix. Accessed on: Aug. 21, 2020. [Online]. Available: https://www.tek.com/dl/76W_30631_0_HR_Letter*.

[94]K. Gentile. The effect of DAC resolution on spurious performance. Analog Devices, Inc. *Accessed on: Aug. 21, 2020. [Online]. Available: https://www.analog.com/media/en/training-seminars/design-handbooks/Technical-Tutorial-DDS/Section4.pdf*, 1999.

[95]W. Kester. Taking the mystery out of the infamous formula, "SNR = 6.02N + 1.76dB," and why you should care. Tutorial MT-001, Ana log Devices, Inc. *Accessed on: Aug. 21, 2020. [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-001.pdf*, 2009.

[96]C. Schmidt, C. Kottke, V. Jungnickel, and R. Freund. High-speed digitalto-analog converter concepts. In *Next-Generation Optical Communication: Components, Sub-Systems, and Systems VI*, volume 10130, pages 133 – 141, 2017.

[97]A. Napoli, M. M. Mezghanni, S. Calabrò, R. Palmer, G. Saathoff, and B. Spinnler. Digital predistortion techniques for finite extinction ratio IQ

Mach–Zehnder modulators. *Journal of Lightwave Technology*, 35(19):4289– 4296, 2017.

[98] Application Bulletin. Noise analysis of FET transimpedance amplifiers. Burr-Brown Corp. *Accessed on: Aug. 21, 2020. [Online]. Available: https://www.ti.com/lit/an/sboa060/sboa060.pdf*, 1994.

[99] Tensorflow. *[Online]. Available: https://www.tensorflow.org/*, 2020.

[100]C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[101]L. Schmalen, A. J. de Lind van Wijngaarden, and S. ten Brink. Forward error correction in optical core and optical access networks. *Bell Labs Technical Journal*, 18(3):39–66, 2013.

[102]E. Agrell and M. Secondini. Information-theoretic tools for optical communications engineers. In *2018 IEEE Photonics Conference (IPC)*, pages 1–5, 2018.

[103]F. Gray. Pulse code communication, US Patent US2632058A, Mar. 1953.

[104]T. A. Eriksson, H. Bülow, and A. Leven. Applying neural networks in optical communication systems: possible pitfalls. *IEEE Photonics Technology Letters*, 29(23):2091–2094, 2017.

[105]M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, 8(1):3–30, 1998.

[106]D.-U. Lee, J. D. Villasenor, W. Luk, and P. H. W. Leong. A hardware Gaussian noise generator using the Box-Muller method and its error analysis. *IEEE Transactions on Computers*, 55(6):659–671, 2006.

[107]Z. Wang. Super-FEC codes for 40/100 Gbps networking. *IEEE Communications Letters*, 16(12):2056–2059, 2012.

[108]N. Farsad and A. Goldsmith. Neural network detection of data sequences in communication systems. *IEEE Transactions on Signal Processing*, 66(21):5663–5678, 2018.

[109]N. Farsad and A. Goldsmith. Neural network detectors for molecular communication systems. In *2018 IEEE 19th International Workshop on Sig nal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5, 2018.

[110]K. Zeger and A. Gersho. Pseudo-gray coding. *IEEE Transactions on Communications*, 38(12):2147–2158, 1990.

[111]F. Glover. Tabu search – part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[112]R. T. Jones, M. P. Yankov, and D. Zibar. End-to-end learning for GMI optimized geometric constellation shape. In *45th European Conference on Optical Communication (ECOC 2019)*, pages 1–4, 2019.

[113]I. Lyubomirsky. Machine learning equalization techniques for high speed PAM4 fiber optic communication systems. *CS229 Final Project Report. Stanford University (2015). Accessed on: Aug. 21, 2020. [Online]. Available: http://cs229.stanford.edu/proj2015/232_report.pdf*.

[114]D. van Veen and V. Houtsma. Strategies for economical next-generation 50g and 100G passive optical networks [invited]. *IEEE/OSA Journal of Optical Communications and Networking*, 12(1):A95–A103, 2020.

[115]O. Sidelnikov, A. Redyuk, and S. Sygletos. Equalization performance and complexity analysis of dynamic deep neural networks in long haul transmission systems. *Opt. Express*, 26(25):32765–32776, 2018.

[116]B. Karanov, D. Lavery, P. Bayvel, and L. Schmalen. End-to-end optimized transmission over dispersive intensity-modulated channels using bidirectional recurrent neural networks. *Opt. Express*, 27(14):19650–19663, Jul 2019.

[117]B. Karanov, G. Liga, V. Aref, D. Lavery, P. Bayvel, and L. Schmalen. Deep learning for communication over dispersive nonlinear channels: performance and comparison with classical digital signal processing. In *Proc. of 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 192–199, 2019.

[118]B. Karanov, P. Bayvel, and L. Schmalen. End-to-end learning in optical fiber communications: concept and transceiver design. In *Proc. of European Conference on Optical Communication (ECOC)*, pages 1–4, 2020.

[119]F. A. Aoudia and J. Hoydis. Model-free training of end-to-end communication systems. *IEEE Journal on Selected Areas in Communications*, 37(11):2503–2516, 2019.

[120]D. J. F. Barros and J. M. Kahn. Comparison of orthogonal frequency-division multiplexing and on-off keying in amplified direct-detection single-mode fiber systems. *Journal of Lightwave Technology*, 28(12):1811–1820, 2010.

[121]L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res*, 9:2579–2605, 2008.

[122]B. Karanov, M. Chagnon, V. Aref, D. Lavery, P. Bayvel, and L. Schmalen. optical fiber communication systems based on end-to-end deep learning. In *Proc. of IEEE Photonics Conference (IPC)*, pages 1–2, 2020.

[123]B. Karanov, M. Chagnon, V. Aref, D. Lavery, P. Bayvel, and L. Schmalen. Concept and experimental demonstration of optical IM/DD end-to-end system optimization using a generative model. In *Proc. of Optical Fiber Communications Conference (OFC)*, pages 1–3, 2020.

[124]B. Karanov, M. Chagnon, V. Aref, F. Ferreira, D. Lavery, P. Bayvel, and L. Schmalen. Experimental investigation of deep learning for digital signal processing in short reach optical fiber communications. In *Proc. of IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 1–6, 2020.

[125]S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[126]F. R. Kschischang and S. Pasupathy. Optimal nonuniform signaling for gaussian channels. *IEEE Transactions on Information Theory*, 39(3):913–929, 1993.

[127]F. Buchali, F. Steiner, G. Böcherer, L. Schmalen, P. Schulte, and W. Idler. Rate adaptation and reach increase by probabilistically shaped 64-QAM: an experimental demonstration. *Journal of Lightwave Technology*, 34(7):1599– 1609, 2016.

[128]R. Dar, M. Feder, A. Mecozzi, and M. Shtaif. On shaping gain in the nonlinear fiber-optic channel. In *2014 IEEE International Symposium on Information Theory*, pages 2794–2798, 2014.

[129]M. Stark, F. Ait Aoudia, and J. Hoydis. Joint learning of geometric and probabilistic constellation shaping. In *IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, 2019.

[130]F. Ait Aoudia and J. Hoydis. Joint learning of probabilistic and geometric shaping for coded modulation systems. *arXiv:2004.05062*, 2020.

[131]J. A. C. Weideman and B. M. Herbst. Split-step methods for the solution of the nonlinear Schrodinger equation. *SIAM Journal on Numerical Analysis*, 23(3):485–507, 1986.

[132]T. Uhlemann, S. Cammerer, A. Span, S. Dörner, and S. ten Brink. Deeplearning autoencoder for coherent and nonlinear optical communication. *arXiv:2006.15027*, 2020.

[133]R. Hunger. Floating point operations in matrix-vector calculus. *Technical report. Technische Universität München (2007). Accessed on: Aug. 21, 2020. [Online]. Available: https://mediatum.ub.tum.de/doc/625604/625604*.