Machine learning techniques to mitigate nonlinear impairments in optical fiber system

Pedro Jorge Freire de Carvalho Souza

Aston University

Doctor of Philosophy

December 2022

Aston University

Machine learning techniques to mitigate nonlinear impairments in optical fiber system

Pedro Jorge Freire de Carvalho Souza

Doctor of Philosophy, December 2022

The upcoming deployment of 5/6G networks, online services like 4k/8k HDTV (streamers and online games), the development of the Internet of Things concept, connecting billions of active devices, as well as the high-speed optical access networks, impose progressively higher and higher requirements on the underlying optical networks infrastructure. With current network infrastructures approaching almost unsustainable levels of bandwidth utilization/ data traffic rates, and the electrical power consumption of communications systems becoming a serious concern in view of our achieving the global carbon footprint targets, network operators and system suppliers are now looking for ways to respond to these demands while also maximizing the returns of their investments.

The search for a solution to this predicted “capacity crunch” led to a renewed interest in alternative approaches to system design, including the usage of high-order modulation formats and high symbol rates, enabled by coherent detection, development of wideband transmission tools, new fiber types (such as multi-mode and –core), and finally, the implementation of advanced digital signal processing (DSP) elements to mitigate optical channel nonlinearities and improve the received SNR. All aforementioned options are intended to boost the available optical systems’ capacity to fulfill the new traffic demands.

This thesis focuses on the last of these possible solutions to the “capacity crunch," answering the question: “How can machine learning improve existing optical communications by minimizing quality penalties introduced by transceiver components and fiber media nonlinearity?". Ultimately, by identifying a proper machine learning solution (or a bevy of solutions) to act as a nonlinear channel equalizer for optical transmissions, we can improve the system’s throughput and even reduce the signal processing complexity, which means we can transmit more using the already built optical infrastructure.

This problem was broken into four parts in this thesis: i) the development of new machine learning architectures to achieve appealing levels of performance; ii) the correct assessment of computational complexity and hardware realization; iii) the application of AI techniques to achieve fast reconfigurable solutions; iv) the creation of a theoretical foundation with studies demonstrating the caveats and pitfalls of machine learning methods used for optical channel equalization. Common measures such as bit error rate, quality factor, and mutual information are considered in scrutinizing the systems studied in this thesis. Based on simulation and experimental results, we conclude that neural network-based equalization can, in fact, improve the channel quality of transmission and at the same time have computational complexity close to other classic DSP algorithms.

Keywords: Coherent Optical communications, Nonlinear Equalization, Machine Learning, Computational Complexity

To my beautiful wife Eliza

Acknowledgements

I would like to thank my supervisor, Professor Sergei Turitsyn, for everything. This work is the result of his encouragement and personal support, insightful discussions, brilliant suggestions, and ideas. The environment he has made for me was by no means more than what I expected and had seen. What I have learned from him is beyond science and engineering.

I would also like to thank my associate supervisor and great friend, Dr. Yaroslav Prilepsky. He is involved in all the details of all works in this thesis and more. His constant support and suggestions made this work possible, especially motivating me through difficult times in the COVID-19 pandemic.

I also would like to thank all my friends and colleagues at Aston Institute of Photonic Technologies who make studying enjoyable.

I would also want to thank my friend and brother Matheus Sena, who has always inspired me to be a better person during our almost fifteen years of academic collaboration and friendship.

Finally, a special thanks to my wife (Eliza), my brother (Joao), my grandmother (Margarida), my mother (Flaviana), and my mother-in-law and father-in-law (Irene and Ariovaldo) for their unconditional love, encouragement, and undivided attention, stimulating me intellectually and emotionally.

Table of contents

Nomenclature

Acronyms / Abbreviations

1D	One-dimensional
ADC	Analog-to-Digital Converter
AE	Autoencoder
ANN	Artificial Neural Network
ASE	Amplified Spontaneous Emission
AWGN	Additive White Gaussian Noise
B2B	Back-to-Back
BCE	Binary Cross Entropy
BER	Bit Error Rate
CCE	Categorical Cross Entropy
CD	Chromatic Dispersion
CDC	Chromatic Dispersion Compensation
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRNN	Convolutional Recurrent Neural Network
DAC	Digital-to-Analog Converter
DBP	Digital Back-Propagation

DP	Dual-Polarization
DSP	Digital Signal Processing
EDFA	Erbium-doped fiber amplifier
EVM	Error Vector Magnitude
FWM	Four-Wave Mixing
Gbps	Billions of bits per second
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HLS	High Level Synthesis
IFWM	Intra-channel four wave mixing
IM/DD	Intensity Modulated/Direct Detection
ISI	Inter-Symbol Interference
IXPM	Intra-channel cross-phase modulation
LDBP	Learned digital backpropagation
LEAF	Large Effective Area Fiber
LMMSE	Linear Minimum Mean-Square Estimator
LO	Local oscillator
LSFR	Linear Feedback Shift Register
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MI	Mutual Information
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLC	Nonlinear Compensantion

NLP	Natural Language Processing
NLSE	Nonlinear Schrödinger equation
NN	Neural Network
Norm	Normalization
NPDE	Nonlinear Partial Differential Equations
PB-NLC	perturbation theory-based NLC
PDF	Probability Density Function
PDM	Polarisation-Division Multiplexing
PMD	Polarisation-Mode Dispersion
PS	Probabilistic Shaping
QAM	Quadrature Amplitude Modulation
QoT	Quality of Transmission
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SC	Single-Channel
SELU	Scaled Exponential Linear Unit
SMF	Single-Mode Fiber
SNR	Signal-to-Noise Ratio
SOTA	State-of-the-Art
SPM	Self-Phase Modulation
SSMF	Standard Single-Mode Fiber

Tanh	Hyperbolic Tangent
TL	Transfer Learning
TPUs	Tensor Processing Units
TWC	True Wave Classic Fiber
WDM	Wavelength-division multiplexed
XPM	Cross-Phase Modulation

List of figures

Chapter 1

Introduction

1.1 Fiber-optic communication

In today's world, there is always a pressing need for improved, more reliable methods of information transmission. Our ability to send a huge amount of information over long distances is the basis for many things in modern life, from big data and cloud computing to the Internet of Things and Industry 4.0, as well as high-quality streaming services and video calls. Since the start of the COVID-19 pandemic in 2020, internet usage has skyrocketed as working from home and doing schoolwork online have become the norm, entailing high-quality conference calls, cloud storage, and entertainment streaming. Most of this enormous traffic was carried by the fiber-optic communication network, widely regarded as the internet's "backbone." Across cities, countries, and continents, optical fiber links are the only method that can reliably and affordably transport a massive amount of data. However, there will be a significant challenge for future generations of optical fiber networks to accommodate the ever-increasing volume of global internet traffic. Although huge efforts are being made to develop and create new forms of fiber to support higher capacity transmission, specifically to allow many transmission paths on the same fiber (Space Division Multiplexing [10]), many of these novel technologies require complete replacement of the current infrastructure [11].

In the early days of fiber-optic communication, the data rate and distance were severely hampered by fiber loss. Transpacific optical lines were formerly unaffordable, but the development of the erbium-doped fiber amplifier (EDFA) [12] to compensate for the loss changed all of that. In the 1990s, new systems evolved with a steadily increasing information rate by taking advantage of the five possible dimensions to multiplex data: time, space, wavelength, polarization, and quadrature. With the right combination of novel codes, dispersion-managed links, and polarization-division multiplexing (PDM) [13], a wavelength-division multiplexed (WDM) system with intensity-modulated/direct detection (IM/DD) configuration could achieve communication rates of up to 1 Tb/s in a laboratory demonstration [14–16]. In 1989, the market saw the release of the first commercial WDM system with two wavelengths running at 1.7 Gb/s each. A dispersion-controlled link, implemented in place of regular SMF, allowed for a data transfer rate of up to 20 Gb/s [17]. At the beginning of the 2000s, commercial systems could transport 40 wavelengths at 10 Gb/s, for a total data throughput of 400 Gb/s [18].

Optical communications took another huge step forward with the development of coherent receivers [19]. Even though an IM/DD communication system can provide a high data rate, the coherent technology allowed for new approaches to handling light at the receiver and paved the way for other ideas to fix problems in fiber communication that emerged from other fields of linear communications. Benefiting from both the low bandwidth requirements of homodyne receivers and the ease of use of analogue heterodyne receivers, digital coherent receivers are the best of both worlds. Coherent communication makes use of the signal's amplitude and phase to decode the message. To eliminate noise, the signal is sent through an optical 90-degree hybrid and combined with a local oscillator (LO). Two sets of balanced photodetectors then read out the hybrid's four optical outputs. Electrical outputs from the square-law detector are then utilized to identify the in-phase and quadrature parts of the received signal, together with their respective labels and relationships. The term "noise" primarily refers to the amplified spontaneous emission (ASE) noise that the optical amplifier introduces, the strength of which varies depending on the particular amplifier in question. Coherent receivers made it possible to use quadrature amplitude modulation and pulse-division multiplexing. Digital signal processing (DSP) also allows for the compensation of some types of chromatic dispersion, PMD, and nonlinear effects that were not possible when using IM/DD systems. Thanks to this breakthrough, the first commercial transponder was developed, boasting a 100 Gb/s [16] data throughput on a single wavelength and later reporting speeds of 800 Gb/s [20] using a real-time DSP ASIC based on probabilistic shaping and digital sub-carrier multiplexing.

It is now clear that the goal of optical communications has always been to increase throughput in order to increase spectral efficiency. This requires more advanced DSP to deal with problems like chromatic dispersion and other nonlinear distortions that occur in fibers. Most of the digital signal processing methods currently used in fiber-optic communication were originally developed for linear channels, such as wireless and twisted-copper, and are therefore not well-suited to the additive white Gaussian noise that is characteristic of linear channels (AWGN). Unlike with these linear media, an increase in signal strength in an optical fiber connection does not result in a higher bit rate that is immune to errors. At high signal powers, the system's performance is dominated by the interference introduced by nonlinearities, which increases in proportion to the signal power [21, 22]. This nonlinear interference appears to be the major barrier to fiber-optic communication. There are two basic types of nonlinear interference in fiber: signal-noise and signal-signal. Most proposed techniques aim to counteract signal-signal nonlinear interference, which is more significant than signal-noise interference. So, it is useful to categorize different nonlinear signal-signal interferences as in-band interference and out-of-band interference [23]. Self-phase modulation (SPM) is an example of in-band signal-signal interference, whereas cross-phase modulation (XPM) and four-wave mixing (FWM) are examples of out-of-band signal-signal interference that cause significant difficulties for WDM systems [24].

To understand where this nonlinear interference originates and how to mitigate it, it is necessary to review how light travels through optical fibers. The optical fiber channel is a nonlinear medium due to the dependence of its refractive index (n_{eff}) on the square of the electric field of the optical signal ($E(t,z)$), as follows:

$$n_{eff} = n_0 + n_2|E(t,z)|^2, \tag{1.1}$$

where n_0 and n_2 indicate the linear and nonlinear parts of the fiber refractive index, respectively. This is known as the "nonlinear optical Kerr effect." After a propagation distance of L, light experiences two kinds of phase shifts. One of them is a result of the linear part of the refractive index, and the other one is because of the n_2 term. Given is the total phase shift:

$$\phi = k_0 n_0 L + \phi_{NL} \tag{1.2}$$

where k_0 is a free-space wave number and ϕ_{NL} can be expressed by:

$$\phi_{NL}(t,z) = j\gamma \int_0^z |E(t,z')|^2 dz' = j\gamma |E(t,0)|^2 L_{eff} \tag{1.3}$$

L_{eff} is defined as effective length, and it is equal to $\frac{1-e^{\alpha z}}{\alpha}$, where α is the fiber loss coefficient. Also, $\gamma = \frac{k_0 n_2}{A_{eff}}$ is the fiber nonlinearity coefficient, and A_{eff} is the fiber effective area. The nonlinear Schrödinger equation (for single polarization) and the Manakov equation (for dual polarization) can be used to estimate how light travels through the fiber [25]. Consider a single polarization transmission case whose propagation is defined by:

$$\frac{\partial E(t,z)}{\partial z} = -j\frac{\beta_2}{2}\frac{\partial^2 E(t,z)}{\partial t^2} - \frac{\alpha}{2}E(t,z) + j\gamma|E(t,z)|^2 E(t,z) \tag{1.4}$$

where β_2 is the dispersion coefficient of the fiber. As can be seen from the similarities with ϕ_{NL}, the last term of this equation is responsible for nonlinear distortion. In WDM systems, the “nonlinear challenge” is even worse because different channels affect each other even though they are at different frequencies. An important remark here is the fact that real-world systems are notoriously difficult to analyze in their entirety due to the sheer number of variables involved in determining the complex interplay between the signal and amplifier-generated noise (ASE). However, the amplitude of the primary signal is significantly greater than that of the noise, so signal-signal interference provides an excellent estimate of total interference. So now, let us focus only on the signal-to-signal interferences. Consider three optical signals going through three channels in the same optical fiber (three different wavelengths). The total electric field can be expressed as:

$$E(t,z) = E_0(t,z) + E_1(t,z) + E_2(t,z) \tag{1.5}$$

Replacing Eq. 1.5 in Eq. 1.4 and ignoring α for understandability purposes, the below equation is found:

$$\frac{\partial(E_0+E_1+E_2)}{\partial z} = -j\frac{\beta_2}{2}\frac{\partial^2(E_0+E_1+E_2)}{\partial t^2} + j\gamma|E_0+E_1+E_2|^2(E_0+E_1+E_2) \tag{1.6}$$

Assuming a perturbation approach and E_0 as the channel of interest, a decoupled equation is found as follows:

$$\frac{\partial E_0}{\partial z} = -j\frac{\beta_2}{2}\frac{\partial^2 E_0}{\partial t^2} + 2j\gamma|E_0|^2E_0 + 2j\gamma(|E_1|^2+|E_2|^2)E_0 + j\gamma {E_1}^2{E_2}^* \tag{1.7}$$

As can be seen from Eq. 1.7, there are three terms that contribute to the nonlinear distortion. The first term just depends on the power of the assumed channel; the second term is dependent on the power of two other channels; and the last term is a cross-product of the two interfering channels. These interferences are the previously mentioned SPM, XPM, and FWM, respectively.

To elaborate further on each of these signal-signal interferences, the SPM is a nonlinear phase variation caused by the light's intensity-dependent refractive index. This causes frequency chirping, which can interact with dispersion in the fiber to result in pulse broadening. The greater the power input, the greater the chirping [26].

In multi-channel systems, the optical fiber's refractive index is affected not only by the intensity of the target channel but also by the intensity of the other channels. This is referred to as "XPM." XPM, like SPM, results in nonlinear phase shifts [26], but XPM is also inversely proportional to channel spacing and increases with the number of channels in the WDM transmission. The other multi-channel interference is FWM, also known as crosstalk, which causes energy transfer between channels. Power can thus be transferred from one channel to another. As a result, FWM decreases as dispersion increases [26].

Over the last decade, these particular aspects of fiber propagation have been widely investigated, together with mitigation techniques, in both the optical and digital domains e.g., digital back-propagation [27], Kalman Equalizer [28], Volterra series nonlinear equalizers [29], optical phase conjugation (OPC) [30] to name a few. A brief discussion on the advantages and disadvantages of these conventional signal processing methods is included in Chapter 2. Although these methods have been shown to be very effective, they have a number of disadvantages. One of the most advanced nonlinear compensation approaches, for example, is digital back-propagation, which requires a massive amount of processing power and is difficult to implement in practice. In this thesis, a new set of solutions will be presented that use machine learning, specifically deep neural networks, to mitigate component and fiber nonlinearity and increase channel capacity.

1.2 Contributions of the Thesis

Scientists have been looking into optical communication systems that use equalizers based on neural networks for a number of years. Although numerical simulations and experimental demonstrations have yielded some promising results, determining the entire potential of such solutions and their limitations remains an unanswered subject for their industrial application in the next DSP generation. In this thesis, for the first time in the coherent fiber-optic community, a large-scale investigation is proposed, proposing new physical-inspired neural network architectures, investigating the potential of black box solutions, developing methods to increase the operational flexibility of such solutions to meet the needs of optical communications, and evaluating the performance versus complexity trade-off of many neural network solutions, which pave the way for a full-scale study proposing methods to lower their computational complexity to industrially-attractive levels.

The contribution of this thesis can be summarized as follows:

- Based on the channel model that treats Kerr nonlinearity as the dominant distortion, it is introduced a new design for a complex-valued neural network to perform equalization at the receiver.

- The Bayesian optimizer was first proposed as a means of determining the optimal hyper-parameters for the NN equalizers, which would simultaneously reduce computational complexity and improve transmission performance.

- Here, it is properly explained how the transfer learning technique was applied for the first time to the problem of signal distortion mitigation in optical coherent communications. The novel contribution is to propose various forms of transfer learning to reduce the training time and dataset size for equalizer training by combining a deep understanding of the channel propagation model (Manakov Equation) with the design of a neural network equalizer.

- Using both synthetic data generated through numerical simulation and experimental data from a benchmark transmission system, it is analyzed and compared to a wide variety of black-box NN-based methods.

- From a software to a hardware perspective, it is presented for the first time the equations and appropriate computational complexity metrics for a zoo of neural network layers.

- In order to lessen the burden on the computer's processing power, it is also considered a detailed description and analysis of various compression techniques that can be used in NN-based equalizers.

- This is the first time a Recurrent-based equalizer is being implemented on FPGA for coherent transmission.

- Addressing the NN-based soft-demapping in coherent optical communication, it is compared the performance and training quirks of the regression and classification predictive models for the first time, highlighting their statistical and machine learning drawbacks.

- For the first time, a systematical study is presented about the typical misconceptions and misinterpretations that arise when using ML techniques for channel equalization in coherent optical communications.

The following publications are the results of the current research and directly or indirectly relate to different chapters of this thesis:

1. **Freire, P.J.**, Napoli, A., Spinnler, B., Costa, N., Turitsyn, S.K. and Prilepsky, J.E., 2021. *[Invited] Neural Networks-based Equalizers for Coherent Optical Transmission: Caveats and Pitfalls*. IEEE Journal of Selected Topics in Quantum Electronics, doi: 10.1109/JSTQE. 2022.3174268.

2. **Freire, P.J.**, Neskornuik, V., Napoli, A., Spinnler, B., Costa, N., Khanna, G., Riccardi, E., Prilepsky, J.E. and Turitsyn, S.K., *Complex-valued neural network design for*

mitigation of signal distortions in optical links. Journal of Lightwave Technology, vol. 39, no. 6, pp. 1696-1705, 15 March15, 2021, doi: 10.1109/JLT.2020.3042414.

3. **Freire, P.J.**, Osadchuk, Y., Spinnler, B., Napoli, A., Schairer, W., Costa, N., Prilepsky, J.E. and Turitsyn, S.K., *Performance versus complexity study of neural network equalizers in coherent optical systems*. Journal of Lightwave Technology, vol. 39, no. 19, pp. 6085-6096, Oct.1, 2021, doi: 10.1109/JLT.2021.3096286.

4. **Freire, P.J.**, Abode, D., Prilepsky, J.E., Costa, N., Spinnler, B., Napoli, A. and Turitsyn, S.K., *Transfer learning for neural networks-based equalizers in coherent optical systems*. Journal of Lightwave Technology, vol. 39, no. 21, pp. 6733-6745, 1 Nov.1, 2021, doi: 10.1109/JLT.2021.3108006.

5. **Freire, P.J.**, Neskornuik, V., Napoli, A., Spinnler, B., Costa, N., Prilepsky, J.E., Riccardi, E. and Turitsyn, S.K., 2020, December. *Experimental Verification of Complex-Valued Artificial Neural Network for Nonlinear Equalization in Coherent Optical Communication Systems*. In 2020 European Conference on Optical Communications (ECOC) (pp. 1-4). IEEE.

6. **Freire, P.J.**, Osadchuk, Y., Spinnler, B., Schairer, W., Napoli, A., Costa, N., Prilepsky, J.E. and Turitsyn, S.K., 2021, June. *Experimental study of deep neural network equalizers performance in optical links*. In Optical Fiber Communication Conference (pp. M3H-2). Optical Society of America.

7. **Freire, P.J.**, Abode, D., Prilepsky, J.E. and Turitsyn, S.K., 2021, July. *Power and Modulation Format Transfer Learning for Neural Network Equalizers in Coherent Optical Transmission Systems*. In Signal Processing in Photonic Communications (pp. SpM5C-6). Optical Society of America.

8. **Freire, P.J**., Spinnler, B., Abode, D., Prilepsky, J.E., Ali, A., Costa, N., Schairer, W., Napoli, A., Ellis, A.D. and Turitsyn, S.K., 2022, March. *Domain Adaptation: the Key Enabler of Neural Network Equalizers in Coherent Optical Systems*. In 2022 Optical Fiber Communications Conference and Exhibition (OFC) (pp. 1-3). IEEE.

9. **Freire, P.J.**, Prilepsky, J.E., Osadchuk, Y., Turitsyn, S.K. and Aref, V., 2022. *Deep Neural Network-aided Soft-Demapping in Optical Coherent Systems: Regression versus Classification*. IEEE Transactions on Communications; doi: 10.1109/TCOMM.2022.3213284.

10. **Freire, P.J.**, Osadchuk, Y., Napoli, A., Spinnler, B., Schairer, W., Costa, N., Prilepsky, J.E. and Turitsyn, S.K., 2021, October. *Experimental Evaluation of Computational*

Complexity for Different Neural Network Equalizers in Optical Communications. In Asia Communications and Photonics Conference (pp. M5H-5). Optical Society of America.

11. **Freire, P.J.**, Anderson, M., Spinnler, B., Bex, T., Prilepsky, J. E., Eriksson, T. A., Costa, N., Schairer, W., Blott, M., Napoli, A., Turitsyn, S. K., 2022. *[Top Score] Towards FPGA Implementation of Neural Network-Based Nonlinearity Mitigation Equalizers in Coherent Optical Transmission Systems*. In 2022 European Conference on Optical Communications (ECOC).

12. **Freire, P.J.**, Srivallapanondh, S., Napoli, A., Prilepsky, J.E. and Turitsyn, S.K, *Computational Complexity Evaluation of Neural Network Applications in Signal Processing*. Submitted in May 2022 to IEEE Transactions on Signal Processing, Available in arXiv:2206.12191.

13. **Freire, P.J.**, Antonio Napoli, Diego Arguello Ron, Bernhard Spinnler, Michael Anderson, Wolfgang Schairer, Thomas Bex, Nelson Costa, Sergei K. Turitsyn, Jaroslaw E. Prilepsky, *Reducing Complexity of Neural Networks for Optical Channel Equalization: From Algorithms to Implementation*. Submitted in September 2022 to Journal of Lightwave Technology, Available in arxiv:2208.12866.

14. **Freire, P.J.**, Sasipim Srivallapanondh, Michael Anderson, Bernhard Spinnler, Thomas Bex, Tobias A. Eriksson, Antonio Napoli, Wolfgang Schairer, Nelson Costa, Jaroslaw E. Prilepsky, Sergei K. Turitsyn, *[Invited] Implementing Neural Network-Based Equalizers in a Coherent Optical Transmission System Using Field-Programmable Gate Arrays*. Submitted in October 2022 to Journal of Lightwave Technology, Available in arxiv: 2212.04703.

15. Ron, D.A., **Freire, P.J.**, Prilepsky, J.E., Kamalian-Kopae, M., Napoli, A. and Turitsyn, S.K., 2022. *Experimental Implementation of a Neural Network Optical Channel Equalizer in Restricted Hardware Using Pruning and Quantization*. Scientific Reports, 12(1), pp.1-12.

16. Sedov, E.V., **Freire, P.J.**, Seredin, V.V., Kolbasin, V.A., Kamalian-Kopae, M., Chekhovskoy, I.S., Turitsyn, S.K. and Prilepsky, J.E., 2021. *Neural networks for computing and denoising the continuous nonlinear Fourier spectrum in focusing nonlinear Schrödinger equation*. Scientific Reports, 11(1), pp.1-15.

17. Sena, M.R., **Freire, P.J.**, Coelho, L.D., Santos, A.F., Napoli, A. and Almeida Jr, R.C., 2022. *Novel evolutionary planning technique for flexible-grid transmission in optical networks*. Optical Switching and Networking, 43, p.100648.

18. Sedov, E., **Freire, P.J.**, Chekhovskoy, I., Turitsyn, S. and Prilepsky, J., 2021, September. *Neural Networks For Nonlinear Fourier Spectrum Computation*. In 2021 European Conference on Optical Communication (ECOC) (pp. 1-4). IEEE.

19. Neskorniuk, V.,**Freire, P.J.**, Napoli, A., Spinnler, B., Schairer, W., Prilepsky, J.E., Costa, N. and Turitsyn, S.K., 2020, December. *Simplifying the Supervised Learning of Kerr Nonlinearity Compensation Algorithms by Data Augmentation.* In 2020 European Conference on Optical Communications (ECOC) (pp. 1-4). IEEE.

20. Liu, Y., Sanchez, V., **Freire, P.J.**, Prilepsky, J.E., Koshkouei, M.J., Higgins, M.D., 2022. *Attention-Aided Partial Bidirectional RNN-Based Nonlinear Equalizer in Coherent Optical Systems*. Optics Express, Optica (former OSA), May 2022.

1.3 Organisation of the thesis

This thesis is organised as follows.

- In Chapter 2, it is presented an introduction to the equalization problem in digital communications. Then, it discussed the traditional methods used to equalize the optical channel in an effort to reduce the impact of nonlinearities introduced during optical transmission. Also, it is discussed the various types of neural network-based equalizers, categorizing them as either data-driven or model-driven. Finally, it presents an overview of the most common neural network layers used in machine learning, categorizing them into feedforward and recurrent types.

- In Chapter 3, it is examined the performance aspect for designing a NN-based equalizer. First, it is presented a model-driven technique built by creating a new complex-value neural network architecture inspired by certain physical elements of nonlinear optical transmission. In addition, it is demonstrated how the Bayesian optimizer can be utilized to determine the NN-based equalizer architecture that minimizes the BER after equalization. Simulations and experiments were used to validate the new complex-value neural network architecture in this instance. In this chapter, it is also discussed data-driven solutions, presenting for the first time the biLSTM+CNN structure for optical equalization and comparing it with a zoo of NN topologies for both single-channel and WDM transmission. In this last investigation, it is confirmed that the proposed NN-based equalizer can partially recover both the SPM and the XPM during transmission.

- In Chapter 4, it is examined the computational complexity aspect for designing a NN-based equalizer. First, it provides a systematic approach for assessing and comparing the computational complexity of neural network layers in digital signal processing. It provides and links four software-to-hardware complexity measures, defining how the different complexity metrics relate to the layers' hyper-parameters. Second, it presents the results of the comparative performance-versus-complexity analysis for the several types of artificial neural networks (NNs) used for nonlinear channel equalization in coherent optical communication systems. The comparison is carried out using an experimental set-up with the transmission dominated by the Kerr nonlinearity and component imperfections. Then, a new methodology is proposed that allows for the low-complexity development of neural network (NN) based equalizers for the mitigation of impairments in high-speed coherent optical transmission systems using cutting-edge strategies such as quantization, weight clustering, and pruning. Additionally, it evaluates the influence these deep model compression approaches have on the performance of each NN equalizer. Finally, it is demonstrated the offline FPGA realization of both recurrent and feedforward neural network (NN)-based equalizers for nonlinearity compensation in coherent optical transmission systems. In this context, it presents a realization pipeline showing the conversion of the models from Python libraries to the FPGA chip synthesis and implementation.

- In Chapter 5, it examined the flexibility aspect of designing a NN-based equalizer. It demonstrates that by using well-developed techniques based on the concept of transfer learning, it can efficaciously retrain NN-based equalizers to adapt to the changes in the transmission system, using just a fraction (down to 1%) of the initial training data or epochs. It evaluates the capability of transfer learning to adapt the NN to changes in the launch power, modulation format, symbol rate, or even fiber plants (different fiber types and lengths). Also, it underlines the specific peculiarities that occur when transferring the learning in coherent optical communication systems and draws the limits for the transfer learning efficiency. This chapter also presented the effectiveness of transfer learning for the fast adaptation of NN architectures to different transmission regimes and scenarios, paving the way for engineering flexible and universal solutions for nonlinearity mitigation.

- Chapter 6 offers a deep, multi-faceted investigation of important issues and typical design pitfalls linked to the construction of efficient neural networks (NN) based nonlinear channel equalizers in coherent optical communication systems. It starts by clarifying the metrics used to evaluate the equalizers' performance, relating them to the loss functions employed in the training of the NN equalizers. The relationships between the channel propagation model's accuracy and the performance of the equalizers are addressed and quantified. Next, it assesses the impact of the order of the pseudo-random bit sequence used to generate the – numerical and experimental – data as well as of the DAC memory limitations on the operation of the NN equalizers both during the training and validation phases. Finally, it examines the critical issues of overfitting limitations, the difference between using classification instead of regression, and batch-size-related peculiarities.

- The thesis concludes with some discussions and suggested future works in Chapter 7.

Chapter 2

Neural Network Algorithms for Optical Channel Equalization

2.1 The Equalization Problem in Digital Communications

Equalizers play a crucial role in high-bandwidth digital communication systems. Their purpose is to reduce or eliminate channel interference and distortion so that the transmitted information, or information at the channel's input, may be recovered. Because of the rapid changes in communication technology over the past 70 years, a plethora of methods have emerged. The three classic linear equalizer algorithms in the communications field are the Zero Forcing Algorithm, Least Mean Square (LMS) Algorithm, and Recursive Least Square Algorithm [31, 32].

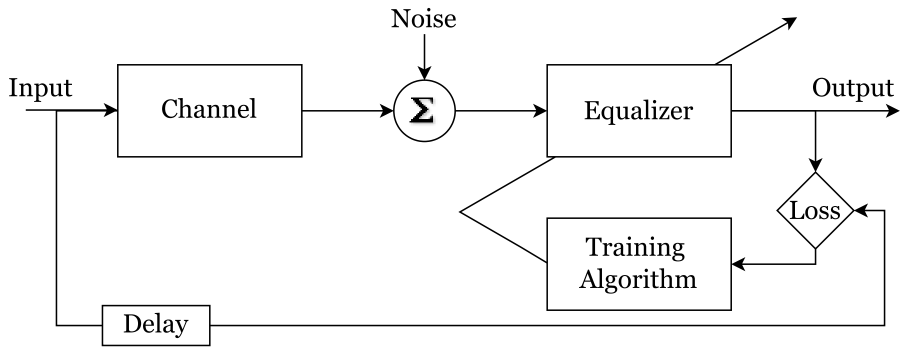


Fig. 2.1 Schematic diagram of a digital communication system with a channel equalizer.

To better understand the adaptive equalization task, we can separate it into two steps: training and inference. For this demonstration, all signals are assumed to have a digital baseband representation. Fig. 2.1 shows the schematic of a digital communication system with an equalizer at the receiver side. Considering a coherent transmission case, the symbol $\{t_k\}$ denotes a sequence of T-spaced complex symbols of an M-QAM constellation in which both in-phase component $\left\{t_{k,1}\right\}$ and quadrature component $\left\{t_{k,Q}\right\}$ take one of the values $\{\pm 1, \pm 2, \ldots, \pm(\sqrt{M}-1)\}$, where $1/T$ denotes the symbol rate and k denotes the discrete time index. The combined effect of the transmitter-side filter and transmission medium is included in the "channel", which can cause linear and nonlinear interferences in the originally transmitted signal t. An FIR filter is a commonly used model for a linear dispersive channel, and its output at the k th instant is as follows:

$$a_k = \sum_{i=0}^{N_h-1} h_i t_{k-i},$$

where h_i denotes the FIR filter weights and N_h denotes the filter order. Considering the channel to be a nonlinear one, the channel block introduces nonlinearity to the filter output. Given is the discrete output of the nonlinear channel:

$$b_k = \psi\left\{a_k, a_{k-1}, a_{k-2}, \ldots, a_{k-N_\mathrm{h}+1}; h_0, h_1, \ldots, h_{N_\mathrm{h}-1}\right\},$$

where $\psi\{\cdot\}$ is a nonlinear function generated by the channel block. The channel output is also assumed to be corrupted by an additive white Gaussian noise q_k with a variance of σ^2. After passing through the nonlinear channel and being mixed with additive noise, the transmitted signal t_k arrives at the receiver, which is denoted by r_k. The received signal at the k th time instant is given by $r_k = r_{k,l} + j r_{k,Q}$, where $r_{k,l}$ and $r_{k,Q}$ are the in-phase and quadrature components, respectively.

The purpose of the equalizer attached at the receiver side is to recover the transmitted sequence t_k or its delayed version $t_{k-\tau}$, where τ is the propagation delay associated with the physical channel. The inferential step reveals that the equalizer receives the corrupted sequence r_k and its delayed variants as input and generates the resultant equalized sequence y_k. A loss function, often performed by the mean squared error [$Loss = MSE(d,y)$], can be computed given the target output $(d_k = t_k - \tau)$ and the equalized symbol y_k. At this point, we enter the training phase, where we can use the loss function obtained from the equalized symbols in the inference phase to adjust the equalizer's filter weights and, thus, reduce the loss using an adaptive algorithm (e.g., LMS algorithm). Once training is complete, the weights are frozen and used for the last inference to estimate the transmission sequence.

2.2 Classical Algorithms for Equalization in Optical Communications

In order to boost fiber capacity, the nonlinear equalization field has become a hot topic due to the fact that, as discussed in Chapter 1, Kerr nonlinearities are the principal source of channel deterioration in fiber optic channels, especially when employing higher-order modulation formats. A few of the sophisticated optical signal equalizers created to counteract signal distortions brought on by fiber nonlinearities include first-order perturbation-based compensation, digital backpropagation, the Volterra series transfer function, pulse shaping, machine learning techniques, and others. This section provides a brief overview of some classic nonlinear compensation techniques used in fiber optic communication systems.

2.2.1 Digital back propagation

Digital backpropagation (DBP) is considered one of the most effective and widely used methods to combat fiber nonlinearity and equalize signal distortion [33]. After solving the NLSE with the asymmetric Split-Step Fourier Method (SSMF), Ip *et al.* showed how to compensate for distortions brought on by dispersion and nonlinearities [27]. In a nutshell, the idea of the DBP technique is to digitally model a fictitious fiber with exactly opposite characteristics when compared to the real fiber used for transmission. DBP is particularly efficient because, in theory, it can completely undo the deterministic distortion brought on by signal-to-signal interference. The authors recommended seeing the fiber as the concatenation of nonlinear and linear sections, based on the fact that the fiber nonlinearities are greatest when the power is greatest, which is often after lumped amplification. Most of the distortion in the linear region is due to dispersion, which manifests itself in the frequency domain as a multiplication of phases. Kerr nonlinearity, which is phase multiplication in the time domain, is the principal source of distortion in the nonlinear region. The output of the linear compensation section in mathematical terms using the SSFM algorithm is as follows:

$$E^{\mathrm{CD}}_{x/y}(z,t) = iFFT\left(FFT(E_{x/y}(z,t))\exp\left(-jh\left(\frac{\alpha}{2}+\frac{\beta_2}{2}\omega^2\right)\right)\right),$$

where h is the length of each step, ω is the frequency variable, and z is the current transmission distance. Following that, the nonlinear section is carried out in the time-domain as follows:

$$E_{x/y}(z+h,t) = E^{\mathrm{CD}}_{x/y}(z,t)\exp\left(-j\varphi\gamma' h\left(\left|E^{\mathrm{CD}}_x\right|^2+\left|E^{\mathrm{CD}}_y\right|^2\right)\right),$$

where $0 < \varphi < 1$ is a real-valued optimization parameter. This procedure can theoretically undo all distortions if the step size is infinitesimally short and there is no noise present in the algorithm [27].

While DBP is excellent at addressing deterministic distortions, it is not able to make up for stochastic impairments brought on by the intermixing of signal and ASE noise. Another drawback of DBP is its complexity. To get a good approximation of the inverse NLSE using DBP, it is necessary to reduce the step size, which considerably increases the complexity of the method. Furthermore, multichannel DBP is required to account for inter-channel distortions in a WDM system (for instance, to mitigate XPM interferences). This adds much more difficulty to enforcing DBP. Consequently, despite its theoretical efficacy, DBP can only achieve remarkable performance at the expense of very expensive computational resources. As a result, single-channel DBP (SC-DBP) has received more attention than its multichannel counterpart since it is seen as more practical in light of the existing hardware constraints [33][1]. SC-DBP solely corrects intra-channel nonlinearity, such as SPM, because it only back-propagates a single wavelength channel [37, 38].

2.2.2 Volterra Equalizer

The Volterra series transfer function (VSTF) can be effectively used to model fiber nonlinearity effects. The Volterra series is a functional expansion widely used to model nonlinear dynamics. Peddanarappagari et al. [39] proposed using the series to solve the NLSE and extract a higher-order nonlinear transfer function inside the SSMF. In this technique, after modeling the optical channel as a series of VSTFs, the pth order is adopted to design the inverse VSTF (IVSTF) kernels. In a nutshell, with $y(n)$ and $\tilde{y}(n)$ representing system input and output, respectively, the p th-order discrete-time Volterra series with M_p memory taps for order p is given as [40]:

$$\tilde{y}(n) = \sum_{p=1}^{P} \sum_{m_1=0}^{M_1} \cdots \sum_{m_p=0}^{M_p} h_p\left(m_1, \cdots, m_p\right) \prod_{k=1}^{p} y\left(n - m_k\right) \tag{2.1}$$

This is the most complete model for nonlinearities with memory, as the p th-order Volterra kernel $h_p\left(m_1, \cdots, m_p\right)$ includes all possible combinations of a product of p time shifts of the input signal up to memory depth m_p. These IVSTF kernels may be used to compensate for

[1]Using a multichannel-DBP (MC-DBP) [34–36] in WDM systems is one technique to combat the inter-channel nonlinear distortions, such as XPM and FWM, caused by the co-propagating neighbor channels. All the WDM channels are back-propagated by the MC-DBP. However, the MC-DBP can only be used for point-to-point connections at the moment due to its computational complexity, which makes it difficult to implement without access to supercomputers capable of massively parallel processing.

the Kerr nonlinearity of the fiber and its interaction with CD. The first and third orders of such Volterra kernels in this scenario serve to exemplify the fiber's nonlinearities. In addition, one of the properties of the nonlinear equalizer based on the Volterra series is the ability to perform the compensatory operation in parallel. Combining the outputs of each nonlinear stage with the linear portion's output results in the output-compensated signal. This reduces the computational complexity compared to DBP.

Another advantage is that nonlinearity equalization with the Volterra series transfer function also works well with WDM because it can be done in the frequency domain and thus separates different channels [39]. However, as the channel memory is increased, the number of Volterra series coefficients grows significantly, and this method does not scale well [41].

2.2.3 Perturbation-based NLC

In the first-order perturbation theory-based NLC (PB-NLC) method, the time-domain solution of the NLSE is approximated by considering the nonlinearity distortion as the first-order perturbation term [42]. The field inside the fiber is considered the sum of the solutions of linear propagation and perturbed terms caused by nonlinearities: $E(t,z) = E_0(t,z) + \Delta E(t,z)$ where $E_0(t,z)$ is the zeroth-order term (solution of the linear propagation) and $\Delta E(t,z)$ is the first-order perturbation term caused by nonlinearities. The PB-NLC method simplifies two things in order to express the perturbation expansion coefficients analytically: the full electrical compensation of CD at the receiver and the Gaussian shape assumption for input pulses.

According to the first-order theory, three input Gaussian pulses at time indices T_m, T_l, T_n interact nonlinearly and generate a ghost pulse. Without loss of generality, the nonlinear distortion field induced at index $k = 0$, i.e., $l = m + n$ is calculated at $t = 0$ by considering the symbol rate operation, and it can be further simplified as:

$$\begin{aligned}\Delta E_{x/y}(L,t) = j\frac{8}{9}\gamma P_0^{3/2} \sum_m \sum_n \Big[& a_{m,x/y} a^*_{m+n,x/y} a_{n,x/y} \\ & + a_{m,y/x} a^*_{m+n,y/x} a_{n,x/y} \Big] \mathbf{C}_{m,n}\end{aligned} \tag{2.2}$$

where $*$ represents the complex conjugate operation, $a_{m,x/y}$ is the symbol complex amplitudes at time m for the x and y polarization, P_0 is the peak power and $\mathbf{C}_{m,n}$ is the first-order perturbation coefficient matrix, which is given as:

$$\mathbf{C}_{m,n} = \begin{cases} \frac{\tau^2}{\sqrt{3}|\beta_2|} \int_0^L dz \frac{1}{\sqrt{\tau^4/(3\beta_2^2)+z^2}}, & m=n=0 \\ \frac{\tau^2}{\sqrt{3}|\beta_2|} \frac{1}{2} E_1\left(\frac{(n-m)^2 T^2 \tau^2}{3|\beta_2|^2 L^2}\right), & m \text{ or } n = 0 \\ \frac{\tau^2}{\sqrt{3}|\beta_2|} E_1\left(-j\frac{mnT^2}{\beta_2 L}\right), & m \neq n \neq 0, \end{cases} \tag{2.3}$$

where $E_1(x) = \int_x^\infty \frac{e^{-t}}{t} dt$ is the exponential integral function [38].

In the PB-NLC technique, the perturbation coefficients are calculated beforehand and saved in a look-up table. This perturbed term can be computed and used to compensate for the nonlinear distortions at the transmitter side or the receiver side [43, 44]. Tao *et al* demonstrated perturbation-based pre-distortion compensation in a dual-polarisation system, whose performance is comparable to that of backpropagation [43]. The nonlinear perturbation coefficients derived in [43] represent the effects of intra-channel (FWM and XPM) and they are fairly complex since they entail a 2-D summation of the product of the transmitted symbol sequence. However, since these coefficients depend on pulse shape, fiber properties, and fiber lengths, the computation can be done for the link configurations and stored in a lookup table [43]. Furthermore, the authors showed that with QPSK symbols, the multiplication can be replaced by a logical operation, thus significantly reducing the complexity [43]. Unfortunately, this method does not apply to higher-order QAM. To reduce the complexity, many methods have been proposed, including aggressive quantization on perturbation coefficients [45], symmetric electronic dispersion compensation and root-raised-cosine pulse shaping [46].

While effective, most classic approaches to compensating for nonlinearities are prohibitively difficult to implement. That is why many cutting-edge approaches cannot be used in a commercial context. A compelling motivation to find an alternative strategy that can produce comparable performance results while simultaneously reducing computational complexity underpins the investigation of deep learning. In the next sections, some deep-learning approaches for nonlinearity compensation will be presented.

2.3 Data-Driven Equalizers using Neural Networks.

This section describes the main neural network (NN) structures used to create a data-driven neural network (NN) equalizer (black box). This category takes advantage of the vast number of symbols flowing across the line, which can be employed as a training data source so that the NN can be used blindly (i.e., without any prior knowledge of physics under investigation).

2.3.1 Learning Process of Neural Network Models

Before discussing the various neural network architectures used to create NN-based equalizers, it is essential to first examine the optimization approaches or the learning process for adjusting the parameters of NNs to the training datasets.

In machine learning, the common practice is to formalize the goals of the training procedure via a loss function $\mathscr{L}$ - a single overall quantitative measure of the quality of training. The loss function is defined in such a way that the desired solution should minimize its value.

Now we'll bring a more mathematically rigorous definition of loss function $\mathscr{L}$. In the following, we focus on the case of training over a labeled dataset, considered in this thesis. Let the training data $\mathscr{D}$ consist of the set of input vectors $\mathbf{x}_i = [x_{i,1}, \ldots, x_{i,n}]$, $x_{i,j} \in \mathbb{R}$ and the desired output vectors $\mathbf{y}_i^{\text{data}} = [y_{i,1}, \ldots, y_{i,n}]$, $y_{i,j} \in \mathbb{R}$, corresponding to each input $\mathbf{x}_i$, i.e., $\mathscr{D} = \left[\{\mathbf{x}_1, \mathbf{y}_1^{\text{data}}\}, \ldots, \{\mathbf{x}_N, \mathbf{y}_N^{\text{data}}\}\right]$. Next, we define the output predicted by the trained algorithm for every dataset input as $\mathbf{y}_i^{\text{model}} = f_{\text{model}}(x_i, \theta)$. Obviously, the model's predictions also depend on its parameters θ. For instance, for a neural network, the parameters are model weights and biases.

In terms of these notations, the loss function $\mathscr{L}$ can be defined as

$$\mathscr{L}(\theta, \mathscr{D}) = \frac{1}{N} \sum_{i=1}^{N} l\left(\mathbf{y}_i^{\text{data}}, f_{\text{model}}(\mathbf{x}_i, \theta)\right) \tag{2.4}$$

where $l(\mathbf{y}_i^{\text{data}}, \mathbf{y}_i^{\text{model}})$ is per-example loss function, and N is the training data cardinality. At the same time, the task of training a deep-learning model can be expressed as

$$\theta_{\text{learned}} = argmin_{\theta} \mathscr{L}(\theta, \mathscr{D}), \tag{2.5}$$

i.e., finding the set of model parameters θ_{learned} minimizing the loss function $\mathscr{L}$ given the training dataset $\mathscr{D}$.

There are several popular loss functions [47, 48]. The supervised learning task is usually subdivided into *regression* and *classification* tasks, each having its own goals and, therefore, requiring, its own loss functions.

For the *regression* task, the model is required to predict the continuous value corresponding to the input. That means that the desired outputs and hence model predictions are defined as real-valued variables $y_i^{\text{data}}, y_i^{\text{model}} \in \mathbb{R}$. The most popular loss functions for this case are mean squared error (MSE).

The per-example loss function $l_{\text{MSE}}(\mathbf{y}_i^{\text{data}}, \mathbf{y}_i^{\text{model}})$ of the mean-squared absolute error is defined as

$$l_{\text{MSE}}(\mathbf{y}_i^{\text{data}}, \mathbf{y}_i^{\text{model}}) = \left(\mathbf{y}_i^{\text{data}} - \mathbf{y}_i^{\text{model}}\right)^2 \tag{2.6}$$

In the *classification* task, the model calculates the probability of the entity represented by the input data used during training being a member of a particular class or group of classes. As a result, the desired and predicted output values are defined as vectors of probabilities of the object belonging to any of the range of classes, i.e. $y_{i,j}^{\text{data}}, y_{i,j}^{\text{model}} \in (0,1)$. Furthermore, if single-class classification is considered, the output should describe a discrete probability distribution, and therefore an additional condition is applied to answers that $\sum_{j=1}^{n} y_{i,j} = 1, \forall i$.

The most popular loss for classification tasks is *cross-entropy*, also known as *log-loss*, with the per-example loss function l_{CE} defined as

$$l_{\text{CE}}(\mathbf{y}_i^{\text{data}}, \mathbf{y}_i^{\text{model}}) = -\sum_{j=1}^{n} y_{i,j}^{\text{data}} \log\left(y_{i,j}^{\text{model}}\right) + \left(1 - y_{i,j}^{\text{data}}\right) \log\left(1 - y_{i,j}^{\text{model}}\right). \tag{2.7}$$

After having gained knowledge about certain loss functions employed in machine learning, it now is crucial to comprehend the learning process of discovering the value of θ that minimizes the loss function. The famous iterative optimization technique is called Gradient descent (GD) [49–51]. The GD method is based on utilizing the famous gradient property - that it points in the direction opposite to the steepest descent of the function. The method assumes that the optimized function is defined and differentiable in a neighborhood of the considered point. This condition is commonly satisfied while minimizing the loss, function as an integral aspect of neural network training.

In general, the gradient descent method proposes making small iterative steps in the direction opposite to the gradient in order to find the set of variables minimizing the studied function. For a scalar function $f(\mathbf{x})$ of a vector variable, $\mathbf{x} \in \mathbb{R}^n$ the method can be stated as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta \frac{\delta f(\mathbf{x}_t)}{\delta \mathbf{x}_t}, \tag{2.8}$$

where $\delta f(\mathbf{x}_t)/\delta \mathbf{x}_t$ is the gradient, $\eta \in \mathbb{R}$ is a learning rate - a hyper-parameter of a gradient descent method. We expect the function value to decrease after every algorithm iteration $f(\mathbf{x}_{t+1}) < f(\mathbf{x}_t)$, the behavior is referred to as the *convergence* of gradient descent. Eventually, the gradient descent is expected to stop at a function minimum, where the gradient is equal to zero. The selection of appropriate learning rate values significantly impacts the convergence of the algorithm - excessively large values of η impede the algorithm's convergence to the minimum, while excessively small values of η augment the number of iterations needed to approach the optimal function.

In the context of deep learning, first, the optimized variable is a set of artificial network parameters θ. Second, the gradient of a loss function over the ANN parameters $\delta\mathscr{L}/\delta\theta$ is calculated over all the objects of the training dataset $\mathscr{D}$. As a result, the gradient descent algorithm for artificial neural network training takes shape:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|\mathscr{D}|} \sum_{(\mathbf{x},\mathbf{y}^{\text{data}})\in\mathscr{D}} \frac{\delta l(\theta_t, \mathbf{x}, \mathbf{y}^{\text{data}})}{\delta\theta_t}, \tag{2.9}$$

where $|\mathscr{D}|$ is the cardinality of the training dataset, and $l(\theta_t, \mathbf{x}, \mathbf{y}^{\text{data}})$ is the pre-sample loss function introduced in the loss function definition Eq. (2.4).

A notorious flaw of the gradient descent optimization technique is that it converges to the *local minimum* : $\lim\limits_{t\to\infty} \mathbf{x}_t \to \mathbf{x}_{\text{LM}}$, i.e. the point where the function takes the minimum value only over some the neighborhood of point, i.e

$$\mathbf{x}_{\text{LM}} : f(\mathbf{x}_{\text{LM}}) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathscr{O}(\mathbf{x}_{\text{LM}}) \in \mathbb{R}^n, \tag{2.10}$$

instead of the desired *global minimum* point $\mathbf{x}_{\text{GM}}$, the smallest possible value reachable by the optimized function overall.

Furthermore, the gradient descent algorithm might even not reach the local minimum. The gradient value may reach zero and cause gradient descent to stop at a point where there is no local minimum, particularly in a *saddle point*, which is a critical point where the gradient is zero in one direction but slopes in another direction [49]. In light of the limitations of gradient descent, notable enhancements to the algorithm have been developed, with the Adam (Adaptive Moment Estimation) optimization algorithm being the most widely adopted in the industry and also in the present thesis. It has shown improved performance over other optimization algorithms in many deep learning tasks [52].

As noted previously, to execute the simple GD iteration Eq. (2.9) gradient has to be averaged over the whole, typically large, training dataset $\mathscr{D}$. This averaging requires a lot of numerical resources, thus, making the gradient descent algorithm computationally too expensive. A popular approach to reducing the complexity of a gradient descent algorithm is to use a stochastic approximation [53] of a gradient during an optimization algorithm iteration, Eq. (2.4). The resulting algorithm is known as a *stochastic gradient descent (SGD)* [49, 52, 54]. In more detail, for every iteration, SGD calculates the gradient using a small randomly selected sample of training data points $\mathscr{D}' \subset \mathscr{D}, |\mathscr{D}'| \ll |\mathscr{D}|$. This subset is referred to as a *mini-batch*. The resulting algorithm can be formulated as:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{|\mathscr{D}'_t|} \sum_{(\mathbf{x},\mathbf{y}^{\text{data}})\in\mathscr{D}'_t} \frac{\delta l(\theta_t, \mathbf{x}, \mathbf{y}^{\text{data}})}{\delta \theta_t}, \quad \mathscr{D}'_t \subset \mathscr{D} \tag{2.11}$$

Notably, since the gradient calculation is approximate in SGD, even the convergence of this algorithm to a local minimum of a loss function $\mathscr{L}$ is not guaranteed, compared to the vanilla gradient descent Eq. (2.9). Nonetheless, the stochastic gradient descent, typically, finds the set of ANN parameters θ resulting in a reasonably low value of a loss function [49].

Here, we specifically used the most popular stochastic gradient descent algorithm named Adam [55] (an acronym for adaptive moment estimation). The Adam algorithm is an extension of stochastic gradient descent that computes adaptive learning rates for each parameter by estimating the first and second moments of the gradients. The intuition behind the Adam optimization algorithm is that it can be visualized as a heavy ball with friction rolling down a slope, where the ball prefers to settle at a flat minimum in the error surface, as opposed to steep and narrow ones. This behavior is in contrast to traditional gradient descent, which behaves more like a light ball that bounces around the surface and may settle in a narrow and steep minimum [52, 56].

Mathematically speaking, first, Adam introduces the exponentially moving average of mean μ_t and variance v_t of gradient:

$$\begin{aligned} \mu_{t+1} &= \beta_1 \cdot \mu_t + (1-\beta_1) \cdot \nabla_t \\ v_{t+1} &= \beta_2 \cdot v_t + (1-\beta_1) \cdot (\nabla_t)^2 \\ \nabla_t &= \frac{1}{|\mathscr{D}'_t|} \sum_{(\mathbf{x},\mathbf{y}^{\text{data}})\in\mathscr{D}'_t} \frac{\delta l(\theta_t, \mathbf{x}, \mathbf{y}^{\text{data}})}{\delta \theta_t}, \quad \mathscr{D}'_t \subset \mathscr{D} \end{aligned} \tag{2.12}$$

where ∇_t is the stochastic estimation of gradient over mini-batch $\mathscr{D}'$ from Eq. (2.11), and $(\cdot)^2$ is point-wise squaring. After some bias corrections, μ_{t+1} and v_{t+1} are used to calculate an update of the artificial neural network parameters

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\mu_{t+1}}{\sqrt{v_{t+1}} + \varepsilon}, \tag{2.13}$$

where $\varepsilon \ll 1$ is a regularization term added for the numerical stability.

Finally, it is important to notice that the speed of gradient calculation is the bottleneck defining the overall execution speed of gradient descent optimization. In neural network training, this problem is addressed by backpropagation - a simple and cost-effective algorithm for gradient computation [49]. The backpropagation is based on the chain rule of calculating the derivatives of a function cascade. The chain rule states that the derivative of the composition of two differentiable functions $h(x) = f(g(x))$ can be calculated as:

$$\frac{\delta h}{\delta x} = \frac{\delta f}{\delta g} \cdot \frac{\delta g}{\delta x}. \tag{2.14}$$

If the considered function can be represented as the recursive repeat of the same function block $h = f_N(x_N, f_{N-1}(x_{N-1} \ldots f_1(x_1)))$, one can utilize the already calculated and stored gradients over the parameters of the later function iterations $\delta h/\delta x_i$, $\delta h/\delta f_{i-1}$ to calculate the gradients over the parameters of earlier iterations, e.g. $\delta h/\delta x_{i-1}$, $\delta h/\delta f_{i-2}$ via chain rule. In general, the backpropagation algorithm can be formulated for any differentiable function which can be represented as an unidirectional computational graph [49]. This generalized backpropagation is implemented in modern deep learning computational packages, e.g., in PyTorch [57]).

2.3.2 Feed-forward structures for the Neural Network Design

A multi-layer perceptron

The first and, perhaps, simplest and well-studied NN-based equalizer that we consider is the MLP, proposed for the short-haul coherent system equalization in [40] and the long-haul systems in [58]. The MLP is a deep feed-forward densely connected NN structure that handles the I/Q components for each polarization jointly, providing two outputs for each processed symbol: its real and imaginary parts. Due to the MLP's ability to process joint I/Q components, the equalizer can learn the nonlinear phase impairments in addition to the amplitude-related nonlinearities. When using the MLP, the channel, and device-induced memory effects are taken into account by incorporating the time-delayed versions of the input signal, as was carried out in [58].

The general equation, in a matrix form, describing the output vector y given the input x passing through the 3-layer MLP, is:

$$y = \phi\left\{\phi\left[\phi(x \times W_{n_1} + b_1) \times W_{n_2} + b_2\right] \times W_{n_3} + b_3\right\} \times W_{out}, \tag{2.15}$$

where x is the input vector with n_i elements, y is the output vector with n_o elements, ϕ is a nonlinear activation function, $W_{n_1} \in \mathbb{R}^{n_i \times n_1}$, $W_{n_2} \in \mathbb{R}^{n_1 \times n_2}$, $W_{n_3} \in \mathbb{R}^{n_2 \times n_3}$ and $W_{out} \in \mathbb{R}^{n_3 \times n_o}$ are the real weight matrices of the respective dimensions participating in each layer of the MLP, $b_{1,2,3}$ are the bias vectors, the indexes $n_{1,2,3}$ stand for the number of neurons in each hidden layer, and $\times$ in (2.15) is the matrix-vector convolution.

In a simulation environment, the MLP equalizer showed performance metrics similar to those delivered by the "traditional" digital back-propagation (DBP) with 2 steps-per-span and 2 samples-per-symbol at 1000 km of standard single-mode fiber [58]. In this thesis, we use the same 3-layer MLP as in [40].

Convolutional neural networks

Due to their feature extraction propensity [59], the CNNs have become one of the most commonly used NN structures in such areas as 2D image classification and 3D video applications [60, 61]. Convolution layers have also been found efficient in the analysis of temporal 1D sequences with several applications to time series sensors, audio signals, and natural language processing [62, 63]. For longer sequences, the CNN layer can be used as a *pre-processing step* due to its ability to reform the original sequence and extract its high-level features used for further processing cycles [64].

The convolutional layer is primarily characterized by three key parameters: the number of filters, the size of its kernel, and the layer activation function. The extracting functionality is achieved by applying n_f filters, sliding over the raw input sequence, and generating the number of output maps equal to n_f, with a fixed kernel size n_k. The convolutional layer is constructed as a squash function, which means that the input is mapped to a lower-dimensional representation, in which only the main (or desirable) characteristics are retained. Since the CNNs were mainly developed in the context of image recognition and spatial feature extraction, other parameters, such as padding, dilation, and stride are also used in the design of the convolutional layers. Considering that the input shape is $(B, M, 4)$, the output shape, after the CNN layer with all those parameters, is defined as (B, L_{out}, n_f), where the parameter L_{out} is the function of the CNN hyper-parameters and defined as:

$$L_{out} = \left\lceil \frac{M + 2 \cdot padding - dilation \cdot (n_k - 1) - 1}{stride} + 1 \right\rceil. \tag{2.16}$$

However, this thesis will not focus on the investigation of those additional parameters. Consequently, the default convolutional layer configuration is fixed with the padding equal to 0 (which corresponds to "valid" in Keras), the dilation equal to 1, and the stride equal to 1. Then, the input-output mapping of the convolution layer for this configuration can be described as follows:

$$y_i^f = \phi \left(\sum_{n=1}^{n_i} \sum_{j=1}^{n_k} x_{i+j-1,n}^{in} \odot k_{j,n}^f + b_{j,n}^f \right), \tag{2.17}$$

where y_i^f is the output feature map of the i-th input element produced by the filter f in the CNN layer, x^{in} is the input raw data vector, k_j^f is the j-th trained convolution kernel of the filter f, and b_j^f is the bias of the filter f. Further, n is the feature index of the kernel and input data, ranging from 1 to n_i, corresponding to the number of features in the data; ϕ, as before, denotes the nonlinear activation function used in the convolutional layer. Note that (2.17) is true for all $i \in [1, ..., L_{out}]$. Moreover, since the pooling layer captures only the most important features in the data and ignores the less important ones [65], the pooling discretization process is not used in our equalizers to avoid the downsampling of feature sequences.

2.3.3 Recurrent Structures for the Neural Network Design

Vanilla Recurrent NNs

Vanilla RNN is different from MLP and CNN in terms of its ability to handle memory, which is quite beneficial for time series analysis and prediction. Here we note that the feed-forward models (e.g., those, described above) can be reckoned, according to J. L. Elman [66], as an "... attempt to "parallelize time" by giving it a spatial representation. However, there are problems with this approach, and it is ultimately not a good solution. A better approach would be to represent time implicitly rather than explicitly." The recurrent structures described in the following subsections just do that implicit representation: RNNs take into account the current input and the output that the network has learned from the prior input. The propagation step for the vanilla RNN at time step t, can be described as follows:

$$h_t = \phi(Wx_t + Uh_{t-1} + b), \tag{2.18}$$

where ϕ is, again, the nonlinear activation function, $x_t \in \mathbb{R}^{n_i}$ is the n_i-dimensional input vector at time t, $h_t \in \mathbb{R}^{n_h}$ is a hidden layer vector of the current state with size n_h, $W \in \mathbb{R}^{n_h \times n_i}$ and $U \in \mathbb{R}^{n_h \times n_h}$ represent the trainable weight matrices, and b is the bias vector. For more explanations on the vanilla RNN operation, see, e.g., Ref. [67]. Even though the RNNs were tailored for efficient memory handling, it still suffers from the inability to capture long-term dependencies because of the vanishing gradient issue [68].

Long short-term memory (LSTM) NNs

LSTM is an advanced type of RNNs. Although RNNs suffer from short-term memory issues, the LSTM network has the ability to learn long-term dependencies between time steps (t), insofar as it was specifically designed to address the gradient issues encountered in RNNs

[69, 70]. The LSTM comprises a gateway architecture that includes three gate types: the input (i_t) gates, the forget (f_t) gates, and the output (o_t) gates, as shown in Fig. 2.2. The compact form of the forward-pass LSTM cell equations for a time-step t, (i.e., when we process the input feature sequence x_t having the size n_i) is [71, 72]:

$$\begin{aligned} i_t &= \sigma(W^i x_t + U^i h_{t-1}), \\ f_t &= \sigma(W^f x_t + U^f h_{t-1}), \\ o_t &= \sigma(W^o x_t + U^o h_{t-1}), \\ C_t &= f_t \odot C_{t-1} + i_t \odot \tanh(W^c x_t + U^c h_{t-1}), \\ h_t &= o_t \odot \tanh(C_t), \end{aligned} \tag{2.19}$$

where C_t is the cell state vector, h_t is the current hidden state vector of the cell with size n_h and h_{t-1} is the previous hidden state vector. Note that n_i is equal to the number of features, and n_h is the number of hidden units that will be chosen in the design process. The trainable parameters of the LSTM network are represented by the matrices $W \in \mathbb{R}^{n_h \times n_i}$ and $U \in \mathbb{R}^{n_h \times n_h}$ with the respective upper indices i, f, o, and c, referring to the particular LSTM gates mentioned previously. More details are given in Fig. 2.2. In (2.19), $\odot$ is the element-wise product, and σ denotes the logistic sigmoid activation function. The aim of the *input* i-gate is to store the content to the cell; the *forget* f-gate defines what information is to be erased; the *output* o-gate defines what information has to be passed to the next cell.

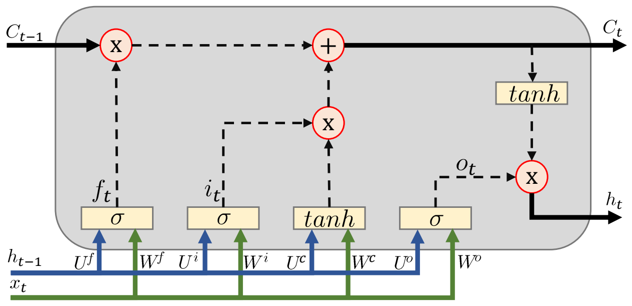


Fig. 2.2 The Long Short-Term Memory (LSTM) cell diagram represents graphically the operations described by (2.19) for one time-step. The arrows represent the "flow" of respective variables (the blue/green ones refer to the previous state and current input), the rectangles identify the nonlinear functions, and the symbols in circles identify the respective mathematical operations.

What makes the usage of the LSTM a dynamical approach is: the time sequence is processed by the array of LSTM cells ranging over the t-interval of interest, which is the memory size in our case. Besides the "dynamical" LSTM property, the *bidirectional* LSTM (biLSTM) provides a more robust solution for time series since with the bidirectional structure, we are learning the weights from the past visible values to the future hidden values, and that corresponds to our learning which features of the past values are useful for a particular symbol value prediction [73]. In the context of optical channel equalization, the key advantage of biLSTM is that it can efficiently handle intersymbol interference (ISI) between the preceding and following symbols.

In the context of channel equalization, the LSTM was suggested in [74, 75] to reduce the transmission impairments in IM/DD systems with pulse amplitude modulation (PAM). The LSTM-based approach was further developed in [4], where, for the first time, the biLSTM was used in an optical coherent system to compensate for fiber nonlinearities, but only in a simulation environment. Additionally, it was shown that the biLSTM also outperformed a low-complexity DBP [4]. More recently, a bi-vanilla RNN was applied as well for the soft-demapping nonlinear ISI [76].

Echo state networks (ESN)

The ESN is a promising type of RNNs due to its relaxed training complexity and its ability to preserve the temporal features from different signals over time [77–81]. The ESNs are in the reservoir computing (RC) category because, in the ESNs, only the output weights are trainable. In Fig. 2.3, the grey-colored area is the reservoir "main" structure containing the randomly connected "neurons" that capture the time features of the signal, while the output weights are trained to define which states are more relevant to describe the desired output. This thesis uses the concept of leaky-ESN [82] containing no output feedback connections. The motivation to choose the leaky-ESN architecture is that there was an experimental observation that the leaky-ESN configuration outperforms the traditional ESN in feature extraction for noisy time series [78]. The latter is, evidently, an important property in optical transmission-related tasks. The leaky-ESN is formalized for a certain time-step t, as follows:

$$a_t = \phi\left(W_r \times s_{t-1} + W_{\text{in}} \times x_t\right), \tag{2.20}$$

$$s_t = (1-\mu)s_{t-1} + \mu a_t, \tag{2.21}$$

$$y_t = W_o \times s_t, \tag{2.22}$$

where $s_t \in \mathbb{R}^{N_r}$ is the system state at time-step t, N_r is the number of hidden neurons units in the dynamic layers, which represents the dimensionality in the reservoir; $x_t \in \mathbb{R}^{n_i}$ and $y_t \in \mathbb{R}^{n_o}$ are the input and the output vector of the ESN, respectively; $W_r \in \mathbb{R}^{N_r \times N_r}$ is a reservoir weight matrix that defines which neuron units are connected (including the self-connections); this matrix is also characterized by a sparsity parameter s_p defining the ratio of connected neurons to the total possible connections number. Finally, $W_{in} \in \mathbb{R}^{N_r \times n_i}$ is the input weight matrix, μ is the leaking rate parameter, and $W_o \in \mathbb{R}^{n_o \times N_r}$ is the output weight matrix which is the only one that is trainable using a regression technique. This training phase in ESN does not affect the dynamics of the system, which makes it possible to operate with the same reservoir for different tasks [77]. A schematic representation of a leaky-ESN, including the sequential input, dynamic, static, and output layers, as depicted in Fig. 2.3.

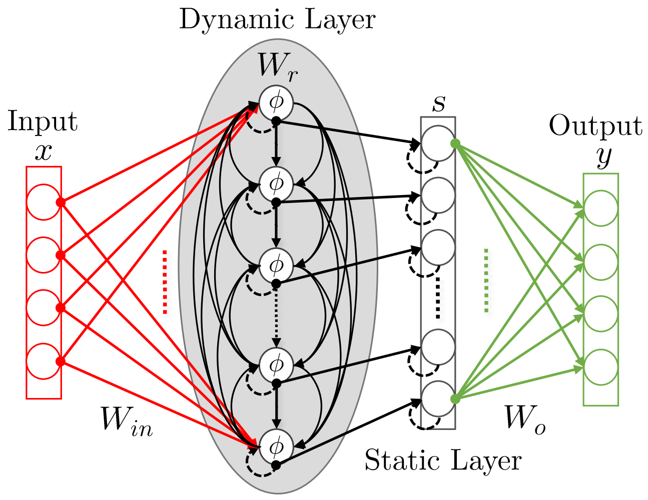


Fig. 2.3 Schematic of a leaky-ESN. A dynamical core called a reservoir is driven by an input signal x. The states of the reservoir s are combined non-linearly to produce the output y. The reservoir consists of N interconnected nodes followed by a static (leaking) layer. The circular dashed lines in the dynamic and static layers represent the past state values, while the solid lines represent the current time step value. The reservoir and input weights are fixed after initialization, while the output weights are learned to use a regression technique.

The signal passing through the dynamic layer in Fig. 2.3 is represented by (2.20), and this layer is the core of the reservoir structure. Then it is followed by a static layer, represented by (2.21), which incorporates the leaky-ESN behavior through accumulating (integrating) its inputs, but it is also losing exponentially (leaking) accumulated excitation over time. Finally, the output layer defines which units are relevant to the description of the current task (for the equalization, in our case), and it is described by (2.22).

Concerning the previous ESN applications for optical channel equalization [80], the ESN was implemented in the optical domain for the distortions' mitigation: a 2 dB gain in Q^2-factor was achieved for 64-QAM 30 GBd signals transmitted through 100 km fiber at 10 dBm input power. In addition, the same as it is in this thesis, the reservoir can be applied in the digital domain. In [79], the leaky-ESN was successfully applied after the analog-to-digital converter to enable 80 km transmission to reach below the KP4-FEC threshold [83] for a 32 GBd on-off keying signal.

2.4 Model-Driven Neural Network Equalizers Inspired by the Nonlinear Schrödinger Equation.

Now, we move to the designing category of an equalizer when it incorporates elements of some known equalization techniques and/or recasts that technique as a trainable/learnable approach. This category is known as "model-driven NN equalizers," and they leverage the knowledge of some element of a channel model or the NLSE.

2.4.1 Neural Network Design Based on Perturbation Theory

One popular approach is to use the perturbation-based NLC approach described in Sec 2.2.3, but now allowing the perturbation parameters to be trainable [1, 84–90].

The original idea came from the article [1], based on the definition of Eq.2.2, intra-channel cross-phase modulation (IXPM) and intra-channel four-wave mixing (IFWM) symbol triplets are considered NN inputs. Figure 2.4 shows the NN architecture proposed by [1]. This feed-forward NN is built from a triplet node input layer, two hidden layers consisting of 2 and 10 neurons, respectively, and two output nodes corresponding to the real and imaginary parts of the nonlinearity estimation. Notice that, before being fed into the NN model, the triplets are divided into real and imaginary parts. During training, a dropout layer with a probability of 0.5 is only added after the second hidden layer to prevent overfitting. In addition, the following activation functions were tested: SELU, RELU, Leak-RELU and Linear. The Leak Relu has finally been selected because the measured BER of the system has been reduced in Ref. [1]. To explain how the equalization process works, it is important to show how the loss function is used. Eq. 2.23 defines the loss function.

$$MSE = \frac{1}{B}\sum_{i=1}^{B} |X_i + \alpha\Delta Y_i - Y_i)|, \qquad (2.23)$$

where X_i is the transmitted symbol, Y_i is the received symbol, ΔY_i is the estimated impact of nonlinearity over the symbol X_i, and α is a normalization factor to account for changes in the launch power. The NN is used to estimate this nonlinearity by minimizing the distance between the transmitted symbol plus the estimated nonlinearity and the received symbol, as can be seen in the loss function.

This proposed method has two phases for the compensation process, unlike equalizers in the other subsections. First, after standard DSP algorithms, the NN is added to the receiver side in the training process to compute nonlinear disturbances. After the training, on the transmitter side, the NN model and all its parameters are applied to produce pre-distorted waveforms to prevent any disturbance to the receiver.

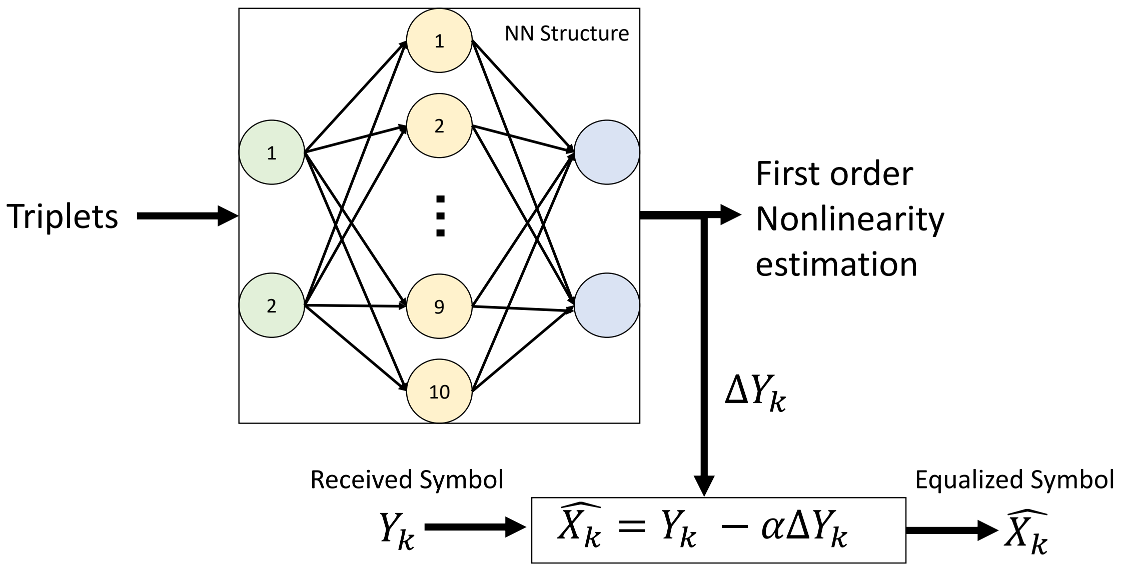


Fig. 2.4 This diagram describes the optimized NN architecture with two hidden layers used in Ref. [1] to estimate the first-order nonlinearity to equalize the received symbols by having the triplets as the input of the NN structure.

To evaluate the performance of this NN, an experiment over 2800 km standard single-mode fiber (SSMF) transmission using a 32 Gbd signal was operated. In this setup, after 5000 epochs of training, this NN algorithm achieved 0.35 dB Q-factor improvement compared to the single-step filtered DBP algorithm and 0.6 dB Q-factor improvement when compared with just linear equalization.

2.4.2 Neural Network design based on Digital back propagation

Another famous model-driven approach is named learned digital backpropagation (LDBP) [91–94]. In this group of solutions, the split-step Fourier method (SSFM) is built as a general structure of a deep neural network, so its operation resembles the one of the standard DBP, but this time with the flexibility to learn different linear and nonlinear steps.

A plain DNN can be considered as an alternation matrix multiplication $\hat{B}^{(k)}[\bar{x}] = \bar{x} * W^{(k)}$ with some particular weight vectors $W^{(k)}$ and point-wise applications of nonlinear functions,

$\hat{\rho}^{(k)}[\bar{x}] = \{\rho(x_j)\ \forall x_j \in \bar{x}\}$

$$\bar{y} = \mathrm{DNN}[\bar{x}] = \hat{\rho}^{(l)}[\hat{B}^{(l)}[\hat{\rho}^{(l-1)}[\hat{B}^{(l-1)}[\ldots\hat{\rho}^{(1)}[\hat{B}^{(1)}[\bar{x}]]], \tag{2.24}$$

where l is the number of layers. In general, weight vectors $b^{(k)}$ and nonlinear functions $\rho^{(k)}$ can be different ones for different layers $k \in [1,...,l]$.

Like in DNN, discrete SSFM can be considered as an alternation of the linear transformations $\mathscr{D}_\delta$ and memory-less (i.e., pointwise) nonlinear transformations $\mathscr{N}_\delta$

$$\bar{y} = \mathrm{SSFM}[\bar{x}] = \mathscr{D}_{\delta/2}[\mathscr{N}_\delta[\mathscr{D}_\delta[\ldots\mathscr{D}_\delta[\mathscr{N}_\delta[\mathscr{D}_{\delta/2}[\bar{x}]]] \tag{2.25}$$

Due to their similarities, we can use the numerical approaches created for optimizing DNNs on SSFM. Comparisons along these lines are shown in Fig 2.5. In particular, SGD algorithms can be employed to facilitate gradient-aware approaches to SSFM parameter optimization. This concept was used in Ref. [91], where linear distortion vectors $\mathbf{W}$ and nonlinear function coefficients ξ were optimized by SGD to reduce the mean error between the restored signal $\bar{x}_{\mathrm{res}} = \mathrm{SSFM}^{-1}[\bar{y}]$ and the restored signal $\bar{x}_{\mathrm{orig}}$.

The authors of [91] considered N_{sp} as a multi-span fiber-optic communication system consisting of the same $L_{\mathrm{sp}} \sim 100$km optical fiber spans. In order to compensate for the signal attenuation α, an optical amplifier is added after each span. The classical SSFM-based DBP [95] is applied to the signal received and sampled by the receiver to compensate for channel distortions. This strategy is schematically represented on the top branch in Figure 2.5.

For inverted amplifiers, multiplicative factors $G^{-1} = \exp(L_{\mathrm{sp}}\alpha/2)$ are considered. The approximately transmitted symbol vector $\hat{\mathbf{x}}_{\mathrm{DBP}} \in \mathbb{C}m$ is obtained by downsampling the SSFM result, followed by a rotational constant phase $\exp(i\phi_{\mathrm{rot}})$, $\phi_{\mathrm{rot}} = \arccos(\frac{\mathbf{x}\cdot\hat{\mathbf{x}}_{\mathrm{DBP}}}{||\mathbf{x}||||\hat{\mathbf{x}}_{\mathrm{DBP}}}||)$.

Authors of [91] used TensorFlow to implement the LDBP and optimize its parameters θ. For the optimization, they used the built-in Adam optimizer, which performed stochastic gradient descent with 30 input-output pairs $(\mathbf{y},\mathbf{x})$ (i.e., mini-batches) per optimization step.

The results obtained by applying LDBP to [91] show that LDBP with 3 StPS outperformed a much more complex DBP framework with 50 StPS. This new achievement was explained by the authors because LDBP was able to take account of the sections of the input signal spectrum filtered out during signal sampling on the receiver. Finally, LDBP gave a 50% decrease in complexity over DBP for comparable efficiency, taking the number of StPS as a rough measure of complexity.

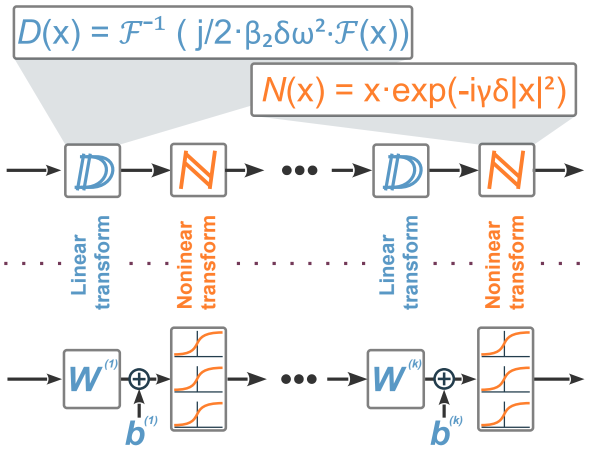

$D(x) = \mathcal{F}^{-1}(j/2 \cdot \beta_2 \delta \omega^2 \cdot \mathcal{F}(x))$
$N(x) = x \cdot \exp(-i\gamma\delta|x|^2)$
Linear transform
Noninear transform
Linear transform
Noninear transform
$W^{(1)}$
$b^{(1)}$
$W^{(k)}$
$b^{(k)}$

Fig. 2.5 Correspondence between linear dispersion D and non-linearity N operators, used within the SSFM to approximate the light evolution down the fiber, and the linear (vector-matrix convolution) and non-linear (activation) transformation steps in an NN structure.

In conclusion, it is important to note that currently, no general recommendation can be made regarding which of the two paths (adopting a black-box approach or a trainable version of an existing method) to pursue, as both directions have their advantages and disadvantages.

Chapter 3

The Performance Consideration for Designing Neural Networks-based Equalizers

3.1 Model-driven design using complex-value neural network

In this section, it is shown a new way to make a neural network with complex values for signal equalization on the receiver side. This approach is based on the assumption that Kerr nonlinearity is the leading distortion. The performance of the suggested neural network is investigated in several fiber systems and compared with the multi-layer neural network proposed in [3] and standard digital back-propagation. Also, it is presented a Bayesian optimizer implementation to fine-tune the parameters of the proposed architecture, i.e., its hyper-parameters, to every considered test case. In this study, the focus was on the metro networks, which it is considered both standard single-mode fiber and a large effective-area fiber base. The presented numerical and experimental results demonstrate that the proposed neural network leads to significant system performance improvements. Moreover, this approach was also shown to be able to mitigate not only nonlinear fiber transmission distortions but also impairments arising in the components of both the transmitter and receiver.

3.1.1 The Channel Model

The averaged evolution of the slowly varying complex-valued envelopes of the electric field in an optical fiber is described by the pass-averaged Manakov equation [96]:

$$\frac{\partial u_{H/V}(t,z)}{\partial z} + \frac{j\beta_2}{2}\frac{\partial^2 u_{H/V}(t,z)}{\partial t^2} = j\frac{8\tilde{\gamma}}{9}\left[|u_H(t,z)|^2 + |u_V(t,z)|^2\right] u_{H/V}(t,z), \quad (3.1)$$

where $u_{H/V}(t,z)$ are the normalized optical fields of horizontal (H) and vertical (V) polarization, respectively, β_2 is the group velocity dispersion, $\tilde{\gamma} = \gamma e^{-\alpha z}$ is the effective averaged nonlinearity coefficient, that includes the effective length scale $L_{eff} = (1 - e^{-\alpha L})/\alpha$ emerging due to averaging over a periodic loss and gain, γ is the fiber nonlinear coefficient, L is the span length and α is the fiber loss coefficient. In case $\beta_2 = 0$, one can show that the analytical solution of Eq. (3.1), at the end of the transmission link, can be expressed for H-polarization in terms of the transmitted x_{H_k} and received y_{H_k} soft symbols as [97]:

$$x_{H_k} = y_{H_k} e^{-8j/9\gamma L_{eff} N_s\left[|y_{H_k}|^2 + |y_{V_k}|^2\right]}, \quad (3.2)$$

where N_s is the number of spans.

Here, it is suggested the application of Eq. (3.2) to recover digitally the transmitted symbols x_k out of the received signal when the chromatic dispersion (CD) was already compensated for. For the Manakov equation (Eq. (3.1)), perturbation methods describing the field evolution in the highly nonlinear regime [98–101] have been developed under some limiting assumptions. Conversely, this thesis proposes adding trainable general nonlinearity functions to the memory-less solution provided by Eq. (3.2): additions Θ_1, Θ_2 to the exponential power, responsible for phase distortion, and additions Ξ_1, Ξ_2, responsible for amplitude distortion. The additional degrees of freedom introduced by these functions will let the equalizer learn the features of system behavior that are not covered by the original Eq. (3.2). Particularly, it is introduced the following model for the dual-polarization case:

$$x'_{H_k} = c_1 Y_{H_k} e^{\left[-c_2|Y_{H_k}|^2 - c_3|Y_{V_k}|^2 + \Theta_1\right]} + \Xi_1, \quad (3.3)$$

$$x'_{V_k} = c_4 Y_{V_k} e^{\left[-c_5|Y_{H_k}|^2 - c_6|Y_{V_k}|^2 + \Theta_2\right]} + \Xi_2, \quad (3.4)$$

where x'_{V_k/H_k} is the k-th recovered symbol in each polarization, $Y_{(V/H)_k}$ is the vector containing the sequence of received symbols $[y_{(V/H)_{k-N}}, \ldots, y_{(V/H)_k}, \ldots, y_{(V/H)_{k+N}}]$, $2N+1$, is the size of the memory in the model, c_k with $k = 1, \ldots, 6$ are complex vectors of size $2N+1$ each, and $\Theta_{1/2}(Y_{H_k}, Y_{V_k})$ and $\Xi_{1/2}(Y_{H_k}, Y_{V_k})$ are the two adaptive nonlinear functions mentioned above, which depend on the vectors Y_{H_k} and Y_{V_k}. It is important to stress that this model is used for the design of the NN, and not for solving the propagation equations.

The point-wise multiplication of the sequence of received symbols $Y_{(V/H)_k}$ and the weight coefficients c_k define the memory introduced into the nonlinear distortion Eq. (3.2) by chromatic dispersion (CD) and transceiver impairments. Furthermore, the nonlinear functions $\Theta_{1/2}$ and $\Xi_{1/2}$ reflect the residual nonlinear distortion introduced by the optical fiber and the transmitter components. In other words, the proposed NN is designed to compensate, at the receiver, for the deterministic intra-channel nonlinear distortion originating in both optical fiber and transceiver equipment.

The model hyper-parameters depend on the total fiber length and on the relationship between β_2 and γ. Consequently, during the numerical study, it is considered links based on two different types of optical fibers: large effective area fiber (LEAF) and standard single-mode fiber (SSMF). The considered link lengths were 6×80km (480 km) and 12×80km (960 km). The characteristics of the SSMF and LEAF are reported in Table 3.1.

Table 3.1 Considered fiber parameters.

Fiber	α [dB/km]	β_2 [ps/(nm.km)]	γ [1/W km]
SSMF	0.21	16.8	1.14
LEAF	0.225	4.2	1.3

For the purpose of describing the applicability of the proposed ML algorithm, it is crucial to demonstrate qualitatively where the strongly nonlinear regime begins in both fibers. The definition of the different regimes can be carried out by taking into account the relationship between chromatic dispersion (L_D) and Kerr nonlinearity (L_{NL}) effective lengths, described as [97]:

$$L_D = \frac{T_0^2}{|\beta_2|}, \quad L_{NL} = \frac{1}{\gamma P}, \tag{3.5}$$

where T_0 is the pulse length and P is the launch optical power. Fig. 3.1 shows the dependency of those two lengths on the launch power, considering a symbol rate equal to 32 GBd. In the “hot” region, the nonlinearity is the dominant signal distortion source whereas, when we are in the “cold” region, the linear distortions are dominant. As expected, LEAF has a stronger nonlinear dependence on the launch power than SSMF, resulting from its chromatic dispersion parameter, which is four times smaller than that of SSMF.

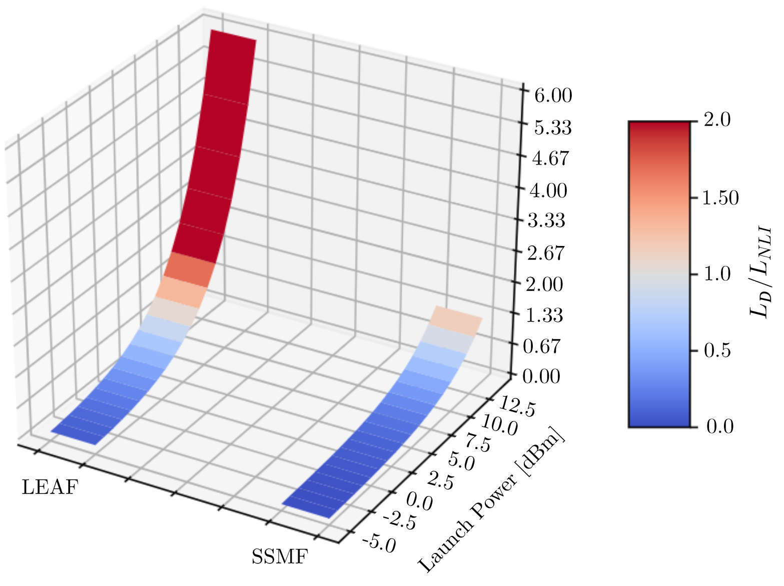


Fig. 3.1 Qualitative estimation of the relative strengths of chromatic dispersion and Kerr nonlinearity [Eq. (3.5)] for LEAF and SSMF at different power levels.

3.1.2 Neural Network Design Based on a Channel Model

Here, it is proposed to build the NN topology based on the loose signal evolution model Eqs. (3.3), (3.4). Contrary to the conventional real-valued NN architectures [49], in such a design, it was implemented using complex-valued weights, activation functions, and input symbols as suggested in [102, 103] to reflect the complex-valued laws describing the signal propagation. Figure 3.2 shows a schematic representation of the proposed NN topology.

The NN depicted in Fig. 3.2 predicts the k-th symbol transmitted in V polarization x_{V_k}. The proposed NN can obtain the transmitted symbols from the Horizontal (H) polarization x_{H_k} by swapping V, y_{V_k}, and H, y_{H_k}, polarization symbol sequences at the NN input. In this design, we used the classic mathematical representation of a neuron: the linear combination of inputs followed by a nonlinear function [49]. The following equations describe the architecture of the suggested NN: Each of the equations defines the exits of the neurons a), b), c), d), and e), as referred to in Fig. 3.2 (the corresponding exit labels are also marked in the figure with the sub-index "a", "b", etc.):

$$Neuron_{a,out} = -\left|\sum_{i=k-N}^{k+N} y_{V_i} a_i\right|^2, \tag{3.6}$$

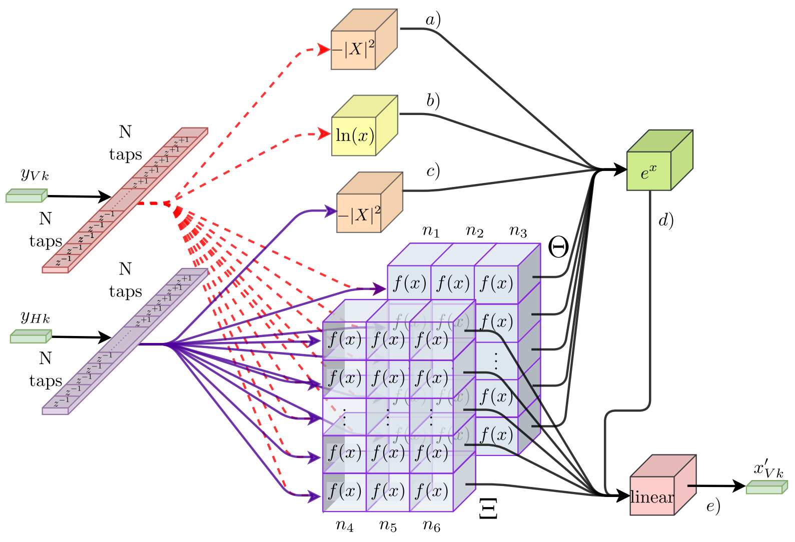
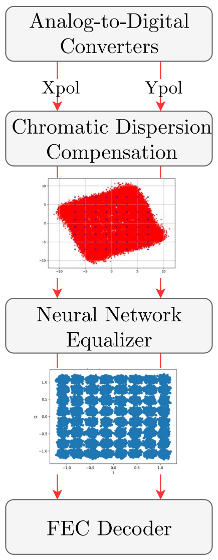


Fig. 3.2 The architecture of the proposed neural network.



Analog-to-Digital Converters
Xpol
Ypol
Chromatic Dispersion Compensation
Neural Network Equalizer
FEC Decoder

Fig. 3.3 Receiver setup in the simulator.

$$Neuron_{b,out} = \ln\left(\sum_{i=k-N}^{k+N} y_{V_i} b_i\right), \tag{3.7}$$

$$Neuron_{c,out} = -\left|\sum_{i=k-N}^{k+N} y_{H_i} c_i\right|^2, \tag{3.8}$$

$$Neuron_{d,out} = \exp\left(Neuron_{a,out} d_1 + Neuron_{b,out} + Neuron_{c,out} d_2 + \sum_{i=0}^{n_3-1} \Theta_i d_{i+3}\right), \tag{3.9}$$

$$Neuron_{e,out} = Neuron_{d,out} + \sum_{i=0}^{n_6-1} \Xi_i e_{i+1}. \tag{3.10}$$

Eqs. (3.6)–(3.10) can be combined in a single pipeline connecting NN output x'_{V_k} to its inputs y_{V_i}, y_{H_i}. This form is convenient to show the similarities between the NN architecture and

Eq. (3.4) (the similar expression can be written down for another polarization component):

$$x'_{V_k} = \left(\sum_{i=k-N}^{k+N} y_{V_i} b_i \right) \exp\left(-d_1 \left| \sum_{i=k-N}^{k+N} y_{V_i} a_i \right|^2 - d_2 \left| \sum_{i=k-N}^{k+N} y_{H_i} c_i \right|^2 + \sum_{i=0}^{n_3-1} \Theta_i d_{i+3} \right) + \sum_{i=0}^{n_6-1} \Xi_i e_{i+1}. \tag{3.11}$$

Here a_i, b_i, c_i, d_i, and e_i are complex tensor weights constituting the parameters of the NN. After a random initialization, these weights are learned from propagated data by means of stochastic gradient descent optimization performed by Adam optimizer [55]. The optimizer minimizes the mean-squared error (MSE) loss function L, describing the difference between the predicted symbols x'_i and the actually transmitted ("desired") ones x_i:

$$L = \frac{1}{M} \sum_{i=0}^{M-1} |x'_i - x_i|^2, \tag{3.12}$$

where M is the batch size.

We used custom activation functions $-|x|^2$, $\ln(x)$, and e^x, to reflect in the architecture our estimation of the distortion form, given by Eqs. (3.3) (3.4). Furthermore, Θ and Ξ, as mentioned before, are the analytically unknown nonlinear functions taking into account the deviation of the considered link from the ideal zero-dispersion case provided by Eq. (3.2); these functions are modeled via the NNs. Particularly, a three-layer perceptron was used for each function, given the fact that the multi-layer perceptron is a universal function approximator [49].

The NN that learns Θ has n_1 neurons in the first layer, n_2 neurons in the second, and n_3 neurons in the third one. In the same way, the NN that learns Ξ has n_4 neurons in the first layer, n_5 neurons in the second one, and, finally, n_6 neurons in the third one. For both NNs, all neurons have a complex-valued activation function $f(x)$:

$$f(x = x_r + jx_i) = \frac{e^{2x_r} - 1}{e^{2x_r} + 1} + j\frac{e^{2x_i} - 1}{e^{2x_i} + 1}. \tag{3.13}$$

3.1.3 Hyper-parameters optimization in NN-based Equalizers

Since the functions approximated by Θ and Ξ may have different nonlinearity types and memory sizes at different power levels depending on the leading propagation effect: the Kerr nonlinearity or the CD (Fig. 3.1), a Bayesian optimization algorithm [2] was implemented to derive the optimal values for the hyper-parameters of Θ and Ξ NNs for each studied communication system: the number of input taps N; neuron numbers n_i in each layer of each NN.

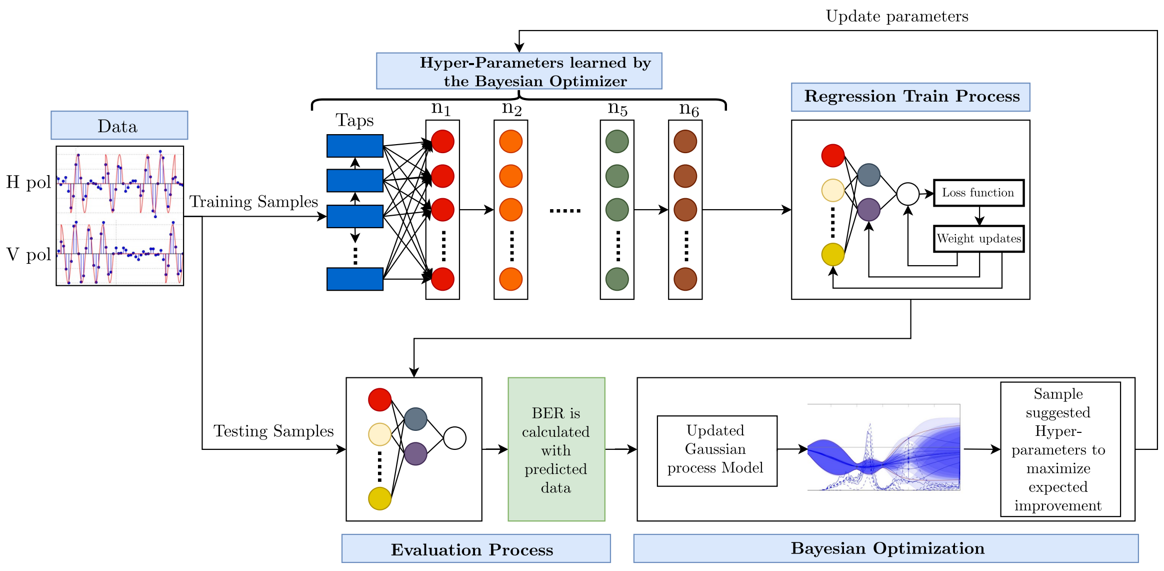

Update parameters
Hyper-Parameters learned by the Bayesian Optimizer
Regression Train Process
Data
Taps
n_1
n_2
n_5
n_6
H pol
V pol
Training Samples
Loss function
Weight updates
Testing Samples
BER is calculated with predicted data
Updated Gaussian process Model
Sample suggested Hyper-parameters to maximize expected improvement
Evaluation Process
Bayesian Optimization

Fig. 3.4 The principal scheme of the Bayesian optimizer [2] used to find hyper-parameters of NNs approximating nonlinear functions Θ and Ξ, Eqs. (3.3–3.4). A detailed explanation of the schematic is given in the main text.

Let us briefly specify the BO approach[1] that seeks the global optimum $\mathbf{x}^*$ of a black-box function opt, where $opt(\mathbf{x})$ can be evaluated for any arbitrary $x \in \mathscr{X}$. That is, $x* = arg\min_{x\in\mathscr{X}} opt(x)$, where $\mathscr{X}$ is a hyperparameter space that can contain categorical, discrete, and continuous variables [104]. For solving the problem formulated above, the BO assumes that the function opt was sampled from a Gaussian process. The BO maintains a posterior distribution for this function when observations are made [105]. For this application, the observations are the outcomes of performing the NN-based equalization trials with different hyperparameters. Fig. 3.4 illustrates the complete hyper-parameters optimization process. During the optimization, it is considered a signal optical launch power 3 dB higher than the CDC optimal level for every test-case in order to have a stronger response of the system performance to the hyper-parameters values. The learning rate and the batch size are fixed at 0.001 and 1000, respectively. In the beginning, the number of input taps N and the numbers of neurons in all layers n_1–n_6 are initialized as 20 and 100, respectively. The optimization cycle starts with training the NN via backpropagation with a fixed set of hyper-parameters for 5000 epochs, i.e., full passes through the whole dataset, over the training dataset containing

[1]The hyperparameter optimization can be done using methods other than the BO, although, as mentioned in Ref. [104, 105], the BO offers numerous advantages over search algorithms in terms of finding good candidates with fewer interactions.

2^{18} restored symbols. After each training epoch, we calculate the BER obtained by the whole NN (Fig. 3.3) on the independently generated testing dataset containing 2^{16} symbols. The best BER obtained during the training is recorded. After training, the best BER is fed as the optimization target to the Bayesian optimizer [2]. The optimizer assumes that the conditional distribution of BERs – given the particular values of hyper-parameters – is a Gaussian process. Having received the new measured BER value, the optimizer updates the process model and generates a new set of hyper-parameters to be tested. Having carried out 20 Bayesian optimizer cycles, we selected the set of hyper-parameters that produced the lowest BER.

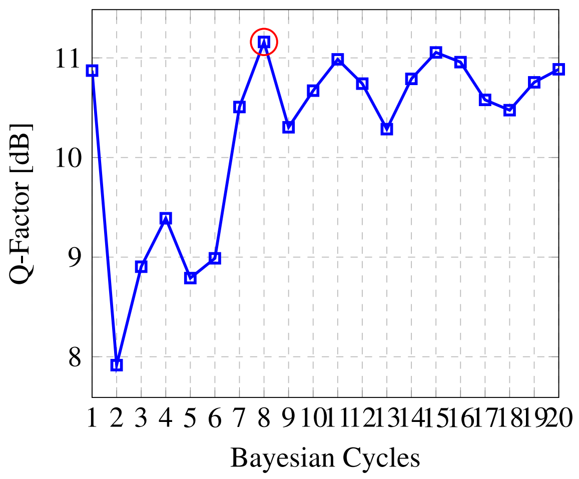


Fig. 3.5 Example of the iterative optimization for the Bayesian method in the case of DP-64 QAM single channel LEAF (6×80km) at 3 dBm.

Fig. 3.5 shows an example of the benefits provided by the Bayesian optimizer method. This figure displays the optimization for the case of transmission of a DP-64QAM single channel over LEAF (6×80km) at a launch power of 3 dBm (that will be discussed in detail in the next section). As can be seen, this method reached the best set of hyper-parameters – and thus the best Q-factor – on the 8th optimization cycle. The best configurations obtained for each considered scenario, in this study, are shown in Table 3.2.

Noticeably, the optimizer suggested several sets with similar performance levels. Among these, one could select the set with the best balance between performance and numerical complexity. However, this study maximized optical performance and, hence, always chose the best-performing point. Seeking the optimal balance between optical performance gain and cost is beyond the scope of this work. Besides fine-tuning the Bayesian optimization, other advanced ML techniques such as pruning [106] and efficient scaling [107] can be used to achieve lower computational complexity levels without reducing the NN performance.

Table 3.2 Hyper-parameters of neural networks approximating Θ and Ξ functions from Eq. 3.11 learned by means of Bayesian optimization.

Scenario	**N (taps)**	n_1	n_2	n_3
		n_4	n_5	n_6
LEAF (Numerical – 6×80km)	11	54	46	40
		4	38	50
LEAF (Numerical – 12×80km)	18	62	34	56
		55	18	40
SSMF (Numerical – 6×80km)	20	51	54	53
		19	22	56
SSMF (Numerical – 12×80km)	26	54	47	50
		41	38	61
SSMF (Trial data – 8×76+4km)	20	90	53	88
		56	60	64

To eliminate some possible dataset periodicity, which could cause overfitting and overestimation [108], the training dataset objects were randomly shuffled at the beginning of every epoch. The numerically generated training and testing datasets were measured to achieve a normalized cross-correlation below 0.6%, to ensure their independence. The code developed to obtain the results depicted in this study is provided online on the platforms Github and Zenodo [109].

3.1.4 Simulation and experimental setups

The system setup considered in the numerical study is illustrated in Fig. 3.6. First, a random bit stream is generated for each polarization. Afterward, gray-coded 32 GBd 64QAM symbols are mapped from the bit sequences. The resulting signals are upsampled with a sampling rate equal to four times the symbol rate and shaped using an RRC with roll-off = 0.06. Finally, a polarization beam combiner (PBC) is used to combine both polarizations into a single-channel dual-polarized signal, which is then fed into the optical fiber.

To simulate the (forward) propagation of the signal in the fiber, it was used a symmetrized split-step Fourier method [97] solving the ordinary Manakov equation [24]. It was considered 6×80 km and 12×80 km systems consisting of LEAF or SSMF. The considered fiber parameters are given in Table 3.1. At the end of each span, the fiber losses are fully compensated by a lumped Erbium-doped fiber amplifier (EDFA). Additive White Gaussian

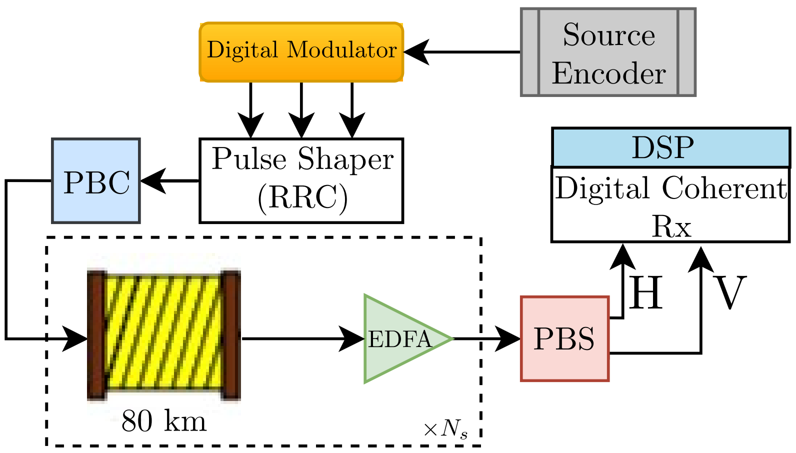


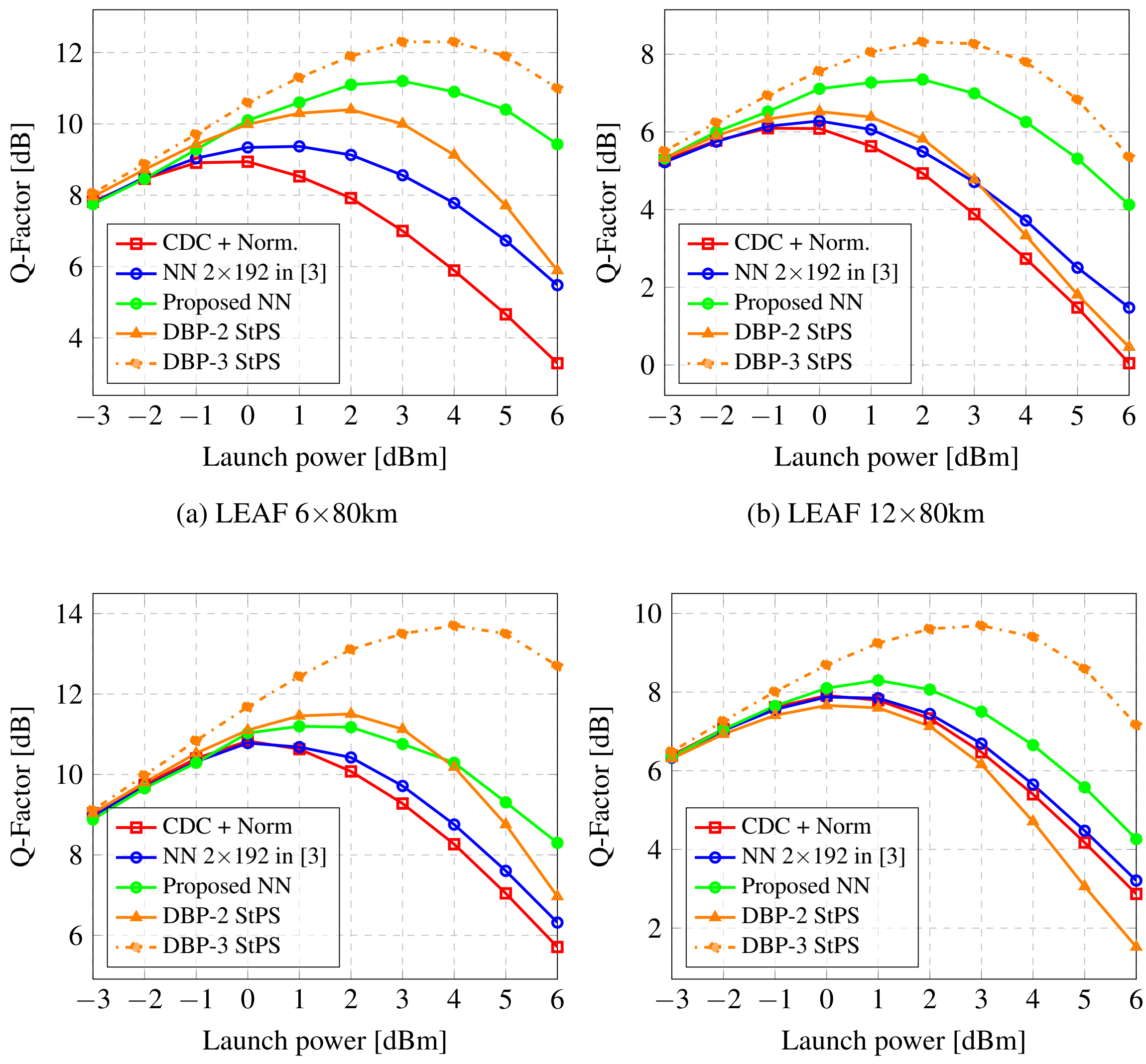
Fig. 3.6 The system setup of the transmission loop considered in our numerical simulations (Single Channel Transmission).

Noise (AWGN) representing the amplified spontaneous emission (ASE) noise is also added. The EDFA noise figure is $NF = 4.5$ dB. At the receiver side, both H and V polarizations are first separated using a polarization beam splitter (PBS) and then fed to a digital coherent receiver, where the signal passes through a matched filter and is downsampled to the symbol rate. The resulting signal is processed offline using the DSP outlined in Fig. 3.3 to recover the transmitted symbols. Since there were no transceiver impairments in numerical simulations, the receiver DSP consisted of: an ideal CDC compensator followed by a single-tap adaptive filter performing amplitude and phase normalization of the received symbol stream to the desired one (CDC + Norm), and, finally, the tested nonlinearity mitigation algorithm.

In the experimental study, the data were obtained by using the setup of the field trial described in [110] and [111]. The transmission link consisted of 8$\times$76 km G.652 SSMF spans along with two 2 km long SSMF connectors deployed between Torino and Chivasso in Italy, leading to a total length of 612 km. The transmitted spectrum consisted of 15$\times$200G neighbour channels (DP-16QAM) on the right- and left-hand sides of the channel under test, resulting in a 31-channel WDM transmission system in the 37.5 GHz grid (33.01 GBd). On the receiver side, the WDM signal was first amplified and then converted into the electrical domain using a coherent front end. An ADC with 18 GHz bandwidth operating at 80 GSample/s was used to capture sets of 5×10^5 samples per tributary. Digital signal processing is then applied at the receiver. Firstly, bulk accumulated chromatic dispersion was compensated using a frequency domain equalizer, followed by the removal of carrier frequency offset. A constant-amplitude-zero-autocorrelation-based training sequence was then located in the received frames, and the equalizer transfer function was estimated from it.

After equalization, the two polarizations were de-multiplexed, and time corrected. Carrier phase estimation was then achieved with the aid of pilot symbols. Finally, DSP processed two independent downsampled measurements and used them as training and testing data for offline testing of the nonlinearity mitigation algorithms.

3.1.5 Results and discussions



(a) LEAF 6×80km

(b) LEAF 12×80km

(c) SSMF 6×80km

(d) SSMF 12×80km

Fig. 3.7 Signal equalization performance of the proposed NN, benchmarked against digital backpropagation (DBP) and the reference neural network (2×192) [3] for various numerically studied fiber-optic links. In all testcases DP-64QAM 32 GBd single-channel optical signal was considered.

The proposed NN was benchmarked against the classic digital backpropagation (DBP) [95] with 2 samples/symbol and 2 and 3 steps per span (StPS), along with the NN-based equalizer proposed in [3]. This reference NN was implemented exactly as stated in [3]: it had two layers with 192 neurons and processed the same number of input taps as the proposed NN. The sequences of symbols from both polarizations were simultaneously fed as input to the reference NN from Ref. [3]. In this case, however, real and imaginary parts are fed as different parts of the input vector [40], since this is a real-value neural network. All the algorithms used in this study are static. As the performance metric, we used the Q-factor expressed through BER as follows:

$$Q = 20\log_{10}\left[\sqrt{2}\,\mathrm{erfc}^{-1}(2BER)\right], \tag{3.14}$$

where erfc^{-1} is the inverse complementary error function. All the main results of this subsection are summarized in Table 3.3.

Table 3.3 Summary of all results obtained by CDC + Norm, NN 2×192 neurons [3], DBP and the proposed NN. The optimal launch power, in dBm, and the peak Q-factor, in dB, are indicated for all considered testcases.

Fiber	Q / Best P [CDC + Norm]	Q / Best P [NN in [3]]	Q / Best P [Proposed]	Q / Best P [DBP-2 StPS]	Q / Best P [DBP-3 StPS]
LEAF 6×80km	8.94 dB / 0 dBm	9.37 dB / 1 dBm	11.16 dB / 3 dBm	10.4 dB / 2 dBm	12.3 dB / 4 dBm
LEAF 12×80km	6.09 dB / 0 dBm	6.28 dB / 0 dBm	7.34 dB / 2 dBm	6.52 dB / 0 dBm	8.31 dB / 3 dBm
SSMF 6×80km	10.83 dB / 0 dBm	10.81 dB / 0 dBm	11.19 dB / 2 dBm	11.50 dB / 2 dBm	13.7 dB/ 4 dBm
SSMF 12×80km	7.91 dB / 0 dBm	7.98 dB / 0 dBm	8.29 dB / 1 dBm	7.66 dB/ 0 dBm	9.68 dB / 3 dBm
SSMF (Trial data)	6.81 dB / 3 dBm	7.58 dB / 3 dBm	8.85 dB / 4 dBm	-	-

Fig. 3.7 shows the comparison of Q-factors obtained by employing the considered algorithms for nonlinearity mitigation in the numerically studied testcases. For LEAF, Fig. 3.7a illustrates the 6×80 km case and Fig. 3.7b the 12×80 km case. In the first case, the proposed NN improved the Q-factor by up to ∼2.1 dB when compared with CDC + Norm and up to ∼1.7 dB when compared with the reference NN from Ref. [3], at their respective optimal optical launch powers. The proposed NN also outperforms the DBP with 2 StPS performance by 0.7 dB. Evidently, it is expected that the DBP with higher complexity (with more steps or sampling points) outperforms the proposed NN. Indeed, the DBP with 3 StPS offers better performance. For the second scenario, a similar improvement trend was observed, see Fig. 3.7b. At the optimal launch powers, the proposed NN increased the Q-factor by up to ∼1.25 dB when compared to the CDC + Norm and by ∼1 dB when compared with the reference NN from [3]. Also, the proposed NN outperforms the DBP 2 StPS performance by 0.8 dB.

For the SSMF system, Fig. 3.7c shows the result for a 6×80 km case, where it was observed an improvement of the Q-factor of up to $\sim$0.4 dB when using the proposed NN with respect to both the CDC + Norm and the reference NN in [3]. In the 12×80 km case, (Fig. 3.7d) the proposed NN led to a Q-factor improvement of up to $\sim$0.4 dB with respect to both the CDC + Norm and the NN from [3]. In this instance, a single-tap adaptive filter performing amplitude and phase normalization was able to perform better than the DBP with 2 StPS. As predicted, using a fiber with a 4 times higher chromatic dispersion (β_2) impacts the performance of the proposed NN, since the proposed NN shows better performance when the nonlinearity produces the dominant contribution, as is expected due to the chosen structure.

In all the numerically considered test cases, the optimal launch power for the proposed NN was higher than for CDC + Norm and the reference NN in [3]. This shows that the performance gains of the proposed NN are obtained by improving nonlinearity mitigation.

Contrary to the numerical studies, in the experiment, it was not possible to apply the DBP technique because the carrier had no appropriate information about the system parameters installed: they were laid out in the field and were poorly accessible. So only the downsampled data was available for the tests. Therefore, we will compare the proposed NN with the traditional CDC + norm and the reference NN from [3].

Fig. 3.8 shows the experimental results obtained in the field trial and processed by the CDC, the NN from [3], and, eventually, the proposed NN. First, the optimal launch power obtained using the proposed NN equalizer increased from 3 to 4 dBm, and the Q-factor improved by up to $\sim$2 dB and $\sim$1.2 dB when compared, at the best power level, with the CDC+Norm and the reference NN from [3], respectively. Moreover, we have highlighted the different gains in the linear and nonlinear regions, which indicates that the proposed NN is compensating for nonlinear effects since the performance improvement increases with the launch power. At the lowest measured launch power (0 dBm), it was observed a gain of 1.7 dB when compared with the traditional CDC + norm. This gain in the linear regime comes from the ability of the neural network, by using the specially designed nonlinear parts $\Xi_{1/2}$, to mitigate the impact of additional limitations such as, e.g., the impact of low-resolution digital-to-analog and analog-to-digital converters, the driver amplifier, and the dual polarization Mach-Zehnder modulator, that was not entirely tackled by the current DSP. The imperfections of the transmitter – e.g., the $S_{21}(f)$ – and of the DAC low-resolution – shown as ENOB versus frequency – are reported in Fig. 3 of [112]. On top of these linear regime gains, we found that, at the optimum power levels, (3 dBm for CDC+Norm and 4 dBm for the proposed algorithm), the proposed NN provides an additional 0.34 dB of Q-factor gain, with the total gain being 2.04 dB. For the proposed NN, the optimum launch power increases in the experiment by $\sim$1 dB, which is in agreement with the numerical analysis. Notably, the reference NN from [3] did not improve the optical performance when increasing the optimal launch power.

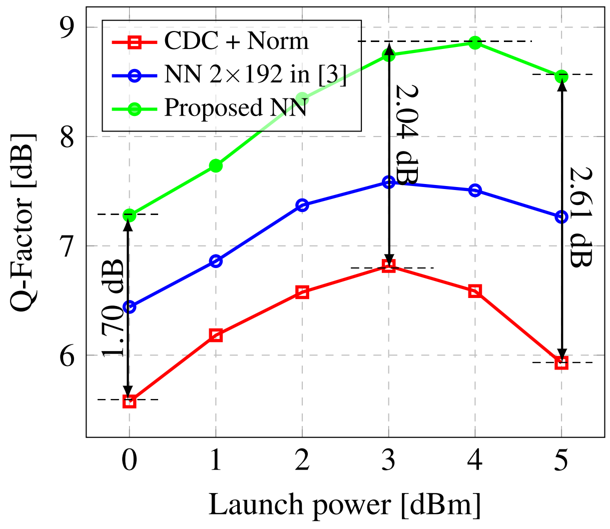


Fig. 3.8 Comparison of the equalization performance obtained by the considered algorithms in the field trial of 612 km SSMF legacy link (31 Channel WDM transmission).

3.2 Data driven investigation in realistic optical transmission setups

This section examines the performance of the previously proposed NN equalizers, the MLP, and biLSTM, using experimental data. Here, it is introduced a novel scheme of the equalizer based on the convolutional-recurrent NN (CRNN) technique that combines the properties of convolutional and recurrent layers. The performance of the proposed equalizer is tested and compared to other NN architectures and DBP [5] in an experimental setup. We analyze the single-channel (SC) and wavelength-division multiplexing (WDM) transmission of a dual-polarization (DP) 16 QAM signal with a 34.4 GBd rate along 9×50km TrueWave Classic (TWC) fiber spans. First, the CRNN is evaluated in an SC case, providing about 1 dB improvement in Q-factor when compared to a traditional DSP [113]. Next, it is investigated the performance of the new equalizer in the WDM scenario with 96 channels. In this experiment, we specifically aim to identify whether the biLSTM is capable of compensating for any of the cross-phase modulation (XPM) distortions, as it was observed numerically in [4]. Our results confirm that the NN equalizer architectures that involve a biLSTM layer are able to partially compensate for the XPM-induced distortions. In the WDM case, the proposed CRNN equalizer increases the maximum Q-factor by up to 1.04 dB, with the optimal launch power increasing by 2 dB. In both scenarios, the CRNN equalizer outperforms the results delivered by the MLP and biLSTM, also showing better performance than the traditional DBP with 3 StPS.

3.2.1 The new convolutional-recurrent neural network equalizer

The combination of convolutional and recurrent NNs has proven to work efficiently for speech enhancement [114] and image classification [115]. In this section, the functionality of this advanced NN combination for impairment mitigation in optical transmission systems is analyzed.

First, it is important to describe the input layer of the NN equalizers studied in this section. For the CRNN and biLSTM equalizers, the input shape is a 3D tensor with dimensions $(B, M, 4)$, where B is the mini-batch size, M is the memory size defined as $M = 2N + 1$, N being the number of neighboring (past and future) symbols used for the equalization; 4 is the number of features referring to the real and imaginary parts of two polarization components. For the MLP, the input layer must have a 2D tensor shape: $(B, 4M)$, but the input symbols are the same as for the CRNN and biLSTM.

Here we consider three NN topologies: i) two densely-connected layers (i.e., the MLP) equivalent to that used in [3]; ii) a single biLSTM layer as in [4]; and iii) the new CRNN equalizer, with a 1D convolutional layer (1D-Conv) defined by X filters and kernel size Z, followed by a biLSTM layer defined by Y hidden units (cells). In the case of CRNN and biLSTM, a flattening layer is used to reduce the output dimensionality. Finally, the output layer with two linear neurons is used in the NN's output to represent the real and imaginary parts of the recovered symbol in one of the polarizations. The proposed CRNN equalizer structure is illustrated in Fig. 3.9.

Unlike previous approaches, this work implemented a Bayesian optimizer (BO) step, following the procedure described in [116], to define the values of N, X, Y, Z, batch size, and the activation function type considered in the CRNN equalizer. After carrying out 20 BO cycles, the set of hyper-parameters corresponding to the lowest BER reached is defined. Each cycle is a full simulation that starts with training the NN with a fixed set of hyperparameters via back-propagation over the training dataset using the traditional MSE loss function and the Adam optimizer. This training dataset contains 2^{20} symbols, and the training is carried out for 500 epochs with a learning rate of 0.001. After every training epoch, we calculate the BER obtained by the whole NN on an independently generated testing dataset containing 2^{17} symbols and record the lowest BER found. The hyper-parameters found by the BO were: $N = 20$ taps, $X = 244$ filters, the kernel size $Z = 3$, $Y = 226$ hidden units in the

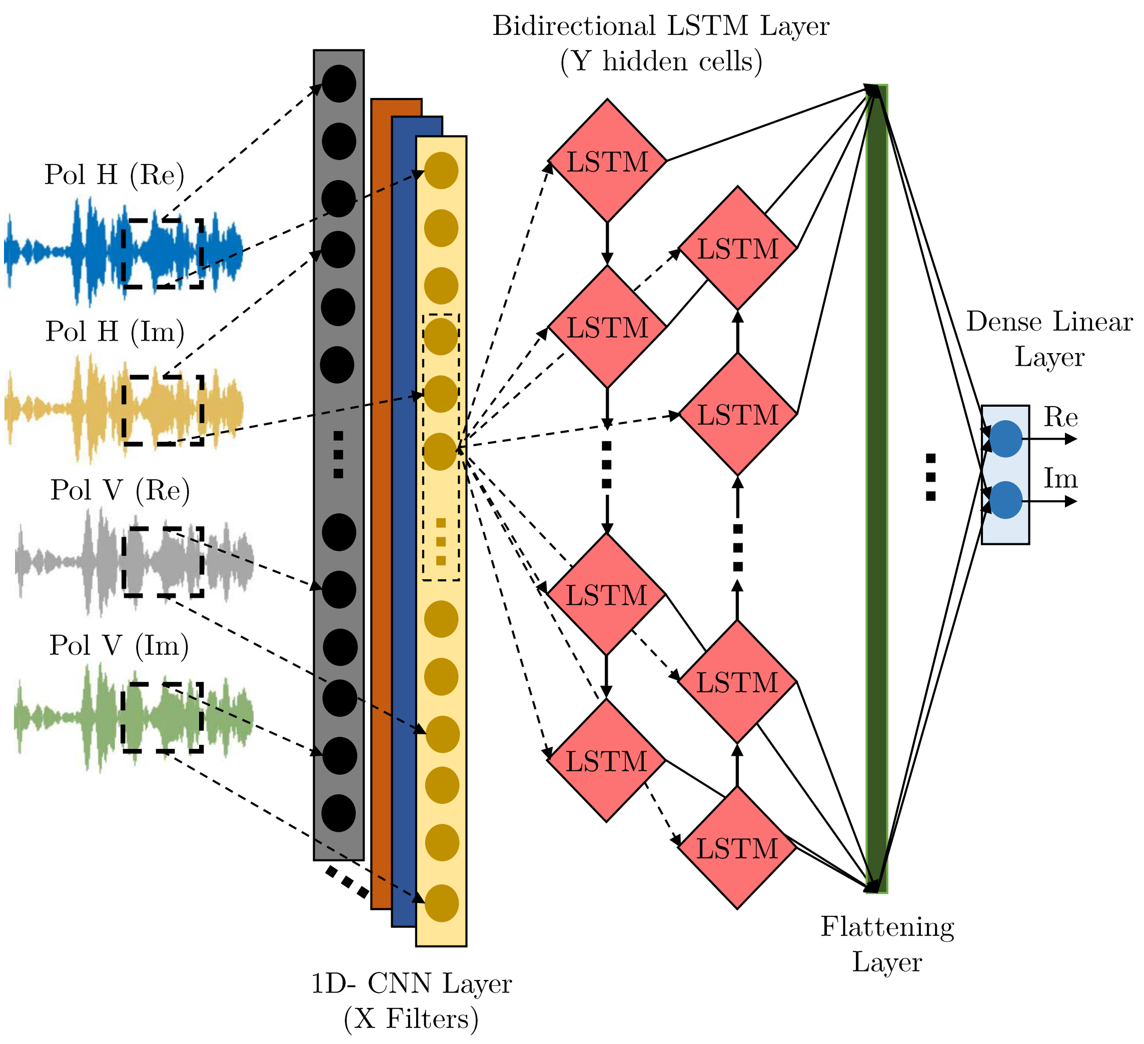


Fig. 3.9 Scheme of the proposed CRNN equalizer.

LSTM layer, and the mini-batch size $B = 4331$. The activation functions found by the BO for the 1D-Conv and LSTM cells were Leaky ReLU with $\alpha = 0.2$ and Tanh, respectively. The BO optimization was carried out using the data corresponding to the highest launch power available. For a fair comparison, it was also used the same number of hidden units ($Y = 226$) found by the BO to the CRNN in the pure biLSTM equalizer, and all equalizers used the same number N of taps. For the MLP, it was set 192 neurons in the hidden layers, since no improvement has been observed when increasing this number further. After having determined the optimal architecture, each NN equalizer was trained for each individual launch power using 2^{20} symbols over 200 epochs, employing the traditional MSE loss function and the Adam optimizer with a learning rate of 0.001. The validation set and evaluation of the resulting BER were carried out using 2^{17} independent symbols. Finally, it is important to

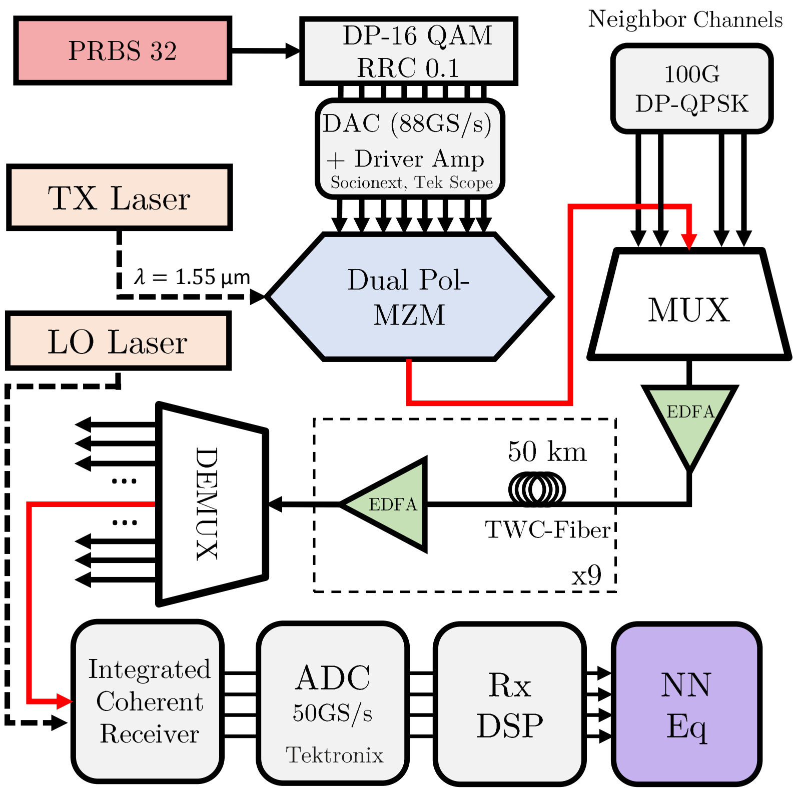


Fig. 3.10 Experimental setup used for data generation.

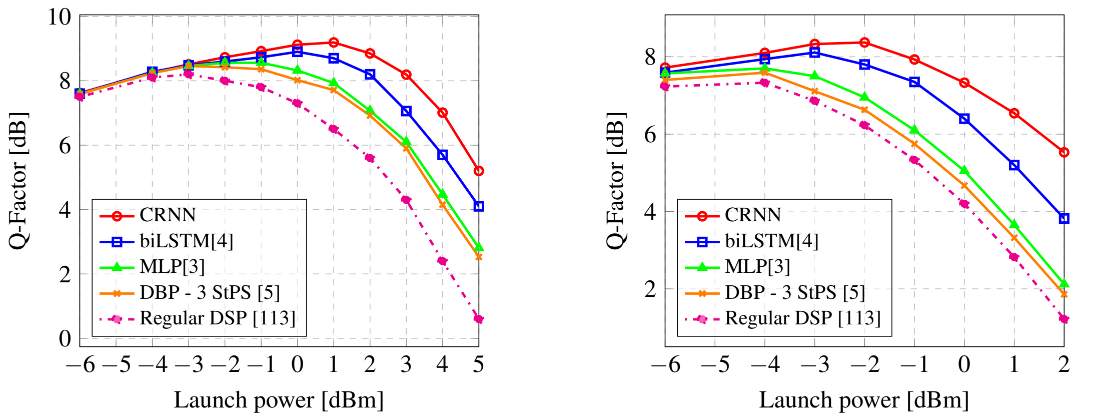
highlight that the NN training data were shuffled using the numpy.random.shuffle function in Python before feeding them into the NN: such a shuffling eliminates any possible data periodicity. The independent datasets were created using a pseudo-random binary sequence (PRBS) of order 32 to avoid overestimation of the Q-factor gain [108].

3.2.2 Simulation and experimental setups

Fig. 3.10 shows the setup used in the experiment. At the transmitter, the DP-16QAM 34.4 GBd symbol sequence was mapped out of data bits generated by an $2^{32}-1$ order PRBS. A digital RRC filter with a roll-off of 0.1 was applied to limit the channel bandwidth to 37.5 GHz. The filtered digital samples were uploaded to a digital-to-analog converter (DAC) operating at 88 Gsamples/s. The outputs of the DAC were amplified by a four-channel electrical amplifier, which drove a dual-polarization in-phase/quadrature Mach–Zehnder modulator, modulating the continuous waveform carrier produced by an external cavity laser at $\lambda = 1.55\mu m$. The resulting optical signal was then transmitted along 9×50 km spans of TWC optical fiber with EDFA amplification only, together with up to 95 neighbouring channels (100G QPSK, 50

GHz ITU grid, for the WDM scenario). The parameters of the TWC fiber span at $\lambda = 1.55\mu m$ are: attenuation coefficient $\alpha = 0.23$ dB/km, dispersion coefficient $D = 2.8$ ps/(nm·km), and effective nonlinear coefficient $\gamma = 2.5$ (W·km)$^{-1}$. On the RX side, the optical signal was converted into the electrical domain using an integrated coherent receiver. The resulting signal was sampled at 50 Gsamples/s by a digital sampling oscilloscope and processed by an offline DSP based on [113] which includes chromatic dispersion compensation, MIMO equalization, clock recovery, and pilot-aided carrier recovery. The system's performance is evaluated in terms of the Q-factor.

3.2.3 Results and discussions



(a) Single Channel - TWC Fiber (450 km) (b) 96 Channels WDM - TWC Fiber (450 km)

Fig. 3.11 Performance of the proposed CRNN, benchmarked against biLSTM equalizer [4], two layers MLP equalizer [3] and DBP 3 StPS [5] for two experimental fiber-optic scenarios.

Fig. 3.11a and Fig. 3.11b display the results obtained by the proposed CRNN for the SC and WDM systems, respectively, and their comparison with the results for MLP and biLSTM equalizers. For the SC, the proposed CRNN improved the Q-factor by 1 dB when compared to using the linear DSP only, by 0.29 dB when compared with the biLSTM equalizer, by 0.63 dB when compared to the results of the MLP equalizer, and by 0.73 dB when compared to the DBP with 3 StPS. Additionally, for the CRNN, the optimum launch power improved from -3 dBm to 1 dBm, which highlights the new method's potential to mitigate nonlinear effects. This result may be a consequence of using the convolutional layer before the recurrent layers. By doing so, it is possible to extract middle-level locally invariant features from the input series, thus creating a pre-enhancement step for the signal fed into the biLSTM layer. The parameters of the DBP were also optimized to produce the best BER. Importantly, we notice that the result in Fig. 3.11a agrees with the conclusions of previous simulations:

the MLP performs slightly better than DBP [3], and the biLSTM Q-factor improvement is noticeably higher than the DBP one [4].

A 96-channel WDM system was also considered, evaluating the performance of the considered NN equalizers in the presence of inter-channel crosstalk. In this case, the CRNN improved the Q-factor by 1 dB when compared to the regular DSP only, by 0.26 dB when compared to the biLSTM equalizer, by 0.67 dB when compared to the MLP equalizer, and by 0.78 dB when compared to the 3 StPS DBP. The optimum launch power also increased by 2 dB when using the proposed CRNN: from -4 dBm to -2 dBm. Moreover, as can be seen, the topologies that use the biLSTM layers were able to partially recover the XPM impact insofar as some higher gain was observed compared to the results of the SC scenario, Fig. 3.11a. As an example, using the CRNN equalizer at -2 dBm launch power, we observe a Q-factor improvement of 0.73 dB in the SC case and a 2.14 dB improvement in the WDM case when compared to the reference Q-factor level (Regular DSP). This effect was also reported by means of numerical simulations in [4], where it was stated that LSTM layers are able to "deterministically" track sufficiently slow XPM effects in scenarios where no information on neighbor channels is fed into the NN-based equalizer. This demonstrates the efficacy of NN-based equalizers, which, in contrast to conventional methods, may partially recover such perturbation by approximating its average behavior in the received sequence.

Chapter 4

The Complexity Consideration for Designing Neural Networks-based Equalizers

This chapter addresses an essential aspect of the computational complexity necessary to create neural network-based equalizers. First, it will address the central issue of how to assess the computational complexity from a software level to a hardware level by mathematically articulating the computational complexity of various types of neural network layers. Next, it will analyze the crucial trade-off between performance and complexity to determine which NN structure provides the highest performance given the same degrees of complexity. Then, it will examine a variety of compression approaches that, when applied to NN-based equalizers, can dramatically reduce their computational complexity. Finally, it will briefly cover the development of neural networks in FPGA, demonstrating the experimental implementation and testing.

4.1 Metrics to estimate computational complexity of neural networks

4.1.1 Four Metrics of Computational Complexity

Accurate computational complexity evaluation is critical in the design of digital signal processing (DSP) devices to better understand the implementation feasibility and bottlenecks for each device's structure. With this in mind, the four most commonly used different criteria

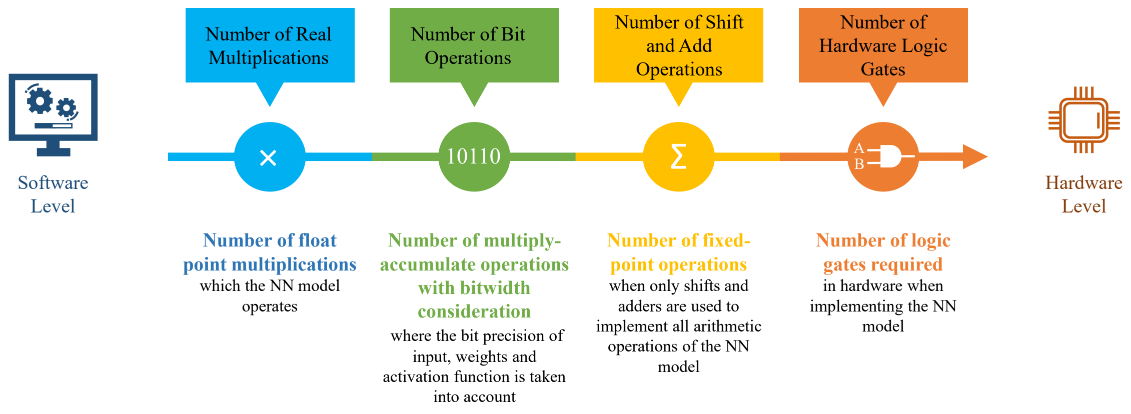

Number of Real Multiplications
Number of Bit Operations
Number of Shift and Add Operations
Number of Hardware Logic Gates
Software Level
×
10110
Σ
A
B
Hardware Level
Number of float point multiplications which the NN model operates
Number of multiply-accumulate operations with bitwidth consideration where the bit precision of input, weights and activation function is taken into account
Number of fixed-point operations when only shifts and adders are used to implement all arithmetic operations of the NN model
Number of logic gates required in hardware when implementing the NN model

Fig. 4.1 Diagram of computational complexity metrics, illustrating the various levels of complexity measurement from software to hardware.

for assessing computational complexity, from the software level to the hardware level, are summarized in Fig. 4.1.

The first, most software-oriented, level of estimation traditionally deals only with counting the number of real multiplications of the algorithm [117, 118] (quite often defined per one processed element, say a sample or a symbol). This metric is the number of real multiplications (RM). When comparing computational complexity, the purpose of this high-level metric is to consider only the multipliers required, ignoring additions, because the implementation of the latter in hardware or software is initially considered cheap, while the multiplier is generally the slowest element in the system and consumes the largest chip area [117, 119]. Ignoring additions can also be easily understood by looking at the Big-O analysis of multiplier versus adder. When multiplying two integers with n digits, the computational complexity of the multiplication instance is $O(n^2)$, whereas the addition of the same two numbers has a computational complexity of $\Theta(n)$ [120][1]. As a result, when you are dealing with float values with 16 decimal digits, multiplication is by far the most time-consuming part of the implementation procedure. Therefore, when comparing solutions that use floating-point arithmetic with the same bitwidth precision, the RM metric provides an acceptable comparative estimate to qualitatively assess the complexity against some existing benchmarks (e.g., against the DSP operations for optical channel equalization tasks [118]).

When moving to fixed-point arithmetic, the second metric known as the number of bit-operations (BOP) must be adopted to understand the impact of changing the bitwidth precision

[1]The Big-O notation represents the worst case or the upper bound of the time required to perform the operation, Big Omega (Ω) shows the best case or the lower bound; whereas the Big Theta (Θ) notation defines the tight bound of the amount of time required; in other words, $f(n)$ is claimed to be $\Theta(g(n))$ if $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$.

on the complexity. The BOP metric provides a good insight into mixed-precision arithmetic performance since it is possible to forecast the BOP needed for fundamental arithmetic operations like addition and multiplication given the bitwidth of two operands. In a nutshell, the BOP metric aims to generalize floating-point operations (FLOPs) to heterogeneously quantized NNs as far as the FLOPs cannot be efficiently used to evaluate integer arithmetic operations [121, 122]. For the BOP metric, it has to include the complexity contribution of both multiplications and additions, since now it evaluates the complexity in terms of the most common operations in NNs: the multiply-and-accumulate operations (MACs) [121–123]. However, the BOP accounts for the scaling of the number of multipliers with the bitwidth of two operands and the scaling of the number of adders with the accumulator bitwidth. Note that most real-world DSP implementations use dedicated logic macros, such as the DSP slice in Field Programmable Gate Arrays (FPGA) or the MAC in an Application Specific Integrated Circuit (ASIC). The BOP metric is a good way to estimate the complexity of a DSP implementation because it also uses the MAC and takes into account the bitwidth of two operands.

The progress in the development of new advanced NN quantization techniques [124–127] allowed implementing the fixed point multiplications participating in NNs efficiently, namely with the use of a few bit-shifters and adders [128–130]. Since the BOP lacks the ability to properly assess the effect of different quantization strategies on complexity, a new, more sophisticated metric is required there. Then, a third complexity metric is now introduced that counts the number of total equivalent additions to represent the multiplication operation, called the number of additions and bit shifts (NABS). The number of shift operations can be neglected when calculating the computational complexity because, in the hardware, the shift can be performed without extra costs in constant time with the $O(1)$ complexity. Even though the cost of bit shifts can be ignored due to the aforementioned reasons, and only the total number of adders has to be accounted for to measure the computational complexity, the metric name is kept as “number of additions and bit shifts” to highlight that the multiplication is now represented as shifts and adders.

Finally, the metric that is closest to the hardware level is the number of logic gates (NLG) that are used for evaluating the method’s hardware (e.g., ASIC or FPGA) implementation. It is different from the NABS metric, as now the true cost of implementation is to be presented. In this case, in contrast to the other complexity metrics, the cost of activation functions is also taken into account because, to achieve better complexity, they are frequently implemented using look-up tables (LUT) rather than adders and multipliers. Additionally, other metrics like the number of flip-flops (FFs) or registers, the number of logic blocks used for general logic and memory blocks, or other special functional macros used in the design, are also relevant. This explanation makes it clear that there will not be a simple equation to convert the NABS to the NLG because the latter depends on the circuit design that the developer chooses. Tools such as Synopsys Synthesis [131] for ASIC implementation can provide this kind of information. However, with regard to the FPGA design, it is harder to get a correct estimate of the gate count from the report of the FPGA tools [132].

This thesis advocates that the NLG metric should be applied to count the number of logic gates used to implement the hardware piece, similar to the concept of the Maximum Logic Gates metric for FPGA devices [133]. The Maximum Logic Gates metric is utilized to approximate the maximum number of gates that can be realized in the FPGA for a design consisting of only logic functions[2]. Additionally, this metric is based on an estimate of the typical number of usable gates per configurable logic block (CLB) or logic cell multiplied by the total number of such blocks or cells [133]. With regard to the correspondence between CLB and logic gate numbers, see Table 4.1.

Table 4.1 Capacity ranges for XC4000 Series CLB Resources given in Ref. [133].

CLB Resource	**Logic Gate Range**
Gate range per 4-input LUT (2 per CLB)	1 to 9
Gate range per 3-input LUT	1 to 6
Gate range per flip-flop (2 per CLB)	6 to 12
Total gate range per CLB	15 to 48
Estimated typical number of gates per CLB	28.5

It should be noted that Table 4.1 is based on an older, now obsolete, 4-input LUT architecture [133]. Newer FPGA families now feature a 6-input LUT architecture, and to address the resource consumption for the new generation of devices, a reasonable approximation would be to increase the ‘maximum gate range equivalent per LUT’ figure used in [133] by 50%. Note that the gate equivalence figures for FF’s (registers) still hold true for the 6-input architecture. It is also worth noting that the CLB architecture has changed substantially since Ref. [133] was published, such that Table 4.2 linking CLB-gates is presented next with a more up-to-date 6-input architecture.

Table 4.2 Estimated capacity ranges for 6 input LUT based CLB Resources.

CLB Resource	**Logic Gate Range**
Gate range per 6-input LUT (8 per CLB)	6 to 15
Gate range per flip-flop (16 per CLB)	6 to 12
Total gate range per CLB	144 to 312

To conclude, it is important to comment on the universal metrics between the FPGA and the ASIC implementations. Calculating an ASIC gate equivalent to an FPGA DSP slice is not

[2]On-chip memory capabilities are not factored into this metric.

a straightforward task because not all features are necessarily required when implementing the specific arithmetic function in an ASIC. However, utilizing the estimation approach laid out in Ref. [133], a figure can be obtained. Using the Xilinx Ultrascale + DSP48E2 slice basic multiplier functionality as an example (see Xilinx UG579 Fig. 1-1 in Ref. [134]) and pipelining it for maximum performance, it is possible to estimate the number of FFs and adders required for such an ASIC equivalence. Taking into account the structure of the multiplication of a m-bit number by a n-bit number, implemented using an array multiplier architecture, it is equivalent to $m \times n$ AND gates, n half adders, and $(m-2) \times n$ full adders[3]. For example, the ASIC equivalence of a 27×18 multiplier in an FPGA would have 486 AND gates, 18 half adders, 450 full adders, and 90 Flip Flops. Table. 4.3 shows an overview of what each metric is and when its application is relevant for comparative studies in the signal processing field.

Table 4.3 Overview of Main Metrics for Evaluating Computational Complexity in NN Models

Metric	Description	Usage
Number of Real Multiplications (RM)	Number of float point multiplications used in the NN model.	This metric is recommended as a first estimation when comparing with reference solutions with the same quantization level and using float point operations.
Number of Bit Operations (BoP)	Number of multiply-and-accumulate operations where the number of multiplications is weighted with the contribution of the quantization used by the input, activation function, and weights in the NN model.	This metric is recommended when bitwidth precision is used and needs to be taken into account when comparing complexity and performance with a reference solution.
Number of Add and Shift Operations (NABS)	Number of fixed-point operations used in the NN model when multiplications are implemented with bit shifters and adders.	This metric is recommended when different bitwidth precision and quantization strategies are used. For example, when the PoT quantization is used and one wants to show its complexity advantages against the Uniform quantization. If different quantization strategies are not used, the BoPs metric should be used.
Number of Hardware Logic Gates (NLG)	Number of logic gates required to implement the NN model in hardware.	This metric is recommended when the size of hardware needs to be assessed. It is highly dependent on the selected implementation approach. As an example, it can be used to compare the size of the NN model with other traditional blocks in ASIC.

4.1.2 Mathematical Complexity Formulation for Different Neural Network layers

In this section, a brief introduction is provided to various types of NN: dense layer, Convolutional Neural Networks (CNN), Vanilla Recurrent Neural Networks (RNN), Long Short-Term Memory Neural Networks (LSTM), Gated Recurrent Units (GRU), and Echo State Networks (ESN). Also, it investigated the computational complexity of each network in terms of RM, BOP, and NABS. In this study, the computational complexity is formulated per layer, and the output layer is not taken into account for the complexity calculation to eliminate redundant computations if multiple layers or multiple NN types are combined. Note that, in the interest

[3]Note that a half adder is equivalent to 1 AND gate + 1 XOR gate, and a full adder is equal to 2 AND gates + 2 XOR gates + 1 OR gate

of being extremely clear, some equations stated in Sec. 2.3 will be revisited for the purposes of showing clear derivations for the computational complexity metrics. Table 4.4 summarizes the formulas for the RM, BOP, and NABS for all NN types studied in this section.

Table 4.4 Summary of the three computational complexity metrics per layer (the number of real multiplications, the number of bit operations, the number of additions and bit shifts) for a zoo of neural network layers as a function of their designing hyper-parameters; the number of neurons (n_n), the number of features in the input vector (n_i), the number of filters (n_f), the kernel size (n_k), the input time sequence size (n_s), the number of hidden units (n_h), the number of internal hidden neuron units of the reservoir (N_r), sparsity parameter (s_p), the number of output neurons (n_o), weight bitwidth (b_w), input bitwidth (b_i), activation bitwidth (b_a), and the number of adders required at most to represent the multiplication (X_w). The expressions for Mult(a, b, c) and Acc(a, b, c) can be found in the main text, and represent the bitwidth cost of the accumulator.

Network type	Real multiplications (RM)	Number of bit-operations (BOP)	Number of additions and bit shifts(NABS)
MLP	$n_n n_i$	$n_n n_i\left[b_w b_i + \text{Acc}(n_i, b_w, b_i)\right]$	$n_n n_i(X_w+1)\text{Acc}(n_i, b_w, b_i)$
1D-CNN	$n_f n_i n_k \cdot OutputSize$	$OutputSize \cdot n_f \text{Mult}(n_i n_k, b_w, b_i)$ $+n_f \text{Acc}(n_i n_k, b_w, b_i)$	$OutputSize \cdot n_f\left[n_i n_k(X_w+1)-1\right]$ $\cdot\text{Acc}(n_i n_k, b_w, b_i)$ $+n_f \text{Acc}(n_i n_k, b_w, b_i)$
Vanilla RNN	$n_s n_h(n_i+n_h)$	$n_s n_h \text{Mult}(n_i, b_w, b_i)$ $+n_s n_h \text{Mult}(n_h, b_w, b_a)$ $+2n_s n_h \text{Acc}(n_h, b_w, b_a)$	$n_s n_h\left[n_i(X_w+1)-1\right]\text{Acc}(n_i, b_w, b_i)$ $+n_s n_h\left[n_h(X_w+1)+1\right]\text{Acc}(n_h, b_w, b_a)$
LSTM	$n_s n_h(4n_i+4n_h+3)$	$4n_s n_h \text{Mult}(n_i, b_w, b_i)$ $+4n_s n_h \text{Mult}(n_h, b_w, b_a)$ $+3n_s n_h b_a^2$ $+9n_s n_h \text{Acc}(n_h, b_w, b_a)$	$4n_s n_h\left[n_i(X_w+1)-1\right]\text{Acc}(n_i, b_w, b_i)$ $+4n_s n_h\left[n_h(X_w+1)+1\right]\text{Acc}(n_h, b_w, b_a)$ $+6n_s n_h b_a$
GRU	$n_s n_h(3n_i+3n_h+3)$	$3n_s n_h \text{Mult}(n_i, b_w, b_i)$ $+3n_s n_h \text{Mult}(n_h, b_w, b_a)$ $+3n_s n_h b_a^2$ $+8n_s n_h \text{Acc}(n_h, b_w, b_a)$	$3n_s n_h\left[n_i(X_w+1)-1\right]\text{Acc}(n_i, b_w, b_i)$ $+n_s n_h\left[3n_h(X_w+1)+5\right]\text{Acc}(n_h, b_w, b_a)$ $+6n_s n_h b_a$
ESN	$n_s N_r(n_i+N_r s_p+2+n_o)$	$n_s N_r \text{Mult}(n_i, b_w, b_i)$ $+n_s N_r s_p \text{Mult}(N_r, b_w, b_a)$ $+n_s N_r \text{Mult}(n_o, b_w, b_a)$ $+2n_s N_r b_a^2$ $+4n_s N_r \text{Acc}(N_r, b_w, b_a)$	$n_s N_r\left[n_i(X_w+1)-1\right]\text{Acc}(n_i, b_w, b_i)$ $+n_s N_r\left[s_p(N_r X_w+N_r-1\right]+4)\text{Acc}(N_r, b_w, b_a)$ $+n_s N_r\left[n_o(X_w+1)-1\right]\text{Acc}(n_o, b_w, b_a)$ $+4n_s N_r b_a$

Dense Layer

A dense layer, also known as a ‘fully connected layer’, is a layer in which each neuron is connected with all the neurons from the previous layer with a specific weight w_{ij}. Dense layers can be combined to form a Multi-Layer Perceptron (MLP), which is a class of a feed-forward deep NN.

The output vector y of a dense layer, given x as an input vector, is written as:

$$y = \phi(Wx+b), \tag{4.1}$$

where y is the output vector, ϕ is a nonlinear activation function, W is the weight matrix, and b is the bias vector. Writing explicitly the matrix operation inside the activation function:

$$Wx+b = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n_i} \\ w_{21} & w_{22} & \dots & w_{2n_i} \\ \vdots & \vdots & \dots & \vdots \\ w_{n_n1} & w_{n_n2} & \dots & w_{n_nn_i} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_i} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n_n} \end{bmatrix}, \tag{4.2}$$

where n_i is the number of features in the input vector and n_n represents the number of neurons in the layer, one can readily see that the RM of a dense layer can be computed according to the simple well-known formula:

$$\text{RM}_{\text{Dense}} = n_n n_i. \tag{4.3}$$

Now, the BOP of a dense layer is calculated, taking into account the bitwidth of two operands, to approximate the computational complexity of NNs when mixed-precision arithmetic is used. The bitwidth, also known as the precision, is the number of bits used to represent a certain element; for example, each weight in the weight matrix can be represented with b_w bits. Fig. 4.2 illustrates the data flow of the MAC operations for a neuron of a dense layer with n_i input features and b_i as input bitwidth. The multiplication of the input vector and the weights for one neuron can be mathematically represented as follows:

$$y_{\text{MUL, one neuron}} = \sum_{n=1}^{n_i} w_n x_n. \tag{4.4}$$

Initially, the n_i multiplications of input vector elements and weights for one neuron take place. When the multiplication of two operands is performed, the resulting bitwidth is the sum of the bitwidths of the two operands $(b_w + b_i)$ as it is shown in the first row of Fig. 4.2.

After that, $n_i - 1$ additions need to be made, and the resulting number of bits can be defined as follows: Considering that the result of the addition of two operands has the bitwidth of the bigger operand plus one bit, we start adding the multiplication results pairwise until only one element remains. In this case, the second row of Fig. 4.2 shows the first level of pairwise additions, with a resulting bitwidth of $b_w + b_i + 1$, and since this pairwise addition process is repeated for $\lceil \log_2(n_i) \rceil$ levels (until having just a final single number), the total

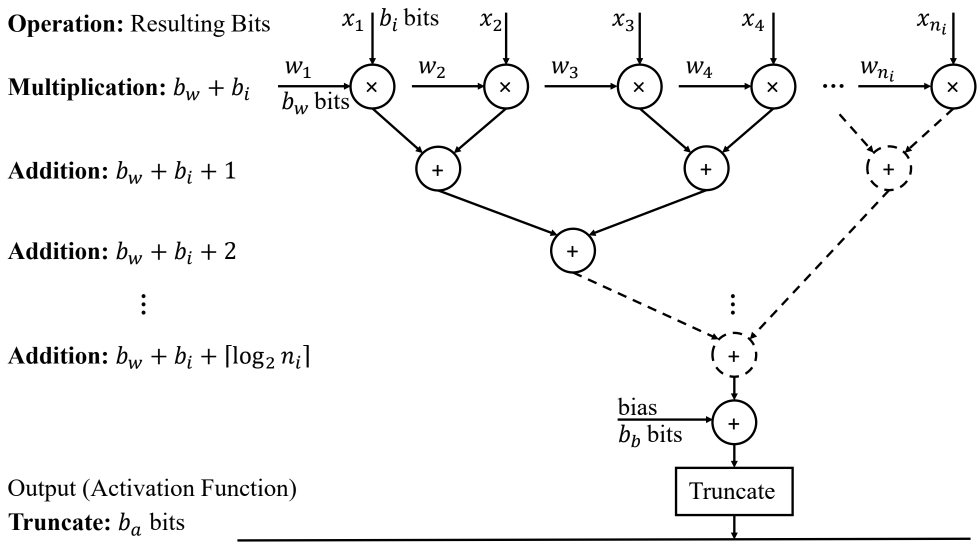

Operation: Resulting Bits
Multiplication: $b_w + b_i$
Addition: $b_w + b_i + 1$
Addition: $b_w + b_i + 2$
Addition: $b_w + b_i + \lceil \log_2 n_i \rceil$
Output (Activation Function)
Truncate: b_a bits
x_1
b_i bits
x_2
x_3
x_4
x_{n_i}
w_1
b_w bits
w_2
w_3
w_4
w_{n_i}
bias
b_b bits
Truncate

Fig. 4.2 Data path of a neuron in a quantized dense layer where x is the input vector with size n_i, w is the weight matrix, b_w is the weight bitwidth and b_i is the input bitwidth, b_a is the activation bitwidth and b_b is the bias bitwidth.

bitwidth of it is given by $b_w + b_i + \lceil \log_2(n_i) \rceil$, i.e., it is the bitwidth required to perform the overall MAC process. Then, the addition of the bias vector is performed. In this work, for all types of networks, it is assumed that the size of the accumulator, defined by the multiplication of the weight matrix and the input vector, is dominant; thereby, the assumption for the bias bitwidth b_b is as follows: $b_b < b_w + b_i + \lceil \log_2(n_i) \rceil$, and the addition of bias, in the end, will not result in the overflow. Finally, the bitwidth of the resulting number is truncated to b_a, where b_a is the bitwidth of the activation function [135].

When calculating the BOP for a dense layer, the costs of both multiplications and additions need to be included. Then, the BOP formula takes the form of the sum of two constituents, BOP_{Mul} and BOP_{Bias}, corresponding to vector-matrix multiplication and bias addition:

$$\text{BOP}_{\text{Mul}} = n_n \left[n_i b_w b_i + (n_i - 1)(b_w + b_i + \lceil \log_2(n_i) \rceil) \right], \tag{4.5}$$

$$\text{BOP}_{\text{Bias}} \approx n_n (b_w + b_i + \lceil \log_2(n_i) \rceil). \tag{4.6}$$

Eq. (4.5) shows the cost of the number of one-bit full adders calculated from the dot product of the n_i-dimensional input vector and weight matrix, as in Refs. [121, 136]. The cost takes into account the bitwidths of the weights and input, b_w and b_i. To compute the product of the two operands, it uses $n_i n_n$ multiplications and $n_n(n_i - 1)$ additions. The multiplication cost can be calculated by multiplying the number of multiplications by $b_w b_i$, which is related to the bit operation, and the number of additions by the accumulator bitwidth required to do the operation. The final BOP is the contribution of multiplication and the addition of bias from the dense layer. For the convenience of presentation, let us define the short notations:

$$\mathrm{Mult}(n_i, b_w, b_i) = n_i b_w b_i + (n_i - 1)\big(b_w + b_i + \lceil \log_2(n_i) \rceil\big),$$

and

$$\mathrm{Acc}(n_i, b_w, b_i) = b_w + b_i + \lceil \log_2(n_i) \rceil .$$

The Acc expression represents the actual bitwidth of the accumulator required for MAC operation, as shown in Fig. 4.2. Then, the BOP of the dense layer, expressed through the layer parameters, becomes:

$$\begin{aligned} \mathrm{BOP}_{\mathrm{Dense}} &= \mathrm{BOP}_{\mathrm{Mul}} + \mathrm{BOP}_{\mathrm{Bias}} \\ &\approx n_n n_i \big[b_w b_i + (b_w + b_i + \lceil \log_2(n_i) \rceil)\big] \\ &\approx n_n n_i \big[b_w b_i + \mathrm{Acc}(n_i, b_w, b_i)\big] . \end{aligned} \tag{4.7}$$

Now, note that with the advancement in NN quantization techniques, there arises the opportunity to approximate multiplication by using a few shifts and adding operations only while still maintaining a good processing accuracy, since the NNs can diminish the approximation error that the quantized approximation introduces[4] [129, 137]. As mentioned in Sec. 4.1.1, the number of shifts can be neglected compared to the contribution of adders. The number of adders is different for different types of quantization. To be more specific, let X represent the number of adders required, at most, to perform the multiplication and let b be the bitwidth of the quantized matrix. For uniform quantization: $X = b - 1$. And, for example, when the weight matrix with a bitwidth of b_w, is quantized, $X_w = b_w - 1$ as the number of adders it is needed at most to perform the multiplication of the weights [5]. In the case of Power-of-two (PoT) quantization: $X = 0$, because each multiplication costs just a shift [6, 128]. Lastly, for the Additive Powers-of-Two (APoT) quantization: $X = n$, where n denotes the number of additive terms. In APoT, the sum of n PoT terms is used to represent each quantization level [124]. Eventually, the NABS of a dense layer can be derived from its

[4]Note that using the shifts and adders to perform multiplications can cause some quantization noise/error since it is converting from a float-point representation to a fixed-point representation with some defined quantized levels. However, in NNs, this noise can be partially mitigated by including those quantized weights in the NN training process, as in Refs. [125–127]

[5]Note that it can be considered other techniques for the representation of such fixed-point multiplication to reduce its complexity. e.g., the double-base number system, where each multiplication with b bits, at worst, needs no more than $b/log(b)$ additions [138]. For the Canonical Signed Digit (CSD) representation, in the worst-case scenario, it is assumed $(b+l)/2$ nonzero bits, and on average, it tends asymptotically to $(3b+l)/9$ [139, 140]

BOP equation, Eq. (4.7):

$$\begin{aligned}\text{NABS}_{\text{Dense}} &\approx n_n n_i \left[X_w \text{Acc}(n_i, b_w, b_i) + \text{Acc}(n_i, b_w, b_i)\right] \\ &\approx n_n n_i (X_w + 1) \text{Acc}(n_i, b_w, b_i).\end{aligned} \tag{4.8}$$

In Eq. (4.8), the multiplication term $b_w b_i$ of Eq. (4.7) is converted into the number of adders needed to operate the multiplication times the accumulator bitwidth required: $X_w \text{Acc}(n_i, b_w, b_i)$.

Convolutional Neural Networks

In CNN, the convolutions are applied with different filters to extract the features and convert them into a lower-dimensional feature set, while still preserving the original properties. CNNs can be used in 1D, 2D, or 3D networks, depending on the applications. This thesis focuses on 1D-CNNs, which are applicable to processing sequential data [141]. For simplicity of understanding, the 1D-CNN processing with padding equal to 0, dilation equal to 1, and stride equal to 1, can be summarized as follows:

$$y_i^f = \phi\left(\sum_{n=1}^{n_i} \sum_{j=1}^{n_k} x_{i+j-1,n}^{in} \cdot k_{j,n}^f + b^f\right), \tag{4.9}$$

where y_i^f denotes the output, known as a feature map, of a convolutional layer built by the filter f in the i-th input element, n_k is the kernel size, n_i is the size of the input vector, x^{in} represents the raw input data, k_j^f denotes the j-th trainable convolution kernel of the filter f and b^f is the bias of the filter f.

In the general case, when designing the CNN, parameters like padding, dilation, and stride also affect the output size of the CNN. It can be formulated as:

$$OutputSize = \left\lfloor \frac{n_s + 2\,padding - dilation(n_k - 1) - 1}{stride} + 1 \right\rfloor, \tag{4.10}$$

where n_s is the input time sequence size.

The RM of a 1D-convolutional layer can be computed as follows:

$$\text{RM}_{\text{CNN}} = n_f n_i n_k \cdot OutputSize, \tag{4.11}$$

where n_f is the number of filters, also known as the output dimension. In Eq. (4.11), there are $n_i n_k$ multiplications per sliding window, and the number of times that sliding window process needs to be repeated is equal to the output size. Then, the procedure is executed repeatedly for all n_f filters.

The BOP for a 1D-convolutional layer, after taking into consideration the multiplications and additions, can be represented as:

$$\begin{aligned}\mathrm{BOP_{CNN}} &= OutputSize \cdot n_f \mathrm{Mult}(n_i n_k, b_w, b_i) \\ &+ n_f \mathrm{Acc}(n_i n_k, b_w, b_i).\end{aligned} \tag{4.12}$$

Eq. (4.12) is derived from Eq. (4.9) and Eq. (4.11). The first term is associated with the convolution operation between the flattened input vector and the sliding windows, and the latter term corresponds to the addition of the bias.

The procedure to derive the NABS is similar to that described in detail in the case of a dense layer. Given is the NABS of a 1D-convolutional layer:

$$\begin{aligned}\mathrm{NABS_{CNN}} &= OutputSize \cdot n_f \left[n_i n_k (X_w + 1) - 1 \right] \\ &\cdot \mathrm{Acc}(n_i n_k, b_w, b_i) \\ &+ n_f \mathrm{Acc}(n_i n_k, b_w, b_i).\end{aligned} \tag{4.13}$$

To obtain the 1D-convolutional layer's NABS, the multiplication in Eq. (4.12) is represented by the number of adders required, at most, to perform the multiplication times the accumulator bitwidth.

Vanilla Recurrent Neural Networks

Vanilla RNNs take into account the current input and the output that the network has learned from the prior input. The equation for the vanilla RNN given a time step t is as follows:

$$h_t = \phi(W x_t + U h_{t-1} + b), \tag{4.14}$$

where ϕ is, again, the nonlinear activation functions, $x_t \in \mathbb{R}^{n_i}$ is the n_i-dimensional input vector at time t, $h_t \in \mathbb{R}^{n_h}$ is a hidden layer vector of the current state with size n_h, $W \in \mathbb{R}^{n_h \times n_i}$ and $U \in \mathbb{R}^{n_h \times n_h}$ represent the trainable weight matrices, and b is the bias vector. For more explanations on the vanilla RNN operation, see Ref. [67]. The RM of a vanilla RNN is:

$$\mathrm{RM_{RNN}} = n_s n_h (n_i + n_h), \tag{4.15}$$

where n_h notes the number of hidden units. From Eq. (4.15), the RM for a time step is $n_h(n_i + n_h)$. It can be separated into two terms; the $n_h n_i$ term corresponds to the multiplication of the input vector x_t and the weight matrix, and the n_h^2 term arises because of the multiplication of the prior cell output h_{t-1}. Finally, n_s, which denotes the number of time steps in the layer, should be taken into account, as the process is repeated n_s times.

The BOP for a vanilla RNN is given as:

$$\begin{aligned}\mathrm{BOP}_{\mathrm{RNN}} = & \; n_s n_h \mathrm{Mult}(n_i, b_w, b_i) \\ & + n_s n_h \mathrm{Mult}(n_h, b_w, b_a) \\ & + 2 n_s n_h \mathrm{Acc}(n_h, b_w, b_a).\end{aligned} \tag{4.16}$$

From Eq. (4.16), the first term is associated with the input vector multiplied by the weight matrix, and the second term corresponds to the multiplications of the recurrent cell outputs. The final term is the contribution of the addition between $Wx_t + Uh_{t-1}$ and the addition of the bias vector in Eq. (4.14); one can see that the size of the accumulator used in this term, is $\mathrm{Acc}(n_h, b_w, b_a)$. It is due to the assumption that $\mathrm{Acc}(n_h, b_w, b_a)$ is dominant because it should be greater than $\mathrm{Acc}(n_i, b_w, b_i)$ as a result of the inequality $n_h > n_i$.

As in the case of a dense layer, the NABS of vanilla RNN can be calculated from its BOP equation by converting the multiplication to the number of adders needed at most (X) depending on the quantization scheme and the accumulator size:

$$\begin{aligned}\mathrm{NABS}_{\mathrm{RNN}} = & \; n_s n_h \big[n_i(X_w + 1) - 1\big] \mathrm{Acc}(n_i, b_w, b_i) \\ & + n_s n_h \big[n_h(X_w + 1) + 1\big] \mathrm{Acc}(n_h, b_w, b_a).\end{aligned} \tag{4.17}$$

Long Short-Term Memory Neural Networks

LSTM is an advanced type of RNNs. Although RNNs suffer from short-term memory issues, the LSTM network has the ability to learn long-term dependencies between time steps (t), insofar as it was specifically designed to address the gradient issues encountered in RNNs [69, 70]. There are three types of gates in an LSTM cell: an input gate (i_t), a forget gate (f_t), and an output gate (o_t). More importantly, the cell state vector (C_t) was proposed as a long-term memory to aggregate the relevant information throughout the time steps. The equations for the forward pass of the LSTM cell given a time step t are as follows:

$$\begin{aligned} i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i), \\ f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f), \\ o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o), \\ C_t &= f_t \odot C_{t-1} + i_t \odot \phi(W^c x_t + U^c h_{t-1} + b^c), \\ h_t &= o_t \odot \phi(C_t), \end{aligned} \tag{4.18}$$

where ϕ is usually the "tanh" activation functions, σ is usually the sigmoid activation function, the sizes of each variable are $x_t \in \mathbb{R}^{n_i}$, $f_t, i_t, o_t \in (0,1)^{n_h}$, $C_t \in \mathbb{R}^{n_h}$ and $h_t \in (-1,1)^{n_h}$. The $\odot$ symbol represents the element-wise (Hadamard) multiplication.

The RM of a LSTM layer is:

$$\mathrm{RM}_{\mathrm{LSTM}} = n_s n_h (4n_i + 4n_h + 3), \tag{4.19}$$

where n_h is the number of hidden units in the LSTM cell. Similarly to RNNs, the RM can be calculated from the term associated with the input vector x_t and the term corresponding to the prior cell output h_{t-1}; however, each term occurs four times, as it can be seen in Eq. (4.18). Therefore, we have $4n_h n_i$ and $4n_h^2$, respectively. Moreover, we also need to include the element-wise product that is operated three times in Eq. (4.18), which costs $3n_h$. Finally, the process is repeated n_s times, hence, n_s is multiplied to the overall number.

The BOP for a LSTM layer is computed based on Eq. (4.19), but also includes the bitwidth of the operands and the number of additions. As a result, the BOP can be represented as:

$$\begin{aligned}\mathrm{BOP}_{\mathrm{LSTM}} = {} & 4n_s n_h \mathrm{Mult}(n_i, b_w, b_i) \\ & + 4n_s n_h \mathrm{Mult}(n_h, b_w, b_a) \\ & + 3n_s n_h b_a^2 \\ & + 9n_s n_h \mathrm{Acc}(n_h, b_w, b_a).\end{aligned} \tag{4.20}$$

To give more details on the expression, the first two terms in Eq. (4.20) are the contribution of the input vector multiplications and the recurrent cell output association, respectively. The term $3n_s n_h b_a^2$ refers to three times the element-wise product of two operands with b_a bitwidth; see Eq. (4.18). For each time step, there are n_h elements in each vector that need to be multiplied. The last term is for all the additions, since it is assumed that $\mathrm{Acc}(n_h, b_w, b_a)$ gives the dominant contribution, as described in Sec. 4.1.2. Finally, the process is restarted n_s times.

The NABS of a LSTM layer is derived from the Eq. (4.20) by replacing the multiplications with the shifts and adders including their cost, as mentioned in Sec. 4.1.1 that the shifts would not be included. The number of adders depends on the quantization technique. The NABS would be as follows:

$$\begin{aligned}\mathrm{NABS}_{\mathrm{LSTM}} = {} & 4n_s n_h \big[n_i(X_w + 1) - 1\big] \mathrm{Acc}(n_i, b_w, b_i) \\ & + 4n_s n_h \big[n_h(X_w + 1) + 1\big] \mathrm{Acc}(n_h, b_w, b_a) \\ & + 6n_s n_h b_a.\end{aligned} \tag{4.21}$$

The first and second terms are the results of input vector multiplications and the recurrent cell operation combined with all addition operations, respectively. In this case, the third term comes from $3n_s n_h(b_a + b_a)$. Due to the element-wise product of two operands with bitwidth b_a, the resulting bitwidth becomes $b_a + b_a$ as mentioned in Fig. 4.2.

Gated Recurrent Units

Like LSTM, the GRU network was created to overcome the short-term memory issues of RNNs. However, GRU is less complex, as it has only two types of gates: reset (r_t) and update (z_t) gates. The reset gate is used for short-term memory, whereas the update gate is responsible for long-term memory [142]. In addition, the candidate hidden state (h_t') is also introduced to state how relevant the previous hidden state is to the candidate state. The GRU for a time step t can be formalized as:

$$\begin{aligned} z_t &= \sigma(W^z x_t + U^z h_{t-1} + b^z), \\ r_t &= \sigma(W^r x_t + U^r h_{t-1} + b^r), \\ h_t' &= \phi(W^h x_t + r_t \odot U^h h_{t-1} + b^h), \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot h_t', \end{aligned} \tag{4.22}$$

where ϕ is typically the “tanh” activation function and the rest of designations are the same as in Eq. (4.18).

The RM of the GRU is calculated in the same way as we did for the LSTM in Eq. (4.19), but the number of operations with the input vector x_t and with the previous cell output h_{t-1} is reduced from four (LSTM) to three times as shown in Eq. (4.22). Thus, the expression for the RM becomes:

$$\text{RM}_{\text{GRU}} = n_s n_h (3n_i + 3n_h + 3). \tag{4.23}$$

The BOP for the GRU can be calculated in the same manner as for the LSTM in Eq. (4.20). However, now the expression is slightly different in the number of matrix multiplications as the number of gates is now lower. The BOP number can be represented as:

$$\begin{aligned} \text{BOP}_{\text{GRU}} = {} & 3n_s n_h \text{Mult}(n_i, b_w, b_i) \\ & + 3n_s n_h \text{Mult}(n_h, b_w, b_a) \\ & + 3n_s n_h b_a^2 \\ & + 8n_s n_h \text{Acc}(n_h, b_w, b_a). \end{aligned} \tag{4.24}$$

The explanation for each line here is identical to that in Eq. (4.20).

The NABS of the GRU is derived similarly to the LSTM case:

$$\begin{aligned}\text{NABS}_{\text{GRU}} = {} & 3n_s n_h \left[n_i(X_w+1)-1\right]\text{Acc}(n_i,b_w,b_i) \\ & +n_s n_h\left[3n_h(X_w+1)+5\right]\text{Acc}(n_h,b_w,b_a) \\ & +6n_s n_h b_a.\end{aligned} \tag{4.25}$$

Again, the explanation for each term in this expression is identical to Eq. (4.21).

Echo State Networks

ESN belongs to the class of recurrent layers, but more specifically, to the reservoir computing category. ESN was proposed to relax the training process while being efficient and simple to implement. The ESN comprises three layers: an input layer, a recurrent layer, known as a reservoir, and an output layer, which is the only layer that is trainable. The reservoir with random weight assignment is used to replace back-propagation in traditional NNs to reduce the computational complexity of training [82]. This work only examined the digital domain implementation. Moreover, it focuses on the leaky-ESN, as it is believed to often outperform standard ESNs and is more flexible due to timescale phenomena [78, 143]. The equations of the leaky-ESN for a certain time step t are given as:

$$a_t = \phi\left(W^r s_{t-1} + W^{\text{in}} x_t\right), \tag{4.26}$$

$$s_t = (1-\mu)s_{t-1} + \mu a_t, \tag{4.27}$$

$$y_t = W^o s_t + b^o, \tag{4.28}$$

where s_t represents the state of the reservoir at time t, W^r denotes the weight of the reservoir with the sparsity parameter s_p, W^{in} is the weight matrix that shows the connection between the input layer and the hidden layer, μ is the leaky rate, W^o denotes the trained output weight matrix, and y_t is the output vector.

The RM of an ESN is given by:

$$\text{RM}_{\text{ESN}} = n_s N_r(n_i + N_r s_p + 2 + n_o), \tag{4.29}$$

where N_r is the number of internal hidden neuron units of the reservoir and n_o denotes the number of output neurons. From Eq. (4.29), the $N_r n_i$ multiplications occur from the input vector operations, and the term $N_r^2 s_p$ is included due to the reservoir layer; to be more specific, the latter term is multiplied with the sparsity parameter s_p which indicates the ratio of zero values in the matrix. Eq. (4.27) results in $2N_r$ multiplications. Unlike the other network types, now it is necessary to explicitly include the contribution of the output layer because it contains the trainable weight matrix, and this layer contributes $N_r n_o$ multiplications. Eventually, the process is repeated for n_s times.

The BOP number for an ESN can be represented as:

$$\begin{aligned}\mathrm{BOP}_{\mathrm{ESN}} = {} & n_s N_r \mathrm{Mult}(n_i, b_w, b_i) \\ & + n_s N_r s_p \mathrm{Mult}(N_r, b_w, b_a) \\ & + n_s N_r \mathrm{Mult}(n_o, b_w, b_a) \\ & + 2 n_s N_r b_a^2 \\ & + 4 n_s N_r \mathrm{Acc}(N_r, b_w, b_a).\end{aligned} \tag{4.30}$$

In Eq. (4.30), the first term is the input vector contribution, the second one is contributed by the reservoir layer, the third term refers to the output layer multiplications, and the fourth term stems from the multiplications in Eq. (4.27). Eventually, the final term will account for all the added operations.

The NABS of an ESN, which can be calculated similarly as in the LSTM case in Sec. 4.1.2, is:

$$\begin{aligned}\mathrm{NABS}_{\mathrm{ESN}} = {} & n_s N_r \left[n_i (X_w + 1) - 1 \right] \mathrm{Acc}(n_i, b_w, b_i) \\ & + n_s N_r \left[s_p (N_r X_w + N_r - 1 \right] + 4) \mathrm{Acc}(N_r, b_w, b_a) \\ & + n_s N_r \left[n_o (X_w + 1) - 1 \right] \mathrm{Acc}(n_o, b_w, b_a) \\ & + 4 n_s N_r b_a.\end{aligned} \tag{4.31}$$

By changing the multiplication terms in Eq. (4.30) to the number of adders required at most, we obtain the ESN's NABS. The input vector multiplication contributes to the first term. The reservoir layer and all the addition operations result in the second term. The third term comes from the output layer of the ESN. The last term is the contribution of Eq. (4.27).

4.2 Computational complexity versus Performance trade-off of data-driven neural network equalizer in coherent transmissions

This section initially describes the numerical and experimental scenarios used in this study to analyze and compare the functioning of the equalizers detailed in Sec. 2.3 and 3.2. After that, the two types of analysis for the set of NN structures are carried out. First, it presents the maximum performance improvement (in terms of Q-factor gain compared to the non-equalized case) that each equalizer can deliver and compares this gain to the respective computational complexity corresponding to each optimized equalizer. Then, it decreases the computational complexity of six NN topologies from Sec. 3.2 and presents the gain improvement provided by each NN-equalizer when all NNs have approximately the same computational complexity. This enables it to investigate the dependence of optical performance on computational complexity and identify which equalizer is better for a certain complexity level.

4.2.1 The computational complexity of the NN-based equalizers

Let the mini-batch size be B, n_s be the input time sequence size, with $n_s = M$, where M is the memory size (see also Sec. 3.2), and n_i be the number of features, which in this case is equal to 4. Since it is recovered the real and imaginary parts of each symbol, the number of outputs per symbol, n_o, is equal to 2. For ESN, biLSTM, and CNN layers, as they require inputs in the form of tensors of rank 3, the input of the NN equalizer can be parametrized as $[B, n_s, n_i]$, the three numbers defining the dimensions of the input tensor, as mentioned above. The parametrization for the MLP equalizer is simpler, with $[B, n_s \cdot n_i]$ defining the dimensions of the 2D tensor input. Flattening layers are used when it is necessary to reduce the dimensionality of the data.

In this case, considering three dense layers with n_1, n_2, and n_3 neurons, respectively, the complexity C_{MLP} of the resulting NN is given by:

$$C_{\text{MLP}} = \underbrace{n_s n_i n_1}_{a_1} + \underbrace{n_1 n_2 + n_2 n_3}_{b_1} + \underbrace{n_3 n_o}_{c_1}, \tag{4.32}$$

where a_1 is the contribution of the input layer, b_1 is the contribution of the hidden layer, and c_1 is the contribution of the output layer. The subindex “1” in a, b, and c explicitly associates these parameters with the MLP architecture.

The next part presents the computational complexity of an NN-based equalizer composed of a biLSTM layer. Assuming that the biLSTM layer has n_h hidden units, the complexity of such a NN is given by:

$$C_{\text{biLSTM}} = 2\underbrace{n_s n_h (4n_i + 4n_h + 3 + n_o)}_{a_2}, \tag{4.33}$$

$$C_{\text{biLSTM}} = 2\underbrace{n_s (4n_i n_h + 4n_h^2 + 3n_h + n_o n_h)}_{a_2}, \tag{4.34}$$

$$C_{\text{biLSTM}} = 2\underbrace{n_s (A + B + 3n_h) + C}_{a_2}, \tag{4.35}$$

where a_2 is the contribution of the only layer, while the subindex “2” attributes the number a to the biLSTM. This expression is easier to understand if you analyze the mathematical description of the LSTM cell, see (2.19) and Fig. 2.2. It has several contributions to the cell's complexity. In the first layer, it has $4n_i n_h$ multiplications associated with the input vector x_t. Then, $4n_h^2$ multiplications are due to the operations with the previous cell output h_{t-1}. Afterward, $3n_h$ and $n_o n_h$ multiplications due to the internal multiplications identified with $\odot$ and involving the current cell output (h_t) going into the output layer, respectively, are added. Lastly, it multiplies the number of operations by the number of time steps in the layer, n_s. Since the topology is bidirectional, the total contribution is also multiplied by 2.

Following Sec. 2.3, now the computational complexity associated with the ESN equalizer is addressed. Before presenting the respective expression, it is important to emphasize two aspects. First, the implementation of the ESN in the digital domain does not benefit from the fact that only the output layer weights are trainable, since, as mentioned previously, the training is not a key bottleneck as it is carried out during the offline calibration process. Second, the complexity of the ESN can potentially drop drastically if it is implemented in the optical domain as an ESN dynamic layer, as noted in [80]. However, this study analyzes the ESN implementation in the digital domain, similarly to [79].

Considering the leaky-ESN definition given by (2.20)–(2.22), the computational complexity of this equalizer can be expressed as:

$$C_{\text{ESN}} = n_s \left(\underbrace{n_i N_r + N_r^2 s_p}_{a_3} + \underbrace{2N_r}_{b_3} + \underbrace{N_r n_o}_{c_3} \right). \tag{4.36}$$

In the expression above, a_3 represents the contributions of (2.20), where the input layer adds $n_i N_r$ multiplications whereas the dynamic layers add $N_r^2 s_p$. b_3 refers to the contributions of (2.21) describing the static layer, and c_3 represents the multiplications in the output layer, (2.22). This overall process is repeated for all n_s time steps. Note that in the case of a potential optical implementation of the ESN, a_3 and b_3 would be equal to zero, and only the final weights would be learned in the digital domain.

Finally, the complexity of the composite structures (CNN+MLP and CNN+biLSTM) is addressed. The computational complexity of a 1-D convolutional layer is described as:

$$C_{\text{CNN}} = n_i n_f n_k \left[\frac{n_s + 2\,padding - dilation(n_k - 1) - 1}{stride} + 1 \right] \tag{4.37}$$

However, (2.17) assumes that the convolutional layer is defined by the number of filters n_f, the kernel size n_k, and that the number of time steps $n_s \geq n_k$, according to (2.16) the output size for each filter of the CNN is $(n_s - n_k + 1)$. Equations (4.38) and (4.39) are the expressions for the complexity of a convolutional layer combined with two dense layers or one biLSTM layer, respectively:

$$\begin{aligned} C_{\text{CNN+MLP}} = {} & \underbrace{n_i n_f n_k (n_s - n_k + 1)}_{a_4} + \\ & \underbrace{(n_s - n_k + 1) n_f}_{b_4} \underbrace{n_1 + n_1 n_2 + n_2 n_o}_{c_4}, \end{aligned} \tag{4.38}$$

$$\begin{aligned} C_{\text{CNN+biLSTM}} = {} & \underbrace{n_i n_f n_k (n_s - n_k + 1)}_{a_5} + \\ & \underbrace{(n_s - n_k + 1)}_{b_5} \underbrace{2 n_h [4 n_f + 4 n_h + 3 + n_o]}_{c_5}. \end{aligned} \tag{4.39}$$

In this scenario, the two-layer MLP has n_1 and n_2 neurons in each layer, and the biLSTM layer has n_h hidden units. In the equations above, a_4 and a_5 are the contributions of the convolutional layer, b_4 is the correction factor for the transition between layers since the flattening layer was placed before the dense layers; b_5 is the number of time-steps for the following biLSTM layer; c_4 is the contribution of the two-layer MLP; and c_5 is the contribution of the biLSTM layer where, in this case, the number of filters, n_f, is equal to the number of features entering the LSTM cell.

Finally, the computational complexity of the DBP-based receiver is used in this study for benchmark purposes. A basic implementation of the DBP algorithm [5] is considered, where each propagation step comprises a linear part for dispersion compensation followed by a nonlinear phase cancellation stage. With a zero-forcing equalizer, you change the signal in the frequency domain and multiply it by the inverse dispersion transfer function of the propagation section to get the linear part. The complexity of the DBP in terms of real multiplication per recovered symbol (RMpS) is [58, 118]:

$$C_{\text{DBP}} = 4N_{span}N_{step}\left(\frac{nN_{\text{FFT}}[\log_2(N_{\text{FFT}})+1]}{(N_{\text{FFT}}-N_D+1)}+n\right), \tag{4.40}$$

where N_{step} is the number of steps per span used, N_{FFT} is the FFT size, n is the oversampling ratio, and $N_D = \tau_D/T$, where τ_D corresponds to the dispersive channel impulse response and $T = 1/R_s$ is the symbol duration. It has been considered that $N_{\text{FFT}} = 256$ and τ_D defined as:

$$\tau_D = \frac{1.1R_s c|D|L_{span}}{f_c^2 N_{steps}}, \tag{4.41}$$

where f_c is the optical carrier reference frequency that in this case was 193.4 THz, c is the speed of light, L_{span} is the span length and D is the fiber dispersion parameter.

4.2.2 Simulation and experimental setups

The setup used in this experiment is depicted in Fig. 4.3[6]. At the transmitter, a DP-16QAM 34.4 GBd symbol sequence was mapped out of data bits generated by a $2^{32}-1$ PRBS. Then, a digital RRC filter with roll-off 0.1 was applied to limit the channel bandwidth to 37.5 GHz. The resulting filtered digital samples were resampled and uploaded to a digital-to-analog converter (DAC) operating at 88 GSamples/s. The outputs of the DAC were amplified by a four-channel electrical amplifier which drove a dual-polarization in-phase/quadrature Mach–Zehnder modulator, modulating the continuous waveform carrier produced by an external cavity laser at $\lambda = 1.55\,\mu m$. The resulting optical signal was transmitted over 9×50 km spans of TWC optical fiber with EDFA amplification. The optical amplifier noise figure was in the 4.5 to 5 dB range. The parameters of the TWC fiber – at $\lambda = 1.55\,\mu m$ – are: attenuation coefficient $\alpha = 0.23$ dB/km, dispersion coefficient $D = 2.8$ ps/(nm·km), and effective nonlinear coefficient $\gamma = 2.5\ (W \cdot km)^{-1}$.

On the RX side, the optical signal was converted into the electrical domain using an integrated coherent receiver. The resulting signal was sampled at 50 Gsamples/s by a digital sampling oscilloscope and processed by an offline DSP based on the algorithms

[6]In this section, the transmission case considered was a single-channel transmission, so no neighbor channels were added.

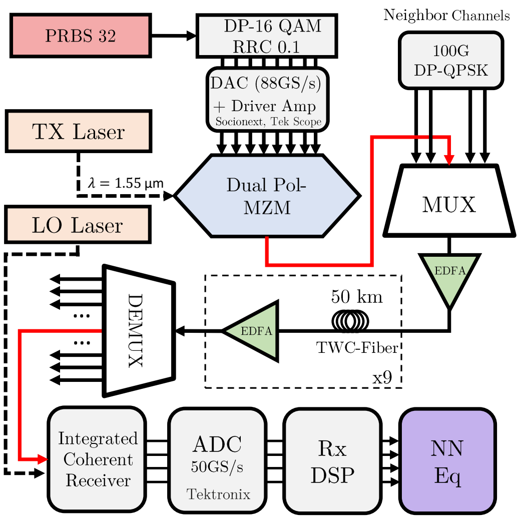


Fig. 4.3 Experimental setup used to analyze the performance of different NN equalizers; The input of the NN (shown as the purple rectangle at the bottom right) is the soft output of the regular DSP just before the decision unit.

described in [113]. Following the removal of the carrier frequency offset, a frequency domain equalizer first compensated for the bulk accumulated dispersion. A constant-amplitude, zero-autocorrelation-based training sequence was then located in the received frame, and the equalizer transfer function was estimated from it. After the equalization, the two polarizations were demultiplexed, and the signal was corrected for clock frequency and phase. Carrier phase estimation was then achieved with the help of pilot symbols. Thereafter, the resulting soft symbols were used as input for the NN equalizers. Finally, the pre-FEC BER was evaluated from the signal at the NN output.

With regard to simulation, it mimicked the experimental transmission setup[7]. The optical signal propagation along an optical fiber was simulated by solving the Manakov equations via the split-step Fourier method (with a resolution of 1km per step). Every span was followed by an optical amplifier with the noise figure $\mathrm{NF} = 4.5$ dB, which fully compensates for fiber losses and adds amplified spontaneous emission noise. At the receiver, the received symbols were made the same as the ones that were sent after full electronic chromatic dispersion compensation (CDC) by the frequency-domain equalizer and downsampling to the symbol rate. Finally, Gaussian noise was added to the signal, representing an additional transceiver distortion that the experiment may have, such that the Q-factor level of the simulated data

[7]It was considered a DP-16QAM, single-channel signal at 34.4 GBd pre-shaped by an RRC filter with 0.1 roll-off transmissions with an upsampling rate of 8 samples per symbol (275.2 GSamples/s) over a system consisting of 9×50 km TWC-fiber spans.

matched the experimental one. The system's performance is evaluated in terms of the Q-factor.

4.2.3 Results for Optimized NN-based architectures

In this section, the maximum achievable Q-factor for all equalizers is shown without constraining the computational complexity. The Bayesian optimization (BO) tool, introduced in [116] for optical NN-based equalizers, was implemented to identify the optimum values of hyper-parameters for each NN topology, which provides the best Q-factor in the experimental test dataset. As it was recently shown, the BO renders superior performance compared to other types of search algorithms for machine learning hyperparameter tuning [144]. The same topologies (without further optimization) were tested for the numerical analysis as well. The search space used in the BO procedure was defined via the allowed hyper-parameters intervals: N =[1 to 50], n_f =[1 to 1000], n_k =[1 to 20], n_h =[1 to 1000], n_1 =[1 to 1000], n_2 =[1 to 1000], n_3 =[1 to 1000], N_r =[1 to 1000], s_p =[0 to 1], μ =[0 to 1], and spectral radius=[0 to 1] .

In Table. 4.5, the line marked with the "**Best Topology**" label, summarizes the hyper-parameters obtained by the BO. These values are used to count the real multiplications per symbol recovery (complexity), and to assess the equalizers' performance expressed via the Q-factor gain, Fig. 4.4. Note that for all equalizers, the same optimal number of taps found by the BO was $N = 20$, which means that the memory in all equalizers is $M = 41$ and the mini-batch size, B, is equal to 4331. Moreover, for the ESN, the BO found the best value $\mu = 0.57$, and the optimal spectral radius (0.667). The activation functions found for every hidden NN layer are summarized as follows: 1D-CNN layer – 'linear' activation function followed by *LeakyReLU* (Leaky version of a Rectified Linear Unit) with a negative slope coefficient alpha = 0.2; biLSTM layer – hyperbolic tangent ('tanh') activation function; ESN layer – 'tanh' activation function; MLP layer – 'tanh' activation function.

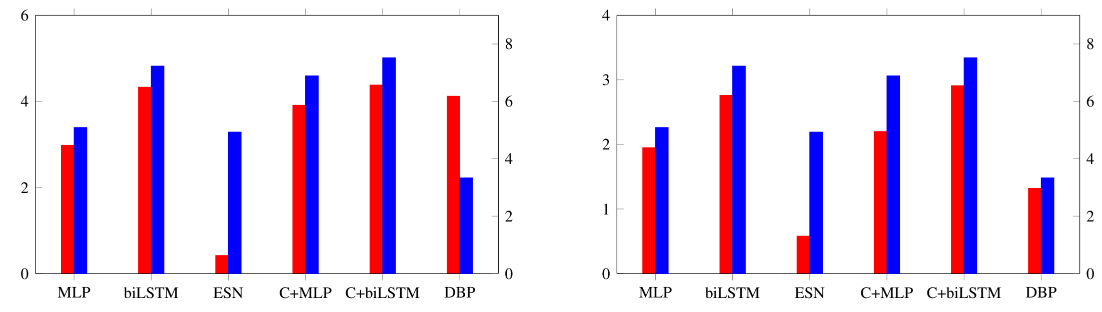
The results obtained by using the numerical synthetic data are presented in Fig. 4.4a. First, the CNN+biLSTM [8] turned out to be best-performing in terms of the Q-factor gain: it achieved a 4.38 dB Q-factor improvement when compared to the conventional DSP algorithms [113], 0.05 dB when compared to the biLSTM equalizer level, 0.47 dB when compared to the CNN+MLP equalizer level, 1.4 dB when compared to the MLP equalizer level, and 3.96 dB when compared to the ESN equalizer level. Second, when adding the convolutional layers to MLP and biLSTM, an improvement was observed in terms of the number of epochs needed to reach the highest performance: the single-layer biLSTM required

[8]Note that the CNN+biLSTM belongs to the same category as the CRNN – convolutional recurrent neural network.

Table 4.5 Summary of the complexity attributing to each NN equalizer topology: the topology type is identified in the leftmost column. The complexity corresponding to each topology and the NN type is expressed in terms of real multiplications per symbol recovered (RMpS), highlighted in red. This table also depicted the hyper-parameters distributions found by the BO: the cell marked as "Best topology" and the other 6 topologies (Topologies from 1 to 6, referring to the increasing complexity threshold number) for the study of complexity versus performance. In addition, for all topologies, the values of n_s, n_i, and n_o were: 41, 4, and 2, respectively, and these are not reported in the table.

	CNN+biLSTM				biLSTM		ESN		
Best Topology	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	244	10	226	2.7E+07	226	1.7E+07	88	0.18	8.6E+04
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	470	10	456	467	7.7E+06	149	132	596	1.2E+05
Topology 1	**CNN+biLSTM**				**biLSTM**		**ESN**		
	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	1	10	1	2.1E+03	1	2.0E+03	6	0.18	2.2E+03
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	2	5	10	10	2.3E+03	10	10	25	2.0E+03
Topology 2	**CNN+biLSTM**				**biLSTM**		**ESN**		
	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	5	10	3	1.3E+04	4	1.2E+04	22	0.18	1.1E+04
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	9	5	12	30	1.1E+04	40	40	80	1.1E+04
Topology 3	**CNN+biLSTM**				**biLSTM**		**ESN**		
	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	20	10	10	1.1E+05	16	1.1E+05	100	0.18	1.1E+05
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	50	9	30	100	1.1E+05	170	170	300	1.1E+05
Topology 4	**CNN+biLSTM**				**biLSTM**		**ESN**		
	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	50	10	41	1.0E+06	53	1.0E+06	350	0.18	1.0E+06
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	300	10	70	200	1.1E+06	600	600	900	1.0E+06
Topology 5	**CNN+biLSTM**				**biLSTM**		**ESN**		
	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	244	10	108	1.0E+07	172	1.0E+07	1150	0.18	1.0E+07
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	600	12	500	500	1.0E+07	2100	2100	2500	1.0E+07
Topology 6	**CNN+biLSTM**				**biLSTM**		**ESN**		
	n_f	n_k	n_h	RMpS	n_h	RMpS	N_r	s_p	RMpS
	400	10	455	1.0E+08	550	1.0E+08	3660	0.18	1.0E+08
	CNN+MLP					**MLP**			
	n_f	n_k	n_1	n_2	RMpS	n_1	n_2	n_3	RMpS
	1000	10	2900	2200	1.0E+08	7050	7050	7000	1.0E+08

119 epochs, while the CNN+biLSTM reduced this number to 89 epochs; the MLP itself needed 214 epochs to reach the best performance level, and the CNN+MLP required just 100 epochs. Thus, one can conclude that the addition of a convolutional layer indeed enhances the NN structure's performance and assists in the training stage.



(a) Simulation results (b) Experimental results

Fig. 4.4 Comparison of the computational complexity versus performance for the different NN-based equalizers considered within this study with their optimized architectures and the DBP with 3 StPS. The C+MLP and C+biLSTM represent the CNN+MLP and CNN+biLSTM architectures, respectively. The number over each bar gives the 10 logarithm of the number of multiplications per recovery symbol.

When considering how NN equalizers function with the experimental data, Fig. 4.4b, two major observations can be highlighted. First, similarly to the numerical results, the CNN+biLSTM is best-performing among all the considered NN structures in terms of the Q-factor gain. The CNN+biLSTM demonstrated a 2.91 dB improvement when compared to the conventional DSP, 0.15 dB when compared to the biLSTM equalizer, 0.61 dB when compared to the CNN+MLP equalizer, 0.96 dB when compared to the MLP equalizer, and 2.33 dB when compared to the ESN equalizer. Additionally, as was also observed in the numerical analysis, a lower number of training epochs was necessary to reach the best performance point when a convolutional layer was added: using the CNN+biLSTM needed 169 epochs, while for the pure biLSTM this number was 232 epochs; the number of epochs required for the CNN+MLP to reach the best performance was 107, and for the pure MLP it was 753 epochs. Second, compared to the simulation, the overall gain of all NN-based equalizers is slightly reduced. This can be explained by the existing "reality gap" between the numerical model and the true experimental transmission results. In a real transmission, extra nonlinearity and the non-ideal behavior of transceivers (signal clipping by the ADC/DAC, harmonic and intermodulation distortions of the driver amplifier (DA), I/Q skew, etc.) add noise and complexity to the channel inversion process. With just the split-step method, the NNs can unroll the synthetic propagation effects more easily than reverting the actual propagation in the experimental condition. Even though the gain numbers are different in the numerical and experimental data, the NN structures' performance followed the same pattern for both numerical and experimental cases: the best performance was attributed to the CNN+biLSTM, the next-level performance pertains to the biLSTM, followed by the CNN+MLP, the MLP and, finally, the ESN.

Finally, of all equalizer types investigated in this study, the DBP 3 StPS applied with two samples per symbol was still the least complex method. In all simulation and experiment test cases, however, the CNN+biLSTM outperformed the 3 StPS DBP, as shown in Fig. 4.4. Even by optimizing the DBP's nonlinear coefficient parameter (γ), the DBP approach was able to enhance the Q-factor only by 1.32 dB, whereas the CNN+biLSTM equalizer improved it by 2.91 dB, in the experimental case. The boost in performance in the experiment scenario provided by the CNN+biLSTM relative to the DBP demonstrates the NN-equalizer's power in mitigating transmission impairments in a practical application.

4.2.4 Comparative analysis of different NN-based equalizers with fixed computational complexity

The analysis given above does not address the question of which NN topology would provide the best gain if the NN structure's complexity were restricted to a certain level. To answer this question, the equalizers were retested to constrain the total number of real multiplications per recovered symbol (RMpS). It was considered that the complexity values were in the range of 10^3 to 10^8 RMpS. Note that NN structures with large RMpS ($\sim 10^8$) can be prohibitively complex for efficient hardware implementation. However, Ref. [145] demonstrated an efficient FPGA implementation of LSTM NN with 256 and 512 hidden units. This result reveals that the architectures outlined in this research are still feasible for realistic signal processing when advanced techniques for NN hardware implementation are used.

The hyper-parameters distributions for each NN architecture with the complexity constraint are summarized in Table 4.5 in the cells marked from ' 'Topology 1" to "Topology 6". The BO also adjusted the parameters of those topologies; for each case, it was decreased the allowed BO search range to adhere to each constraint on computational complexity.

As seen in Fig. 4.5, for different allowed computational complexity levels, the performance ranking of equalizer types changes. Several conclusions can be drawn by analyzing the results emerging from the simulated (Fig. 4.5a) and experimental (Fig. 4.5b) data. First, in the experimental scenario, the best complexities corresponding to the maximum gain coincide with the complexities identified by the BO procedure, which confirms the effectiveness of the BO in finding the "right" NN architecture. Second, in simulations, the maximum performance is already reached at a lower complexity level compared to the experimental results. As it can be seen from the experimental figure, the CNN+biLSTM, CNN+MLP, and biLSTM equalizers need $\approx 10^7$ RMpS, while in the simulation $\approx 10^6$ RMpS was already enough to achieve the best performance. This observation further confirms that the NN can cope with the reversion of the simulated channel more easily than with the reversion of experimentally

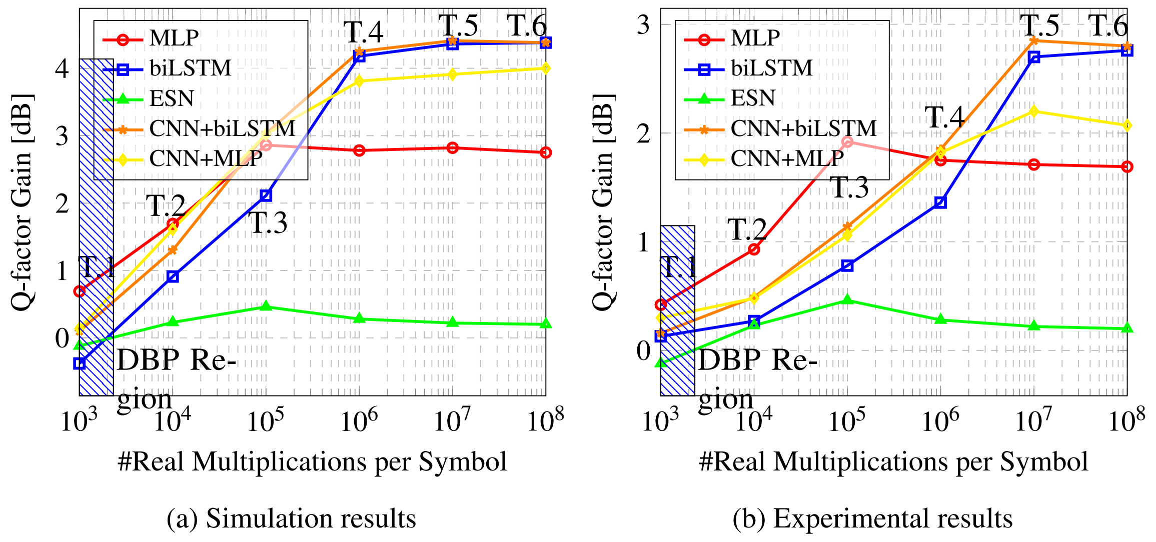


Fig. 4.5 Q-factor gain dependence on the constrained multiplications number for the equalizers having different architectures, presented in Sec. 2.3, in the case of DP-16 QAM single channel TWC-fiber 9×50km. The power level is 2 dBm, which guarantees the high enough nonlinearity transmission regime.

obtained data. Third, the gain stays essentially constant as we raise the complexity above the BO-determined level. This is due to overfitting, and it is particularly pronounced in the MLP scenario. The key concept of the function approximation capability of the MLP belongs to its number of i) feed-forward hidden layers and ii) hidden neurons; these two parameters define the NN's capacity [146]. Changing the MLP's capacity by adjusting the complexity levels frequently leads to unpredictable changes in the NN's performance. Starting at the 10^5 complexity level for both simulation and experimental layouts, one can see that the MLPs with oversized capacity suffer from overfitting, as the network memorizes the properties of the training set in such detail that it can no longer efficiently recover the information from the inference dataset [146]. The latter blocks the equalizer from providing further Q-factor improvement. Thus, the architectures found by the BO identify the most appropriate NN equalizer's capacity (structure) matching this problem, and a further increase in complexity cannot render any noticeable performance improvement.

Next, note that for the high level of RMpS (Topologies 4, 5, and 6), the best-performing equalizer is the CNN+biLSTM. However, once the number of real multiplications from Topology 3 and below is reduced, the best-performing equalizer turns out to be the traditional MLP. This is a result of the fact that sophisticated architectures, like CNN and biLSTM, necessitate more filters and hidden units, respectively, in order to fully understand the dynamics of the data. Also, the CNN+biLSTM performs similarly to the CNN+MLP at low complexity levels (orange and yellow curves in Fig. 4.5), and similarly to the biLSTM (blue line) at high complexity. Consequently, it can be inferred how the addition of a convolutional layer works: while for high complexity the blue and orange curves are approximately the same, at a lower allowed complexity level the CNN+biLSTM performs better.

Also, the hatched blue zone represents the traditional DBP with 3 StPS, in both simulations and experiments to show how well the NN equalizers worked with the same amount of computation as the DBP. Then, it is evident that reducing the number of neurons, filters, and hidden units is not the optimal technique to achieve low-complexity architectures, because the performance fell below the DBP level. As a possible alternative, pruning, and quantization techniques [147, 148] can be used to minimize the computational complexity of the NN equalizers without compromising their performance, making the NN equalizers appealing not only for their good performance but also for their decreased complexity.

Last but not least, the ESN's performance fell short of expectations; this can be seen by its lowest achievable gain numbers. However, [149] contains the results explaining the poor ESN performance for the nonlinear wireless scenario. It was shown that in a channel with a high level of noise, the ESN-based system indeed performs poorly. Furthermore, in that reference, it was demonstrated that by increasing the ESNs' number of neurons (i.e., its complexity), and, thus, effectively increasing the hidden dimensionality of the representation, the equalization performance worsens. Moving to the nonlinear optical channel equalization, both of the aforementioned effects were observed: the performance was relatively poor due to the high level of noise, and the performance did not improve when increasing the complexity, as can be seen from the behavior of the green curve in Fig. 4.5.

4.3 Compression Techniques to Reduce the Computational Complexity of Equalizers

As demonstrated in the previous section, simplifying the structure of the NN equalizer is insufficient to reduce the amount of work required to operate it while maintaining competitive performance. This section will present a collection of compression techniques with the aim of reducing the computational complexity of NN-based equalizers to levels that are suitable for future industrial applications without sacrificing performance significantly.

4.3.1 Multi-symbol Architecture for NN-based Equalizers

As it was seen in the previous section, one of the most promising black box equalizers in terms of performance is the bidirectional LSTM equalizer in a configuration of many-to-one (1D regression task), i.e., when a window of symbols is used to recover just the central one, which has a computational complexity in terms of real multiplications per recovered symbol (RMpS) given by:

$$C_{\text{biLSTM}} = 2n_s\big(\underbrace{4n_h n_i}_{a} + \underbrace{4n_h^2}_{b} + \underbrace{3n_h}_{c} + \underbrace{n_o n_h}_{d}\big), \tag{4.42}$$

where n_s is the size of the input sequence in the time-domain, n_i is the number of input features, n_o is the output dimension (which is equal to 2 - the real and imaginary parts of the symbol), and n_h is the number of hidden units in the LSTM cell. In Eq. (4.42), a is attributed to matrix multiplication of input and weights; b to matrix multiplication of hidden states and weights; c to pointwise multiplications occurring internally within the LSTM cell; and d to matrix multiplication of hidden states and output weights[9]. During the investigation of computational complexity reduction, it was found that simply applying compression techniques would not be enough to reduce the complexity of DBP because such compression strategies reduce the multiplications between input and weights, as in a, b, and d, but do not impact internal multiplications as in c. As a result, the multiplication $n_s n_h$ would become the bottleneck to achieving a reduction in complexity. To mitigate this effect, two different strategies can be used. As suggested in [150], it is possible to utilize basic vanilla recurrent neural networks (RNNs) as the equalizers insofar as they lack an intrinsic point-wise multiplier and, therefore, do not suffer from this issue. The second possibility, again indicated in Ref. [150], is to recover several symbols at a time rather than just the central one, which allows for eliminating some n_s multiplications.

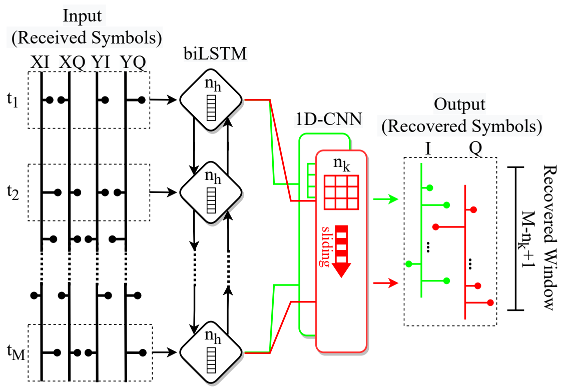


Fig. 4.6 Schematic of the biLSTM+CNN equalizer: the input consists of M real (I) and imaginary (Q) parts of the symbols. The LSTM cells are indicated by lozenges containing n_h hidden units. The LSTM output is sequentially processed by the convolutional layer with two filters to compute the I and Q components of the symbols.

[9]Here, it is considered that a flatten layer was applied to achieve a many-to-one configuration. Instead, if the output comes from just one cell, the complexity of the term d instead of $2n_s n_o n_h$, would read as: $2n_h n_o$

Firstly, it was tested the vanilla RNN and optimizing it using the BO. However, while comparable performance was shown in Ref. [150] when using the LSTM and vanilla RNNs, it was observed quite different performances when using these NNs (with both tuned using the BO). Indeed, using the standard DBP as the reference comparison scenario, as is typically done in the literature, the vanilla RNN barely outperformed the 1 STpS DBP, while the LSTM-based architecture showed better performance than a 3 STpS DBP. In this case, the BO showed substantially low ($\approx 5.10^{-5}$) learning rates in the vanilla RNN scenario, in an attempt to reduce the impact of exploding gradients (that such layers are known to have). Consequently, the training process got stuck in the local minima of the loss function landscape, which limited the optical performance improvement attainable by the equalizer. The LSTM cell, an enriched variant of the vanilla RNN cell with several gating units that help propagate the gradient and govern the flow of information through the NN, solves this gradient problem. However, at the cost of additional complexity.

The better-performing LSTM equalizer is considered henceforth. However, it was enhanced by recovering multiple symbols with the same NN structure instead, as proposed in Ref. [150]. To recover multiple symbols, it is needed to consider that, since chromatic dispersion plays an important role in fiber perturbation, if the NN equalizer processes a window of M symbols as input, it will be able to recover $\mathrm{M}-x$ symbols only, where M is the number of input symbols and x depends on the system memory length. Since the initial and final symbols of the window will lack important information from their neighbours (due to dispersion-induced memory), they may not be recovered properly. The simplest way to reduce the dimensionality of the time window tensor without losing information is by using a 1D convolutional layer. For this purpose, it uses a 1D-CNN with a kernel size n_k, the padding set to zero, the dilation and stride set to 1, and just two filters to represent the real and imaginary parts of each symbol. In this case, the number of recovered symbols (the recovery window) is $\mathrm{M}-n_k+1$. The BO was used to estimate the appropriate values for M, n_h, n_k, learning rate, and mini-batch size, limiting the number of hidden units to at most 150 for complexity constraint reasons. The equalizer scheme is shown in Fig. 4.6.

The computational complexity of the bidirectional LSTM + 1D-CNN equalizer, in terms of RMpS, can be represented using the formulae in Ref. [151], but this time taking into consideration the parallel recovery of n_s-n_k+1 symbols as:

$$\mathrm{RMpS}_{\mathrm{NN}} = \frac{2n_s n_h(4n_i+4n_h+3)}{n_s-n_k+1} + 2n_h n_o n_k, \tag{4.43}$$

The analysis of Eq. (4.43) shows that the number of multiplications has decreased when compared to that of the initial biLSTM equalizer, but compression techniques are still required to further reduce the number of multiplications to a level at least below 1 STpS DBP without affecting the resulting model's performance.

4.3.2 Pruning Approaches for Complexity Reduction

Pruning is the process of removing parameters, neurons, or even layers or parts of a NN that do not significantly impact its performance to reduce its computational complexity. The area of NN pruning is wide and encompasses several subcategories: (a) static or dynamic; (b) one-shot or iterative; (c) structured or unstructured; (d) magnitude-based or information-based; (e) global or layer-wise. Detailed information on the different types of pruning can be found in, e.g., Refs. [106, 152–156]. In this investigation, we prune the lowest magnitude weights globally throughout the NN [106]. This low complexity, traditional unstructured global magnitude pruning has already proven to be quite effective [106, 157–160]. To be more specific, it was considered, in this thesis, a static, iterative, unstructured, global magnitude-based pruning. In this case, it is removed weights offline from the network after training and before inference. Moreover, iterative pruning allows for pruning more weights while preserving accuracy.

The four (most promising) strategies for the iterative-pruning retraining process that are applied in this study are schematically depicted in Fig. 4.7. The four approaches are referred to as fine-tuning, weight rewinding, learning rate rewinding, and Bayesian optimizer-assisted.

Fine-Tuning approach

This method prunes the model once it has been trained. In the second step, it trains the weights that remain after pruning using a constant learning rate, which is usually the same as the final learning rate of the original training procedure. The first panel of Fig. 4.7 shows how the fine-tuning scheme is implemented. After determining the fine-tuning period, the traditional gradual pruning method (a polynomial decay) [161] is used. The pruning polynomial decay approach quickly prunes the network at the beginning, when there are many redundant connections, and gradually reduces the number of weights pruned each time. This procedure results in a smooth loss function during the fine-tuning period, which is beneficial for the learning process to maintain an accuracy close to the original NN model.

This approach can be used when employing the other "deterministic" methods for the equalization of optical fiber nonlinearities. For example, this pruning technique can be used (in a simpler way) to eliminate the less relevant coefficients of the Volterra equalizers [29, 162, 163] and to trim the unimportant triplets (making the triplet feature vector more sparse).

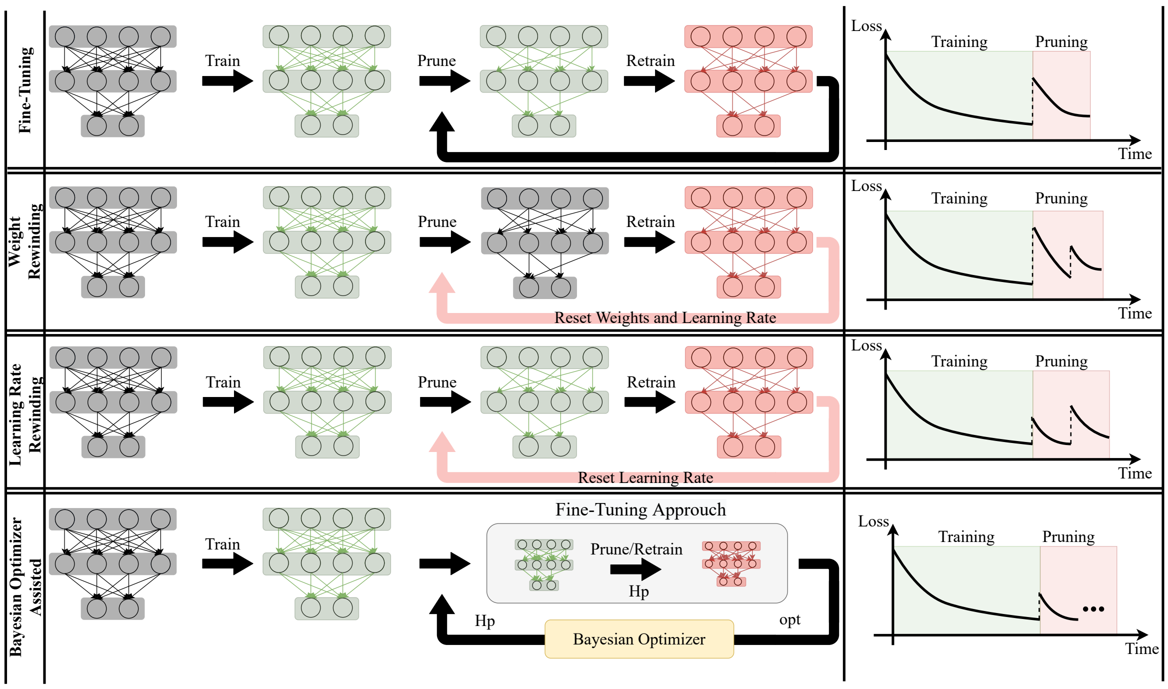


Fig. 4.7 A schematic of the fine-tuning, weight rewinding, learning rate rewinding, and Bayesian optimizer-assisted pruning strategies is shown. A qualitative representation of the evaluation loss over the training time process is shown on the right-hand side. H_p is the set of hyperparameters suggested by the BO for the pruning phase.

It can also be used when employing perturbation approaches [1, 86, 164]. Several works investigated the use of fine-tuning, mainly in short-reach intensity-modulated systems [165–170], to reduce the complexity of the model. So far, the analysis of pruning in optical channel equalization has been restricted to the case of feed-forward NN models only. This investigation will also present such an analysis for a recurrent equalizer and deal with the case of coherent optical transmission.

Weight rewinding approach

This method was introduced in Ref. [157] dealing with the lottery ticket hypothesis. The main idea supporting it is that a dense NN with random initialization contains a subnetwork that, when trained in isolation, can match the test precision of the original network after training. This approach is separated into three parts, as shown in the second line of Fig. 4.7. First, during the initial training process, the weights of each epoch are saved. Then, at the end of the initial training, a percentage of the connections are pruned, and the remaining weights and the learning rate are reset to their prior values (at the k-th epoch of the initial training); the choice of the particular epoch number k, used in this investigation, is explained below. Subsequently, the retraining restarts from the k-th epoch and goes up to the last epoch, followed by a fresh round of pruning using the remaining weights. The cycle of resetting weights and learning rates is repeated until a specific degree of sparsity is achieved. In this approach, the loss function oscillates over the pruning time since the loss increases every time the weights are reset, but the process tends to converge to the reference before pruning.

In the context of channel equalization, the first work that applied this method was the recent work by Koike-Akino et al. [125], where such a technique has been tested in the feedforward model called ResMLP. It was shown that this approach can give a sparsity of 99% compared to the initial overparameterized solution with 6 layers and more than 10^6 parameters. In this investigation, and similarly to Ref. [125], the first epoch was rewound, meaning that $k = 1$.

Learning rate rewinding approach

This method was introduced in [159] and combines fine-tuning with weight rewinding. The third panel of Fig. 4.7 shows how this method operates. While the weight rewinding described above rewinds both the weights and the learning rate, the learning rate rewinding simply rewinds the learning rate, leaving the weights to be re-trained after pruning from their values at the end of the initial training phase (like in the fine-tuning approach described above). In a nutshell, after initial training, a percentage of connections are pruned and re-trained while just the learning rate schedule is rewound. This cycle is repeated until the network sparsity is at the desired level. This approach has not been evaluated previously for the optical equalization task.

Bayesian optimizer assisted approach

The weight and learning weight rewinding approaches were proposed because just fine-tuning the initial hyperparameters of the NN does not guarantee that the performance of the equalizer remains similar. A possible explanation for this effect is that, once the pruning of the NN starts, the optimization problem's target changes. Consequently, the hyperparameters of the new architecture may also need to be adjusted. Indeed, and as stated in Ref. [171], it may lose performance if the hyperparameters are set to a default value when tuning the NN's hyperparameters. Solutions, as in [172] leverage reinforcement learning to provide the model compression policy, determining layer-wise pruning rates. Alternatively, the BO-based approach was used to not only define the pruning policy but also other important hyperparameters of the model (the number of tuning epochs, learning rate, batch size, and initial/final sparsity), thus optimizing the trade-off between performance and computational complexity. We note that this is a completely new approach that has not been tested in any other applications.

Let us briefly specify the BO approach[10] that seeks the global optimum $\mathbf{x}^*$ of a black-box function opt, where $opt(\mathbf{x})$ can be evaluated for any arbitrary $x \in \mathscr{X}$. That is, $x^* = \arg\min_{x \in \mathscr{X}} opt(x)$, where $\mathscr{X}$ is a hyperparameter space that can contain categorical, discrete, and continuous variables [104]. For solving the problem formulated above, the BO assumes that the function opt was sampled from a Gaussian process. The BO maintains a posterior distribution for this function when observations are made [105]. For this application, the observations are the outcomes of performing the NN-based equalization trials with different hyperparameters. To choose the hyperparameters for the next trial, this investigation optimized the expected improvement over the current best result; see more in Ref. [173].

In this study, the optimization process involves the following procedure. After the initial training phase, the NN model with hyperparameters $H_i \in \mathscr{X}$, has a total computational complexity (say, expressed in terms of real multiplications) C_i, and a certain performance P_i (the P_i is evaluated using a testing dataset). Then, we use the BO to minimize the following

[10]The hyperparameter optimization can be done using methods other than the BO, although, as mentioned in Ref. [104, 105], the hyperparameter optimization strategy, the BO offers numerous advantages over search algorithms in finding good candidates with fewer interactions.

objective function:

$$opt = \begin{cases} (P_i - P_p)\dfrac{C_p}{C_i}, & P_i > P_p \\ -\dfrac{C_i}{C_p}, & \text{if } P_p \geq P_i \end{cases}, \tag{4.44}$$

where P_p and C_p are the performance and computational complexity observed when using a set of hyperparameters $H_p \in \mathscr{X}$ in the pruning and fine-tuning process, respectively. The two possible scenarios that may occur when pruning is applied are covered by Eq. (4.44): i) the first one corresponds to the usual case where P_i is better than the pruned performance P_p. In such a situation, the goal of minimizing opt is equivalent to minimizing the $P_i - P_p$ gap and, at the same time, reducing the number of multiplications C_p when compared to the initial ones, C_i. ii) the second case takes place when the pruned NN improves performance, $P_p > P_i$. This case occurs when pruning enables escaping a local minimum, thus improving the NN performance. The focus is then on reducing the computational complexity. According to Eq. (4.44), this means that the reduction of opt can only be achieved by reducing C_p, (since C_i is constant). This procedure is a new approach (even in ML science): it aims at identifying the best balance between the model's performance and computational complexity, by selecting a good candidate for the parameter set H_p.

4.3.3 Weight sharing approach for complexity reduction

Weights clustering, also referred to as the "weight-sharing compression approach," is another method that can be explored to reduce the NN model's complexity by reducing the number of effective weights used by the model. This approach takes into account that several connections may share the same weight value and then fine-tunes those shared weights. In the case of feedforward structures, this strategy was already successfully employed to minimize the complexity of NN models [123, 156, 174, 175]. This section uses the same method as in [156], but modifies it for the recurrent layers as well. Following the selection of a centroids' initialization technique, [156], a minimal distance from each weight to such centroids is used to determine the shared weights for each layer of a trained network so that all weights in the same cluster share the same weight value. The weights are not shared between the layers to prevent further performance loss and because sharing weights between sequential layers does not lower computing complexity. Fig. 4.8 illustrates how this strategy is applied jointly with the BO. To apply weight clustering, it is needed to define three parameters: i) the number of clusters, ii) the centroids' initialization technique, and iii) the weight fine-tuning process. The BO is used to select these parameters so that performance degradation is minimized. The objective function depicted in Eq. (4.44) was also used

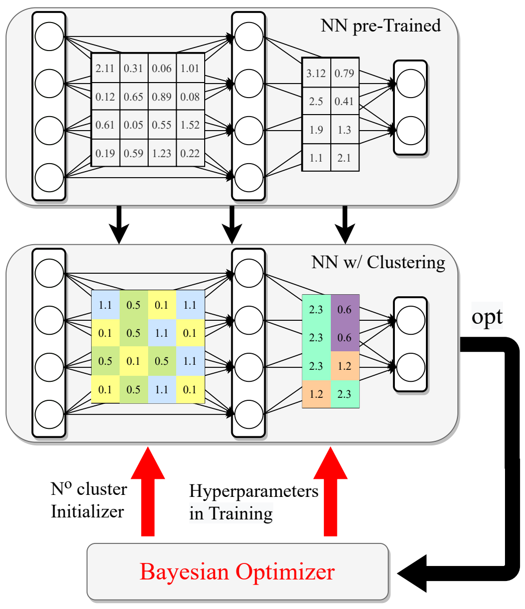


Fig. 4.8 Scheme of weight clustering over dense layers using the BO of its design. Once the NN weights are trained, a selection of weights per layer is forced to be in the closest centroid learned using stochastic gradient descent.

for the BO. Four possible centroid initializers to choose from were provided to the BO: linear-, random-, density-, and K-means-based. Using the weight clustering approach has the advantage of reducing the number of distinct multipliers in matrix multiplication to at least the number of clusters per input element. Then, the results of the multipliers are sent to the different adders. To illustrate the weighted clustering operation, consider the first matrix in Fig. 4.8. Suppose that the input vector I, the output vector O, and the weight matrix W before clustering are linked as follows:

$$O = W \times I = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \begin{bmatrix} i_1 & i_2 & i_3 & i_4 \end{bmatrix}. \tag{4.45}$$

In Fig. 4.8, we cluster this matrix with 3 centroids, c_1, c_2, and c_3, so the new equation connecting input and output, becomes:

$$\begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \begin{bmatrix} c_1 & c_3 & c_2 & c_1 \\ c_2 & c_3 & c_1 & c_2 \\ c_3 & c_2 & c_3 & c_1 \\ c_2 & c_3 & c_1 & c_2 \end{bmatrix} \begin{bmatrix} i_1 & i_2 & i_3 & i_4 \end{bmatrix}. \tag{4.46}$$

This result shows that, in the worst case scenario, the new number of multiplications would decrease from 16 (input size $\times$ output size = 4$\times$4) to (input size $\times$ number of clusters = 4$\times$3), because in this case we can carry out all possible unique multiplications (i_1c_1, i_1c_2, ... ,i_4c_3), and the rest of the operations are additions. However, by properly designing such a matrix multiplication, the number of multiplications can even be further reduced. In the same example, the output O can be defined as follows:

$$\begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \begin{bmatrix} i_1c_1 + (i_2+i_4)c_2 + i_3c_3 \\ (i_1+i_2+i_4)c_3 + i_3c_2 \\ i_1c_2 + (i_2+i_4)c_1 + i_3c_3 \\ (i_1+i_3)c_1 + (i_2+i_4)c_2 \end{bmatrix}. \tag{4.47}$$

Notice that the number of multiplications is reduced to 8 unique multiplications in Eg. (4.47), which is half of its original value (the number of additions remains the same). It is important to note that the benefit resulting from using this technique depends on the lengths of the input vector and the weight matrix, as well as on how the learned weight pattern is spread over the weight matrix. In addition, weight clustering is also used as a form of heterogeneous quantization. In this sense, when assuming a quantization of, for instance, 3 bits, the weight clustering approach will try to identify the 8 unique weights that can best describe the original weight distribution of the NN model. In this case, this type of nonuniform quantization is implemented by maintaining a codebook structure that stores the shared weights, and the weights are grouped by index after calculating the gradient of each layer [156, 176]. Importantly, the weight clustering contributes the most to the NN-based equalizer complexity reduction. Moreover, note that clustering has never been used in the NN-based optical channel equalization.

Finally, it is worth clarifying how the learning process occurs, when backpropagation is used to update the clusters of centroids and the original weights. The TensorFlow implementation used in this section works with a lookup table to hold the centroid values during the model training, as described in Ref. [177]. The weights array is populated with a "gather" operation so that, during the backpropagation, the gradients can be calculated in the usual way. The lookup table is then adjusted using the cumulative gradient values for the weights that correspond to the same centroid. The original weights are also updated by using a straight-through estimator to overwrite the non-differential structure of clustering with an identity function, which allows all upstream gradients to be used in the updated original non-clustered weights of the layer [177, 178].

4.3.4 Different Quantization Techniques for Complexity Reduction

Quantization is used to lower the bitwidth of the numbers participating in arithmetic operations along the signal processing path, which typically helps to significantly reduce the computation complexity of the processing. This means that a quantized model can use, for example, integers, instead of floating-point numbers for some or all operations. Therefore, quantization allows the model to use less memory while doing high-performance vectorized operations on a variety of hardware platforms [179].

Quantization has demonstrated excellent and consistent results when used during training and inference using different NN models [147, 152, 179, 180]. Particularly, it is effective during inference because it saves computing resources without significantly decreasing accuracy. NNs benefit from quantization because NNs are remarkably robust to aggressive quantization and extreme discretization. This robustness emerges from the large number of parameters involved in the NN, meaning that they are typically working with over-parameterized models. This subsection presents the categories of quantization addressed in this study in terms of their mode (post-training quantization [181] or quantization-aware training [182]) and quantization approach (homogeneous [183] or heterogeneous [184]).

Homogeneous or Heterogeneous Quantization

Homogeneous is the most common quantization approach. The homogeneous quantization consists of reducing the precision of all NN weights to the same number of bits. In this case, we use the same type of quantization and number of bits across the entire NN model. However, because the layered structure of multilayered NN models offers high quantization flexibility, it is natural to assume that different layers may impact the loss function differently, which favors a mixed-precision quantization approach. The process of quantizing the layers differently across the NN is known as heterogeneous quantization, and it can be a critical step toward improving the complexity-performance trade-off. In this case, distinct layers with varying bitwidth are quantized into their fixed-point representation, as in Ref. [184].

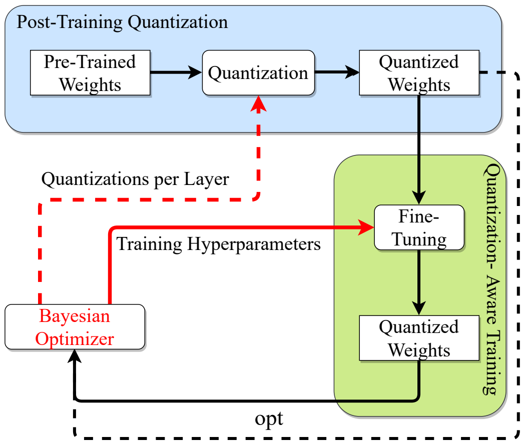


Fig. 4.9 Schematic of the NN quantizer. BO can help with the post-training quantization by finding the best bitwidth precision per layer, and with the quantization-aware training by finding the best hyperparameters for the fine-tuning training after quantization.

There are several types of quantization that may be used. This work focuses on the uniform quantization [185], the power of two quantization (PoT) [186], and the additive power of two quantization (APoT) [124] since they have a wide range of applications and usually deliver quite good results. Regarding those types of quantization, they usually convert the floating-point representation to a fixed-point representation, thus using integer mathematics instead of a floating-point one. This approach reduces both the memory and computing requirements for the realization of a particular solution.

Uniform quantization is the most common and simplest quantization approach. The uniform quantization applied to the NN elements can be expressed as [185]:

$$Q(x)_{BW} = R(x, \alpha L_{uni}(BW)), \tag{4.48}$$

$$L_{uni}(BW) = \left[-1, 0, \underbrace{\pm\frac{1}{2^{BW-1}}, \ldots, \pm\frac{2^{BW-1}-1}{2^{BW-1}}}_{2^{BW-1}-1 \text{ amplitudes}} \right], \tag{4.49}$$

where $Q(\ldots)_{\ldots}$ is the quantization operator, x is a real-valued input (it can be weights or an activation function), BW is the quantized bitwidth value, $R(x, \alpha L_{uni})$ is the function that rounds x to the nearest element on the list L_{uni} that contains all quantization levels, and α is a scaling level that guarantees that the largest weight in the NN will not be clipped. The quantization error is introduced by the rounding functions, depending on the BW precision. Note that, in this section, we use a representation format that, besides the “-1” and “0” values involves $2^{BW-1}-1$ additional amplitudes, defining a total of $2^{BW}-2$ positive or negative levels. The int8 quantization ($BW=8$) is one of the most widely used uniform quantization schemes, not only for the ML frameworks such as TensorFlow and PyTorch, but also for the hardware toolchains such as NVIDIA TensorRT [187] and Xilinx DNNDK [188]. The int8 quantization has the advantage of typically not leading to relevant performance degradation (as can be observed from the results later in this section– see Fig. 4.17). This work will not restrict itself to int8 quantization only but, in contrast, will also use the BO to determine the best number of bits for the quantization process.

The PoT quantization is a logarithmic quantizer [189] designed to approximate the weights to the closest power of two in the range defined by the considered number of bits. The PoT quantization with $2BW$ elements can be represented mathematically as follows:

$$Q(x)_{BW} = R(x, \alpha L_{Pot}(BW)), \tag{4.50}$$

$$L_{Pot}(BW) = \left[-1, 0, \underbrace{\pm\frac{1}{2}, \ldots, \pm\frac{1}{2^{(2^{BW-1}-1)}}}_{2^{BW-1}-1 \text{ amplitudes}} \right]. \tag{4.51}$$

The POT quantization leads to much smaller computational complexity when compared to the uniform quantization because all multiplications can be represented in terms of bit-shift operations (since it uses power-of-two values only). However, as pointed out in several works [124, 186, 189, 190], the performance of the PoT-quantized system can degrade compared to the uniform quantized one due to this scheme's rigid resolution problem.

The APoT quantization was recently proposed to encompass the benefits of PoT and uniform quantization types. As stated in the original work in Ref. [124], the PoT and uniform quantization are special cases of APoT with specific design parameters. The goal of APoT is to have fewer shift-adds than the uniform quantization but, at the same time, to take advantage of its non-uniform quantization levels as the PoT does. Mathematically, the APoT quantization [124] considering 2^{BW} levels can be described as:

$$Q_{BW}(x) = R\big(x, \alpha L_{\text{APot}}(BW)\big), \tag{4.52}$$

$$L_{APot}(BW) = \left[-1, \left\{ \sum_{i=0}^{n-1} p_i \right\} \right], \tag{4.53}$$

$$p_i \in \left[\underbrace{0, \pm\frac{1}{2^{i+1}}, \pm\frac{1}{2^{n+i+1}}, \ldots, \pm\frac{1}{2^{[(2^K-2)*n+i+1]}}}_{2^K-1 \text{ amplitudes}} \right], \tag{4.54}$$

where n is the number of additive terms, k is defined as $k = (BW-1)/n$, and $\{\ldots\}$ is the set containing all possible combinations of n additions from the 2^K different elements in the list p_i[11]. In this description, by setting $n = 1$ it is equivalent to having the PoT case, whereas setting $k = 1$ leads to the uniform case[12]. This work has considered n fixed and equal to 2, 3, or 4. It has also addressed the case from the original APOT paper in Ref. [124], where k was fixed equal to 2. Importantly, the drawbacks of using the APoT quantization with $k = 2$ when the model has already been pruned will also be presented.

Post-Training Quantization

The post-training quantization (PTQ) [179, 181, 185, 191] is a conversion technique in which all trained weights and activations of the NN model are converted to some fixed point representation, following some quantization precision established after the training phase. As indicated in Fig. 4.9, blue box, a quantization approach is applied after training the neural network weights, and the quantized weights are saved for future use. As a result, the PTQ is an extremely fast method of quantizing NN models. Moreover, when using the PTQ, a quick grid search was already enough to analyze all possibilities and get a satisfactory result. Thus, here, it was not used the BO to determine the optimal precision (bitwidth per layer). However, this approach usually leads to a small degradation of the model's performance, independent of the quantization approach selected.

Quantization Aware Training

As stated previously, the inference performance of the quantized integer models is generally worse than that of the floating-point models due to the information loss induced by quantization. To address this limitation, a method known as quantization-aware training (QAT) [122, 179, 192] was proposed. QAT accounts for the loss of information during the training phase, resulting in a smaller performance degradation during inference. This work uses the QAT approach proposed in Ref. [193], where the quantized weight levels are

[11]To explain the notations better, consider the case where $n = 2$ and $k = 3$, such that we have two sets, $p_0 = \{p_0^1, p_0^2, p_0^3\}$, and $p_1 = \{p_1^1, p_1^2, p_1^3\}$. Then, $\{\sum_{i=0}^{1} p_i\} = \{p_0^1 + p_1^1, p_0^1 + p_1^2, \ p_0^1 + p_1^3, p_0^2 + p_1^1, \ldots, p_0^3 + p_1^3\}$.

[12]Eqs. (4.49), (4.51), (4.53) are valid for $b_w > 1$; when $b_w = 1$, we have the same set of values, 1 or 0, independently of the type of quantization.

optimized. Afterward, the quantization is reversed, but the final forward-propagated values also include the errors aggregated by the weight quantization scheme.

The implementation of the QAT is illustrated in the green box in Fig. 4.9. In the QAT case, the fine-tuning block operates as follows: 1) It receives the weights quantized via the chosen quantization strategy; 2) Then, it performs the forward propagation; 3) Afterward, it converts all variables to float precision; 4) Finally, it does the backpropagation. This cycle is repeated until the weights are definitively quantized. The inference about the quantized structure's performance is then completed.

For practical reasons, the QAT scheme for learning depicted in Fig. 4.10 is similar to the one used with weight clustering. In the case of weight clustering, the quantizer box is an identity since the cluster centroids are the alphabet that the NN is training to learn (a nonuniform quantizer), and both centroids and weights are updated in the training process. Instead, the centroids were forced to be fixed into a defined alphabet (e.g., uniform, Eq. (4.49); POT, Eq. (4.51); APOT, Eq. (4.53), and update the weights only (the centroids will fall into one of the possibilities in the quantization alphabet). For the forward propagation, all weights in the NN, W, are quantized to the nearest element of the quantized alphabet c_q, resulting in the quantized weights W_{qc} that will be used to compute the loss function. However, in the backpropagation stage, the gradients were computed using the floating-point values (W). This is possible because the backpropagation engine is forced to "ignore" the quantization step used in the forward propagation. The latter is done by assuming that $\frac{\partial W_{qc}}{\partial W} = 1$. As stated in [6], this process is known as a Straight-Through Estimator, and it results in a smoother transition between consecutive quantization levels in the learning process.

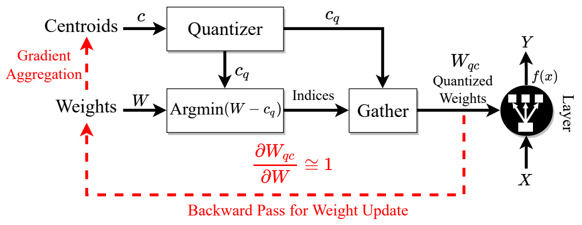


Fig. 4.10 Quantization-aware training scheme for forward and backpropagation. The forward propagation uses the quantized alphabet c_q to generate the quantized weights W_{qc} and, in the backpropagation, such a weight is "skipped" by imposing its gradient to be one (straight-through estimator, Ref. [6]).

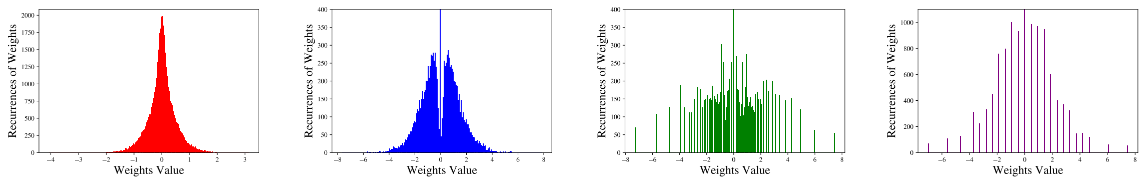
Finally, BO was used to fine-tune, i.e., find the best hyperparameters, in the QAT. Note that, differently from the other two compression approaches, where the computational complexity C is measured in terms of the number of real multiplications, it is now measured in terms of the number of bit operations.

Quantization Applications in Optical Channel Equalization: the Current State of the Art

Several quantization strategies have already been proposed to equalize optical channels. Regarding the post-training quantization, the authors of Ref. [194] implemented an MLP-based equalizer with two hidden layers in an FPGA (XCZU9EG FFVC900) using post-training quantization with traditional uniform int8 precision; the quantized equalizer was tested in an experimental setup of a 50Gb/s PON with a 30 km SSMF link. Next, this time using a recurrent NN-based equalizer, Ref. [195] tested the equalizer in a PAM4-based 100-Gbps PON signal transmission over a 20 km SSMF fiber testbed and applied post-training quantization, changing the bitwidth of the weights from 8 to 2 bits, to evaluate the BER degradation resulting from the quantization. Also, the authors of Ref. [195] implemented such an equalizer in an FPGA using the Xilinx Vivado toolset for high synthesis. The authors of Ref. [196] focused on coherent transmission. In this case, a complex-valued dimension-reduced triplet input neural network was proposed and experimentally tested with a 16-QAM 80 Gbps single polarization signal transmitted along 1800 km of SSMF (100 km SSMF loop). In this study, to validate the robustness of such a NN equalizer on quantization, they reduced the bit precision of weights to up to 2 bits, observing mostly minor performance degradation. Finally, in [197], an MLP equalizer was used to mitigate the impairments in a 30 GBd 1000 km system. In this case, the PTQ strategy together with the traditional uniform 8-bit quantization was demonstrated using low-performing hardware (Raspberry Pi and Jetson nano).

Regarding the QAT strategy description, an important discussion on the quantization of NN weights was held in Ref. [198] where it was emphasized that the equalizer inference should be performed by a fixed-point system to address a more hardware-friendly situation. An MLP-based equalizer was used, and its weights were quantized with a PoT quantization strategy. The authors incorporated the quantization error in the training of the equalizer by using the Learning-Compression (LC) algorithm, which is a possible QAT strategy. The authors of Ref. [199] used a deep CNN equalizer to assess their proposed quantization strategy, which combines QAT and post-equalization to find the most appropriate number of bits for uniform quantization. Using only 5-bit weights, the CNN equalizer was able to achieve performance that was similar to that of the full-precision model for a theoretical dispersive channel with AWGN noise and ISI. More recently, Ref. [125] demonstrated that the APoT strategy could provide much higher resilience to quantization than the ordinary PoT. In this case, a ResMLP equalizer was tested (using simulation results) considering the transmission of a dual-polarization 64/256QAM, 34 GBd 11Ch-WDM signal over 22 spans of 80 km of SSMF.

Finally, for better visualization of what was discussed so far in this section, Fig. 4.11 illustrates the weight distribution of the recurrent kernel in the LSTM layer of the NN equalizer from the different phases of the NN design in this section: (a) Original trained NN; (b) The pruning phase; (c) The weight clustering phase; (d) The quantization phase.



(a) Original. (b) After Pruning. (c) After clustering. (d) After quantization.

Fig. 4.11 Illustration of the weight distribution of the recurrent kernel in the LSTM layer of the NN equalizer from the different phases of the NN design in this section. (a) Original trained NN; (b) The pruning phase; (c) The weight clustering phase; (d) The quantization phase.

4.3.5 Simulation and experimental setups

The performance of the NN-based equalizers with reduced complexity is assessed using data not only from numerical simulations but also from a real experimental setup to make the analysis as complete as possible. The setup used in this experiment is depicted in Fig. 4.12. At the transmitter side, a dual-polarized probabilistic shaped 64QAM (8bits/4D symbol) 34.4 GBd symbol sequence was mapped out of data bits generated by a Marsenner twister generator [200]. Then, a digital root-raised cosine (RRC) filter with a roll-off factor of 0.1 was applied to limit the channel bandwidth to 37.5 GHz. The resulting filtered digital samples were resampled and uploaded to a digital-to-analog converter (DAC) operating at 88 GSamples/s. The DAC outputs were amplified by a four-channel electrical amplifier that drove a dual-polarization in-phase/quadrature Mach–Zehnder modulator, modulating the continuous waveform carrier produced by an external cavity laser at $\lambda = 1.55\,\mu m$. The resulting optical signal was transmitted along 9×110 km spans of SSMF with lumped (EDFA) amplification. The EDFA noise figure was in the 4.5 to 5 dB range. The SSMF is characterized at $\lambda = 1.55\,\mu m$ by an attenuation coefficient $\alpha = 0.21$ dB/km, a dispersion coefficient $D = 16.8$ ps/(nm·km), and an effective nonlinear coefficient $\gamma = 1.14$ (W· km)$^{-1}$.

At the Rx side, the optical signal was converted to the electrical domain using an integrated coherent RX. The resulting signal was sampled at 50 Gsamples/s with a digital sampling oscilloscope and processed by an offline DSP based on the algorithms described in [113]. First, the bulk accumulated dispersion was compensated using a frequency domain equalizer,

Table 4.6 The best hyperparameters were found by the BO for the two transmission setups considered in this work. The same NN configuration is employed for the numerical and experimental setups.

Transmission Case	Input Window	Output Window	Hidden Units	Kernel Size	Mini-Batch size	Learning Rate
20×50km SSMF link 64QAM 30GBd	221	171	100	51	3502	0.001
9×110km SSMF link 64QAM 34GBd	221	195	117	27	2153	0.0013

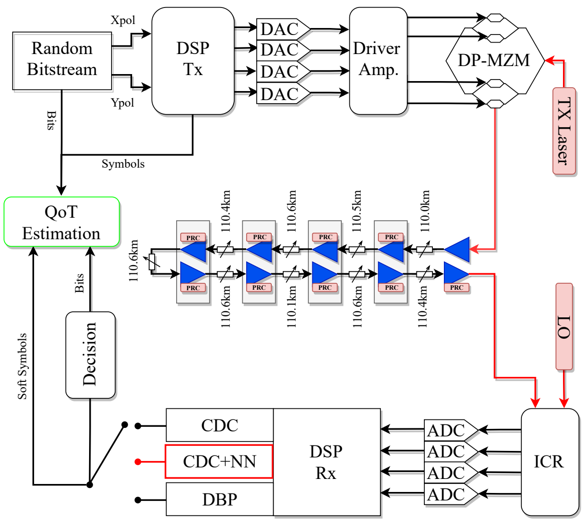


Fig. 4.12 Experimental setup. The input of the NN (shown as the red rectangle after DSP RX) is the soft output of the regular DSP before the decision unit.

which was followed by the mitigation of the carrier frequency offset. A constant-amplitude zero autocorrelation-based training sequence was then located in the received frame and the equalizer transfer function was estimated from it. Afterward, the two polarizations were demultiplexed and the signal was corrected for clock frequency and phase offsets. The carrier phase estimation was then done with the help of pilot symbols. Subsequently, the resulting soft symbols were used as input for the NN equalizer. Finally, the pre-FEC BER was evaluated from the signal at the NN output.

The experimental transmission setup was mimicked by simulation. In this case, the transmission of a DP-64QAM, single-channel (SC) 34.4 GBd signal pre-shaped by an RRC filter with 0.1 roll-off, with an upsampling rate of 8 samples per symbol (275.2 GSamples/s) over the same fiber link is assumed. An additional simulated setup was also tested, consisting of the transmission of a DP-64QAM signal (but with a symbol rate of 30 GBd) along 20×50 km SSMF spans. The propagation of the optical signal along the optical fiber was simulated by solving the Manakov equations (3.1) using the split-step Fourier method (with a resolution of 1 km per step). Each fiber span was followed by an EDFA with the noise figure $\mathrm{NF} = 4.5$ dB, which fully compensates for fiber losses and adds amplified spontaneous emission noise. At the RX, after the full electronic chromatic dispersion compensation (CDC) by the frequency-domain equalizer and downsampling of the signal to the symbol rate, the received symbols are normalized to the transmitted ones. The performance of the system was evaluated in terms of the Q-factor, defined as: $Q = 20\log_{10}\left[\sqrt{2}\,\mathrm{erfc}^{-1}(2\,\mathrm{BER})\right]$.

Focusing now on the biLSTM + CNN NN implemented in this work, the mean square error (MSE) loss estimator and the classical Adam algorithm for the stochastic optimization step [201] were used when training the weights and bias of the NN. The training hyperparameters (mini-batch size and learning rate) and the NN design hyperparameters (output window, hidden units of the LSTM, and kernel size of the 1D-CNN) were found using the BO procedure described in [116]. An input window with 221 symbols was selected because it allows equalizing many symbols simultaneously, thus reducing the computational complexity. The BO optimization cycle starts with the training of the NN via backpropagation for 1000 epochs with a fixed set of hyperparameters. The BER is evaluated after each training epoch. For training, it was used a fixed dataset with 2^{20} symbols, and, at every epoch, it was picked 2^{18} random input symbols from this dataset. For testing, a never-seen-before dataset with 2^{18} symbols was used. Both NN models were trained and tested using the same datasets. The best BER estimated during training was stored. Following the training phase, the best BER was fed to the BO as an optimization target [116] (the optimizer assumes a Gaussian conditional distribution of BERs). Using this input, the optimizer updates the process model and generates a new set of hyperparameters to be tested. After 20 Bayesian optimizer cycles, it is selected the set of hyperparameters leading to the lowest BER. The BO grid space considered was: mini-batch size [32 to 5000], learning rate [0.0001 to 0.002], hidden units [1 to 150], and kernel size [1 to 200]. In this case, the output window is directly defined by the input window and the kernel size of the 1D-CNN. The BO was used to learn the best hyperparameters for the two different transmission setups considered in this study. The results of the optimization process are summarized in Table 4.6.

4.3.6 Results and discussions

Complexity Metric for Compression

It was presented in Sec.4.1 a general way of estimating the computational complexity for different-type of NN layers. The proposed metrics were the number of multiplications [117, 118], the number of bit operations [121–123], the number of shift and add operations [202] and the number of hardware logic gates [203, 204]. A brief description of each of these metrics is presented in the Table. 4.3. This subsection focuses on the expressions of computational complexity when combining the biLSTM and 1D-CNN layers, and on how the compression techniques impact the computational complexity of NN equalizers.

Traditionally, the simplest estimation of complexity refers to the number of real multiplications of the algorithm only. This metric is also known as the number of real multiplications per recovered symbol (RMpS) [151]. The RMpS corresponding to the bidirectional LSTM + CNN equalizer is given by Eq. (4.43). Since this work considers unstructured pruning, which prunes all layers in the same way, the number of multiplications is:

$$\begin{aligned}\mathrm{RMpS}_{\mathrm{NN}} = \left(\frac{2n_s n_h(4n_i + 4n_h)}{n_s - n_k + 1} + 2n_h n_o n_k\right)(1-\mu) \\ + \frac{6n_s n_h}{n_s - n_k + 1},\end{aligned} \tag{4.55}$$

when assuming that the achieved sparsity level is equal to μ. The explanation for the variables entering Eq. (4.55) can be found below Eq. (4.42). Note that the pruning coefficient reflects the multiplications' reduction only for the weight multiplications. Since the recurrent layers are the focus, the number of pointwise multiplications that occur internally in the recurrent cell is not affected by pruning. Aside from pruning, weight clustering can also be used to reduce the RMpS. As indicated in subsection 2.3, and also taking into account the equations that describe the LSTM cell and the 1D-CNN layer in Ref. [151], each LSTM cell depends on four input kernel matrices, $W^{i,f,o,c}$, and four recurrent kernel matrices, $U^{i,f,o,c}$. These four matrices for input and recurrent kernel are usually treated as two matrices (say, W and U) with shapes $[n_i, 4n_h]$ and $[n_h, 4n_h]$, respectively. Now, consider that in the input matrix, W, we can identify some number of clusters, c_i, and for the recurrent kernel matrix, U, we can identify c'_h clusters. Therefore, the number of unique real multipliers would be $n_i * c_i$ for the multiplications involving the matrix W, and $n_h * c'_h$ for the multiplication with the matrix U[13]. Then, the contribution of unique multiplications to the overall complexity of the LSTM, is:

$$\mathrm{C}_{\mathrm{LSTM}} = n_s\left(n_i c_i + n_h c'_h + 3n_h\right), \tag{4.56}$$

For the complexity analysis, it is also needed to include the contribution of the 1D-CNN layer. Since the 1D-CNN layer receives the output of the biLSTM layer, the 1D-CNN layer with kernel size n_k and the number of filters n_o will possess a CNN kernel tensor with the shape [n_k, $2n_h$, n_o]. Suppose that we have identified c''_j clusters in each of n_o the filter and the biLSTM the output has the shape [n_s, $2n_h$]. Now we can split the operation for the

[13]Note that, once those multiplications are performed, a synthesis data-flow / routing algorithm [205] would be needed to distribute the result of such a multiplier to the correct adders, but here it was not considered for the complexity of this design step.

convolution of the biLSTM output with a CNN filter as c''_j multiplications between each of the clustered kernel values to the sum of the selected input elements which share the same clustered weight value. This operation has to be repeated $n_s - n_k + 1$ times (the output shape of the CNN layer) for one filter, and then for the total number of operations we multiply this value by the number of filters n_o. The remaining operations are just additions. Therefore, the 1D-CNN layer complexity contribution can be expressed as:

$$\mathrm{C}_{\mathrm{CNN}} = (n_s - n_k + 1)\left(n_o c''_j\right). \tag{4.57}$$

And now, the ultimate complexity for the biLSTM + CNN equalizer with clustering and pruning, becomes:

$$\mathrm{RMpS}_{\mathrm{NNp+c}} = \frac{\mathrm{C}^{forward}_{\mathrm{LSTM}} + \mathrm{C}^{backward}_{\mathrm{LSTM}} + \mathrm{C}_{\mathrm{CNN}}}{n_s - n_k + 1}, \tag{4.58}$$

Note that the pruning simplifies the clustering method since it has to group fewer weights. Also, when defining the number of clusters in each layer (c_i, c'_h, and c''_j), one of the clustered values is almost always zero. Consequently, this cluster does not add multiplications, and it has even lower computation complexity.

When comparing solutions that use floating-point arithmetic with the same bitwidth precision, the RMpS is usually a meaningful metric for comparative estimates. When moving to fixed-point arithmetic, a second metric known as the number of bit-operations (BOPs) should be adopted to understand the impact of changing the bitwidth precision on the complexity[14]. Because one can readily find the number of bit operations required by the additions and multiplications, it is possible to calculate the BOPs associated with the NN inference process, expressed in terms of multiply-and-accumulate operations (MACs)[121–123, 202]. As described in Ref. [202], the BOP complexity for the LSTM and 1D-CNN layers can be expressed as:

$$\begin{aligned}\mathrm{BOP}_{\mathrm{LSTM}} = {} & 4 n_s n_h \mathrm{Mult}(n_i, b_w, b_i) \\ & + 4 n_s n_h \mathrm{Mult}(n_h, b_w, b_a) \\ & + 3 n_s n_h b_a^2 \\ & + 9 n_s n_h \mathrm{Acc}(n_h, b_w, b_a),\end{aligned} \tag{4.59}$$

[14]Two assumptions are made in the definition of the BoPs. First, it was assumed that each parameter is only fetched once from an external memory; second, the cost of fetching a b-bit parameter is assumed to be equal to b-BOPs [121, 122]. Also, the bias is supposed to be quantized in the same way as the weights.

$$\begin{aligned}\text{BOP}_{\text{CNN}} = & \, \mathit{OutputSize} \cdot n_f \text{Mult}(n_i n_k, b_w, b_i) \\ & + n_f \text{Acc}(n_i n_k, b_w, b_i),\end{aligned} \tag{4.60}$$

where, in the context of NNs, b_w is the number of bits used to represent the weights of the NN, b_i is the number of bits used to represent the input, and b_a is the number of bits used to represent the NN's activation functions. For the convenience of further presentation, we have used short notations, Mult and Acc:

$$\text{Mult}(n_i, b_w, b_i) = n_i b_w b_i + (n_i - 1)\big(b_w + b_i + \lceil \log_2(n_i) \rceil\big),$$

and

$$\text{Acc}(n_i, b_w, b_i) = b_w + b_i + \lceil \log_2(n_i) \rceil.$$

The Acc expression represents the actual bitwidth of the accumulator[15] required for MAC operations. Therefore, for the proposed NN-based equalizer (biLSTM+1D-CNN) with 4 input features, 2 output features, n_h hidden units in the LSTM cell, n_k convolutional kernel size, and $n_s = M$ memory time window, the required number of BoPs considering that the output of biLSTM is the input of 1D-CNN can be represented, as follows:

$$\begin{aligned}\text{BOP}_{\text{LSTM}}^{for/backward} = & \, 4Mn_h \text{Mult}(4, b_w, b_i) \\ & + 4Mn_h \text{Mult}(n_h, b_w, b_a) \\ & + 3Mn_h b_a^2 \\ & + 9Mn_h \text{Acc}(n_h, b_w, b_a),\end{aligned} \tag{4.61}$$

$$\begin{aligned}\text{BOP}_{\text{CNN}} = & \, (M - n_k + 1) \cdot 2\text{Mult}(2n_h n_k, b_w, b_i) \\ & + 2\text{Acc}(2n_h n_k, b_w, b_i),\end{aligned} \tag{4.62}$$

$$\text{BoP}_{\text{NN}} = \frac{\text{BoP}_{\text{LSTM}}^{forward} + \text{BoP}_{\text{LSTM}}^{backward} + \text{BoP}_{\text{CNN}}}{M - n_k + 1}. \tag{4.63}$$

Most real DSP implementations use dedicated logic macros (e.g., DSP Slice in FPGAs or MAC in ASIC), where the BoP metric fits as a good complexity estimation/comparison metric. However, with the advances in new quantization techniques [124–127], those multiplications

[15]The accumulator is the register in which the intermediate arithmetic logic unit results are stored. For a more detailed explanation, see [202].

can also be implemented using just bit shifter- and adder-based algorithms [128–130], when the fixed-point multiplications are used[16]. Therefore, to account for the impact of using different quantization strategies, it is possible to utilize the metric that evaluates the number of additions and bit shift (NABS) operations.

Next, it is discussed how to translate the complexity of the NN equalizer from RMpS into the NABS metric, in the cases when utilizing uniform quantization, PoT quantization, and APoT quantization (see also Section 4.3). According to Ref. [202], the NABSs metric takes into account the conversion of all multipliers into adders and shifters, and computes the complexity of the total number of adders (including the pre-existing adders of the NN structure) based on bit precision while ignoring the cost of the bit shift operations. The NABSs complexity for the LSTM and 1D-CNN layers can be expressed as [202]:

$$\begin{aligned} \mathrm{NABS}_{\mathrm{LSTM}} = {} & 4n_s n_h \big[n_i(X_w+1)-1\big]\mathrm{Acc}(n_i, b_w, b_i) \\ & +4n_s n_h \big[n_h(X_w+1)+1\big]\mathrm{Acc}(n_h, b_w, b_a) \\ & +6n_s n_h b_a. \end{aligned} \tag{4.64}$$

$$\begin{aligned} \mathrm{NABS}_{\mathrm{CNN}} = {} & OutputSize \cdot n_f \big[n_i n_k(X_w+1)-1\big] \\ & \cdot \mathrm{Acc}(n_i n_k, b_w, b_i) \\ & +n_f \mathrm{Acc}(n_i n_k, b_w, b_i). \end{aligned} \tag{4.65}$$

In these expressions, X represents the number of adders required, at most, to perform the multiplication when considering that the first bit represents the sign and the remaining ones contain the magnitude of the weight. For the uniform quantization: $X = b_w - 2$, whereas in the case of POT quantization: $X = 0$, because each multiplication costs only a shift [6, 128]. Lastly, for the APOT quantization: $X = \mathrm{n}$, where n denotes the number of additive terms. These equations are in line with the expected complexity behavior from Ref. [124], where it is stated that by using the APOT with $k = 2$ (which means that $\mathrm{n} = (b_w - 2)/2$), the multiplication would be approximately 2 times faster than when using the uniform quantization. Thus, for the biLSTM + 1D-CNN equalizer considered in this work, the

[16]Note that the translation from multiplications to additions and shift operations adds some quantization noise/error since rounding the original coefficients when converting them from a float representation to a fixed representation. However, in the context of NNs, this can be partially mitigated by training the NN with the quantized weights, as it was done in Refs. [125–127] and in this current work.

following expressions for the NABSs complexity per recovered symbol can be drawn:

$$\begin{aligned}\mathrm{NABS}_{\mathrm{LSTM}}^{\mathrm{for/backward}} = & 4Mn_h\left[4(X_w+1)-1\right]\mathrm{Acc}(4,b_w,b_i) \\ & +4Mn_h\left[n_h(X_w+1)+1\right]\mathrm{Acc}(n_h,b_w,b_a) \\ & +6Mn_hb_a,\end{aligned} \tag{4.66}$$

$$\begin{aligned}\mathrm{NABS}_{\mathrm{CNN}} = & (M-n_k+1)\cdot 2\left[2n_hn_k(X_w+1)-1\right] \\ & \cdot\mathrm{Acc}(2n_hn_k,b_w,b_i) \\ & +2\mathrm{Acc}(2n_hn_k,b_w,b_i),\end{aligned} \tag{4.67}$$

$$\mathrm{NABS}_{\mathrm{NN}} = \frac{\mathrm{NABS}_{\mathrm{LSTM}}^{forward}+\mathrm{NABS}_{\mathrm{LSTM}}^{backward}+\mathrm{NABS}_{\mathrm{CNN}}}{M-n_k+1}. \tag{4.68}$$

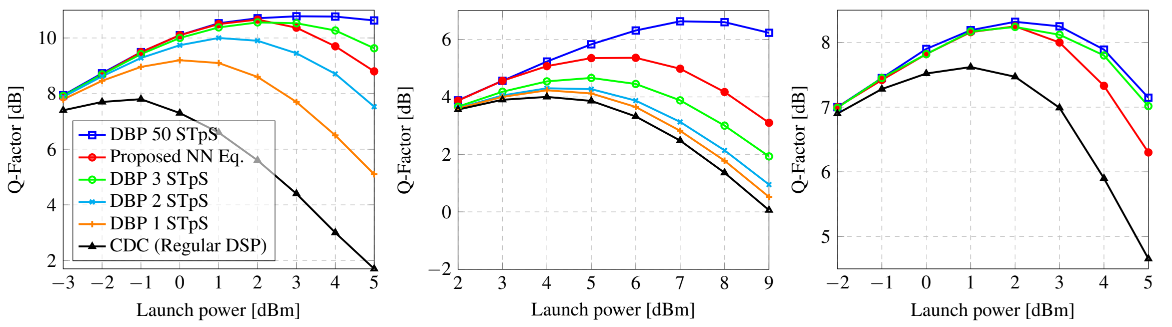
Notably, the BoPs and NABSs expressions given above do not take into account the effects of pruning and weight clustering, but they can be corrected, similarly to how the RMpS metric at the beginning of this subsection.

Finally, the metric that is even closer to the hardware level is the number of logic gates (NLGs) that are used for the hardware (e.g., ASIC or FPGA) implementation of a signal processing device. It is different from the NABSs metric because it indicates the real cost of implementation. Within this metric, the cost of activation functions, represented by look-up tables (LUTs), is also taken into account. However, this metric is not used in this study since it already depends on the particular hardware type that we do not consider here.

Multi-Symbol Equalizer Performance

The results start by presenting the benchmark scenario obtained using nonlinear equalization, i.e., using the equalizers without compression. This case shows the increase in optimum launch power after equalization and the corresponding Q-factor improvement concerning the case without nonlinear equalization. To speed up the training process and the acquisition of results, we have trained this model at the highest launch power and applied the transfer learning strategy [206] for the remaining lower launch powers. For these lower power levels, we fine-tuned the NNs for around five epochs. Fig. 4.13 shows the results of Q-factor over launch power dependence for three transmission scenarios. For the simulated transmission with 20×50 km, the NN equalizer increased the optimum launch power from -1 dBm to 2 dBm. Furthermore, the maximum Q-factor increased by about 2.8 dB, showing a similar maximum performance as that achieved by 3 STpS DBP. For the transmission over 9×110 km system, which has a similar total transmission length but more than doubled span length, it is clear the impact of ASE noise. This effect can be observed in the results depicted in Fig. 4.13 (b).

In this case, the optimum power increased from 4 dBm to 6 dBm and the optimum Q-factor improved by around 1.3 dB due to the equalization. Although the performance improvement enabled by the NN equalizer was lower than in the previous case, it was still higher than the one enabled by the 3 STpS DBP (about 0.7 dB performance improvement).



(a) SC-DP 30GBd; 64QAM; 20×50km SSMF-link (Sim1). (b) SC-DP 34.4GBd; 64QAM; 9×110km SSMF-link (Sim2). (c) SC-DP 34.4GBd; 64QAM-PS; 9×110km SSMF-link (Exp).

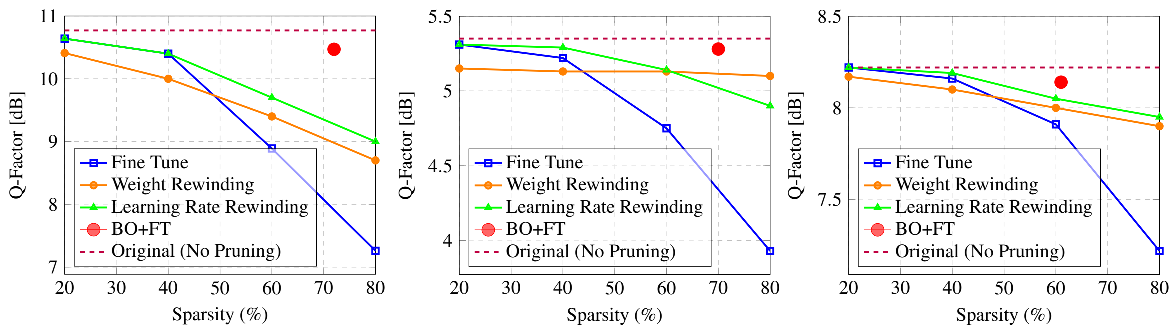
Fig. 4.13 Performance of the proposed NN equalizer, benchmarked against DBP [7] for three different transmission scenarios.

Finally, Fig. 4.13 (c) shows the results obtained in the experimental setup described in Sec. 4.3.5. In this case, it is observed that the NN equalizer leads to an increase in the optimum launch power of about 1 dB, and an increase in the maximum Q-factor of about 0.7 dB. The NN was unable to improve the performance further because probabilistic shaping (considered in the experimental setup) lessens the influence of nonlinearity in fiber-optic transmission and also reduces some other non-deterministic limitations [207]. Following this initial analysis, the best launch power is selected in each transmission setup to evaluate the performance degradation resulting from using the different compression approaches.

Pruning Study

This comparative study starts by doing an analysis similar to what was done in Ref. [159] for image classification, but this time in the context of coherent optical channel post-equalization. Additionally, besides the fine-tuning approach, the weight rewinding, and the learning rate rewinding, in this work, the fine-tuning assisted by the Bayesian optimizer is also considered.

The results for the three transmission setups obtained after pruning are depicted in Fig. 4.14. Similarly to the results shown in Ref. [159], the weight rewinding and the learning rate rewinding outperform the fine-tuning when it is applied a high level of compression ($\geq 50\%$). As an example, for the 60% sparsity and when employing the fine-tuning for retraining, the Q-factor is reduced by 1.9 dB, 0.6 dB, and 0.3 dB for the three considered transmission scenarios, as compared to the original performance. If instead, the rewinding



(a) SC-DP 30GBd; 64QAM; 20×50km SSMF link (Sim1). (b) SC-DP 34.4GBd; 64QAM; 9×110km SSMF link (Sim2). (c) SC-DP 34.4GBd; 64QAM-PS; 9×110km SSMF-link (Exp).

Fig. 4.14 Optical performance when using different pruning techniques for several NN sparsity level. The optimum launch power (without pruning) is set for each case.

approaches are used, the Q-factor degradation of only 1.1 dB, 0.2 dB, and 0.2 dB, and of 1.4 dB, 0.2 dB, and 0.2 dB, are observed for the learning rate and weight rewinding, respectively, when considering the same three transmission scenarios. However, when the fine-tuning is assisted by the BO to select the hyperparameters (as described in Sec. 4.3.2), it is observed that the performance can be significantly improved compared to the other approaches. This approach enabled reaching high sparsity (even higher than the 60% example mentioned above), leading to a Q-factor degradation not exceeding 0.3 dB, 0.1 dB, and 0.1 dB for the three considered transmission scenarios. This result shows the potential of the BO-assisted fine-tuning approach, to outperform the previous model compression techniques.

In fact, the superior performance of the BO-assisted pruning comes from the ability of this approach to cope with the dimensionality changes in multidimensional trainable parameters' space when the NN architecture is pruned. Therefore, the training hyperparameters to achieve a good local minimum may differ from the initial ones, and the BO is capable of identifying this new set of training hyperparameters, while the other methods use their previous values obtained before pruning. However, we emphasize that the BO requires significant computational effort, which means that this method is appropriate mainly for offline applications. When we are interested in achieving the result in the fastest way, the learning rate rewinding is the recommended approach.

Interestingly, the weight rewinding approach performed worse than the fine-tuning approach in cases where the sparsity was lower than 50%, while the learning rate rewinding led to similar or even better performance as compared to fine-tuning. This result can be explained by recalling that the original model was learned using the transfer learning approach, which aids in the learning process by improving generalization and avoiding local minima. When fine-tuning and learning rate rewinding are used, the original weights are the starting point of the pruning process, preserving the good initialization provided by transfer learning. However, in the case of weight rewinding, the weights are reinitialized randomly after the pruning, which can be detrimental to training, thus leading to higher performance degradation.

Regarding the computational complexity reduction in terms of RMpS when using the BO plus fine-tuning (BO+FT) approach, in the result depicted in Fig. 4.14 (a), the BO+FT approach achieved a sparsity of 72%, which represents a reduction from 1.29e+5 to 3.66e+4 in the RMpS value. In the case of Fig. 4.14 (b), the achieved sparsity was 70%, which represents a reduction from 1.42e+5 to 4.31e+4 in RMpS. Finally, in the case depicted in Fig. 4.14 (c), the attained sparsity was 61%, which gives a reduction from 1.42e+5 to 5.58e+4 in RMpS number.

Clustering Study

This section evaluates the weight clustering compression technique. This is the first time that the trade-off between optical performance and computational complexity when using such a technique in optical communications has been assessed. Note that quantization and clustering can be implemented by maintaining a codebook structure that stores the shared weights for each layer. However, in this work, we have also used weight clustering as a pre-step to simplify the problem for the next step, where the traditional quantization techniques are used. The first goal of this subsection is to assess if the weight clustering can reduce the number of multiplications without impacting the performance significantly. The BO described in Chapter 3 is used in this work to find the new training hyperparameters and the number of k-weight clusters throughout the NN-structure, so that the RMpS is given by Eq. (4.58).

Fig. 4.15 depicts the impact of weight clustering on the performance and on computational complexity in the three considered transmission scenarios. This figure demonstrates that weight clustering leads to a small degradation in the Q-factor while still allowing to lower the computational complexity considerably. In Sim1 in Fig.4.15, when 74 clusters were used, it is observed a Q-factor degradation of 0.2 B and a reduction in complexity from 36k to 20k RMpS, when compared to the pruned architecture results. In Sim2 in the same figure, it is observed a similar degradation of the Q-factor and a reduction of complexity from 43k to 19k RMpS when 68 clusters are used. Finally, for the experimental data, and using 62 clusters only, it is observed that the Q-factor remains mostly unchanged, and the complexity is reduced from 55k to 17k RMpS. It is observed that clustering the weights after pruning leads to better results than clustering the original weights. Moreover, the training time is also improved in the former case since fewer parameters need to be learned during the training phase.

Now the focus is on the complexity part of the weight clustering technique, i.e., how much can the number of weight clusters be reduced while still enabling relevant optical performance improvement? Only the Sim1 transmission scenario is considered in the analysis, as this is the case where nonlinear mitigation shows the most noticeable improvement.

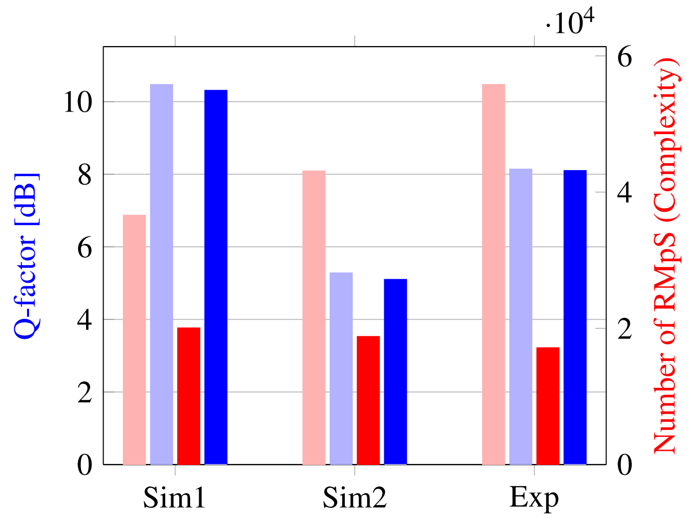


Fig. 4.15 Optical performance and complexity evaluation of the pruning + clustering (dark blue and dark red columns) and pruning only (light blue and light red columns) approaches for the three considered transmission systems (Sim1, Sim2, and Exp).

The potential of the weight clustering technique is now evaluated when using up to four distinct weights. Launch powers in the range from -1 dBm to 2 dBm are tested to assess if the optimum launch power changes when using such an aggressive compression approach. The achieved results are compared to the ones obtained when using linear equalization only (CDC) or 1 STpS DBP. Fig. 4.16 depicts the Q-factor for each equalization approach, as well as the number of RMpS[17]. Fig. 4.16 shows that, when using the CDC, the optimum launch power is -1 dBm, leading to the Q-factor of 7.8 dB (113 RMpS are used in this case). If the reference 1 STpS DBP is used, the optimum launch power changes to 0 dBm, enabling the Q-factor of 9.2dB but requiring 1673 RMpS. We notice that the NN-based equalizer enables outperforming the 1 STpS DBP, which is often used as a benchmark. Indeed, the NN with 4 clustered weights per layer [NN(4W) case in the figure], enables achieving a Q-factor of 9.7 dB (at 1 dBm optimum launch power) using 1091 RMpS. Instead, if only 3 clustered weights are used [NN(3W) case in the figure], a Q-factor of 9.4 dB (at 1 dBm) can

[17]The sparsity of the NN structure was not preserved while doing the clustering, because it was observed that by allowing the zero-value weights, where pruning removes the nodes, Fig. 4.11 (b), to acquire a different (small, but non-zero) value helped in improving the overall performance when an ultra-low number of weight clusters is used. Additionally, the training phase, in this case, took much longer (10k epochs).

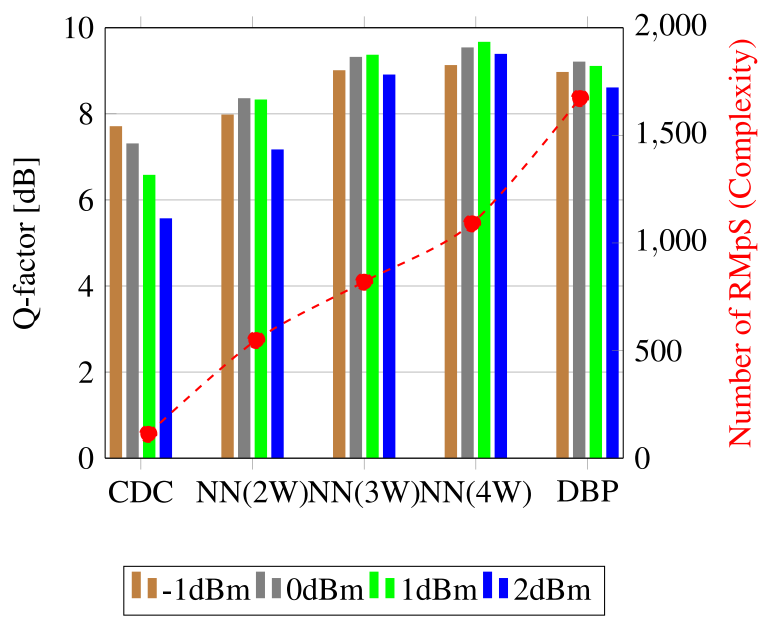


Fig. 4.16 Optical performance and complexity results when employing very aggressive weight clustering in the Sim1 transmission scenario: 2 weights clustering [NN(2W)], 3 weights clustering [NN(3W)], and 4 weights clustering [NN(4W)]. The traditional CDC and reference 1 STpS DBP results are shown as benchmark.

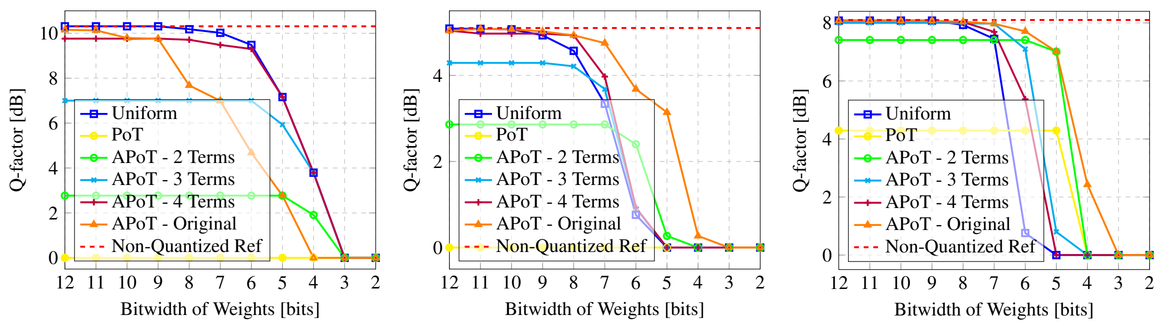
be reached, requiring 820 RMpS only. As expected, using 2 clustered weights [NN(2W) case in the figure] leads to the worst performance, where it can be achieved the Q-factor of 8.4 dB (at 0 dBm), thus still outperforming the CDC, but at the expense of 549 RMpS complexity.

Finally, if it is considered that the multiplier complexity is proportional to $b_i b_w$ (as described in Sec. 4.1), the savings in complexity enabled by the clustering technique can be even higher than the ones indicated above. Indeed, considering that $b_i = 8$ bits for all cases, the coefficients of the CDC and DBP filters are also represented by 8 bits. However, for the NN with 3 and 4 clustered weights, the weights can be encoded using just a 2-bit format. So, for the cases of 3W and 4W NN, which performed better than 1 STpS DBP, the complexity calculated as RMpS $* b_i * b_w$, is just 1.82 and 2.42 times higher than the CDC one (and 8.15 and 6.13 times lower than the 1 STpS DBP), respectively.

Quantization Study

Quantization is the other approach considered in this work to significantly reduce the computational complexity of equalizers. The PTQ homogeneous, PTQ heterogeneous, QAT homogeneous, and QAT heterogeneous approaches are considered in this subsection. In each case, a combined biLSTM+CNN equalizer whose weights have undergone the clustering procedure depicted in Fig. 4.15 is quantized. The performance and complexity in terms of

BoPs and NABSs of the different quantization techniques are further assessed for different bit precisions.



(a) SC-DP 30GBd; 64QAM; 20×50km SSMF link (Sim1)
(b) SC-DP 34.4GBd; 64QAM; 9×110km SSMF link (Sim2)
(c) SC-DP 34.4GBd; 64QAM-PS; 9×110km SSMF-link (Exp)

Fig. 4.17 Performance of the Post Training Quantization (Homogeneous Approach).

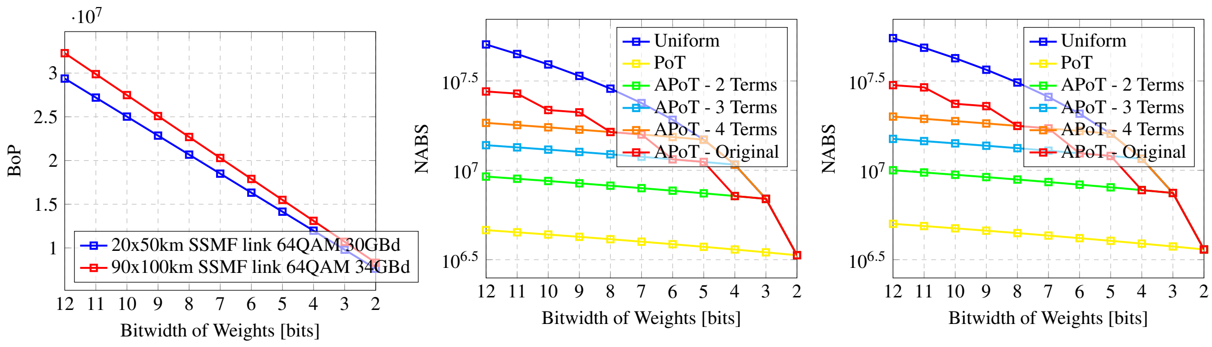
PTQ homogeneous approach Let us assume that all weights in the structure are quantized uniformly and with the same bit precision. Fig. 4.17 depicts the Q-factor as a function of the bit precision for the three considered transmission scenarios and using the APoT with 2, 3, and 4 additive terms quantization technique as well as the original version in [124], the uniform quantization and PoT[18].

From the results depicted in Fig. 4.17, it is possible to underline the noticeable impact of sparsity. The PoT and APoT techniques were purposely designed to have the majority of the quantization levels close to zero since the weight distribution after training also shows a concentration of weights close to zero value, see Fig. 4.11. However, when pruning the NN structure, such weights are removed, Fig. 4.11 (b), the quantization levels above the pruning threshold are no longer used and, more importantly, the remaining weights are underrepresented. Consequently, the uniform quantization shows the best performance (for the reduced bitwidth of the weights) in the case shown in Fig. 4.17 (a), where the sparsity is 72%, whereas the APoT and POT reveal a better performance in the scenario depicted in Fig. 4.17 (c), where the sparsity is 60%. Interestingly, up to 8 bits precision, it was always possible to find a quantization scheme that provides similar optical performance as when using the original 32 bits precision for the three considered transmission scenarios. For a high precision bitwidth, the uniform quantization always shows superior results, whereas for a lower bit precision (say, for less than 8 bits) and when the weight distribution is not heavily compromised by sparsity, the original APoT introduced in [124] results in the best

[18]t was established a floor value of 0dB for the Q-factor since lower Q-factor means the information is completely corrupted.

performance. Here it is highlighted that, when doing with the PTQ strategy, the weight distribution must serve as the main indicator to select the best type of quantization.

Now it is assessed the computational complexity of the different quantization techniques. In such analysis, it was considered $b_i = b_a = 16$ bits and b_w changing between 12 and 2 bits. Fig. 4.18a depicts the BoP metric for equally compressed models, showing that the BoP decreases almost linearly with the value of b_w. Since the Sim2 transmission scenario requires a NN structure with more hidden units and CNN filters than Sim1, the number of BoPs for this case is also higher than that for Sim1. This analysis evaluates the total number of operations needed, as it is usually done in the literature [122], and therefore, does not account for the benefits stemming from weight clustering.



(a) BoPs for the NN used in Sim1 and Sim2.

(b) NABS for the NN used in Sim1.

(c) NABS for the NN used in Sim2 and Exp.

Fig. 4.18 Computational complexity when using uniform quantization (a) and when different types of quantization are used (b/c). b_i and b_a have 16 bit precision whereas b_w has a value in the range of 12 to 2 bits.

Nevertheless, and as was mentioned in Sec. 4.1, when comparing the use of different quantization strategies and bitwidth precisions, the BoPs metric can not be recommended, as it, actually, does not account for the effect resulting from different quantization strategies. To have a better metric, the NABS metric ought to be used, as it allows comparing the result of the model compression in terms of the number of additions and bit shifts. Figs. 4.18b and 4.18c show the NABS as a function of b_w (the bit-precision) for the different quantization strategies employed in this work. As expected, the uniform quantization technique leads to the highest complexity, whereas the PoT quantization gives the smallest one. The APoT quantization leads to a complexity in-between the uniform and PoT approaches. In the APoT case, the complexity depends not only on b_w, but also on the number of additive terms that are considered. Interestingly, the least complex APoT strategies, i.e., those with the smaller number of additive terms, are the ones leading to better performance for the low bit precision region, see Fig. 4.17. It is also interesting to note that, when we reduce the bit precision to 5 bits, which already leads to high-performance degradation, the NABS using uniform quantization becomes the same as that when using the APoT with 4 additive terms. If the bit precision is further reduced to 4 bits, the NABS using uniform quantization is the same as that when using the APoT with 3 additive terms. In the same way, if the bit precision is reduced to 3 bits, the NABS using uniform quantization is the same as that when using the APoT with 2 additive terms and, finally, if the bit precision is reduced to 2 bits, the NABS using uniform quantization is the same as that when using the PoT quantization.

PTQ heterogeneous approach

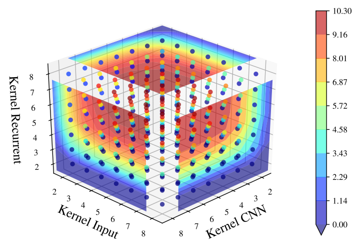
When using the heterogeneous approach, the bit precision and quantization method are allowed to vary in different parts of the NN structure. For simplicity, here we only consider the uniform quantization, the original APoT, and the mix, where different types of quantization are used throughout the NN structure.

Fig. 4.19 depicts 3D plots with the Q-factor as a function of i) the bitwidth of the input and recurrent kernel of the LSTM layer, and ii) the bitwidth of the filter kernel of the CNN layer. A gradient of colors is used, with the warmer colors corresponding to the higher Q-factor, so we can identify which combination of b_w values gives the best performance.

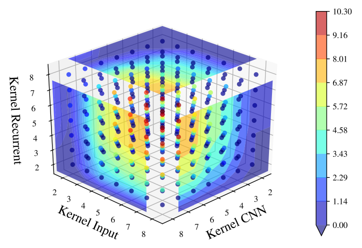
Similarly to the homogeneous approach (see Fig. 4.17), the uniform quantization is the most interesting solution for the Sim1 transmission scenario, whereas the APoT leads to better performance for the Sim2 and Exp scenarios.

However, when using heterogeneous quantization, it is observed that lower complexity can be achieved. For example, considering the Sim1 transmission, and in order to achieve the same optical performance as when using the 1 STpS DBP, one may quantize all weights with 6 bits (homogeneous quantization), or further reduce the recurrent kernel and CNN kernel to 5 bits using heterogeneous quantization without any significant degradation in optical performance. Similar results can be observed in the Sim2 and Exp transmission scenarios.

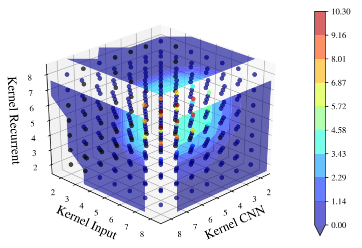
In addition to using different bit precision in different parts of the NN, it can also use different types of quantization to improve the performance. For this objective, this work has used a grid search, testing different combinations of quantization types. The result of this optimization is referred to in Fig. 4.19 as Mixed Quantization. By following such an approach, it can be observed an improvement in the hot area of the Sim 1 results when it is quantized the input and CNN kernels with uniform quantization and the recurrent kernel with APoT using the original terms; for Sim2, it is quantized the input with the 3 terms APoT and the recurrent and CNN kernels with the original APoT. Unfortunately, no significant optical improvement was observed when compared to using just the original APoT. In this case, just an improvement of 0.1dB in Q-factor is achieved in the mix quantization, where it



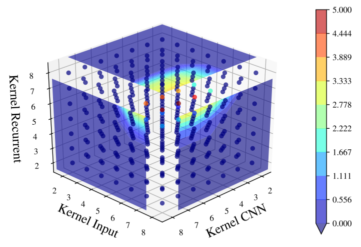
(a) Uniform Quantization (Sim1).



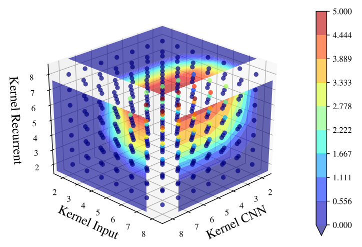
(b) APoT Original (Sim1).



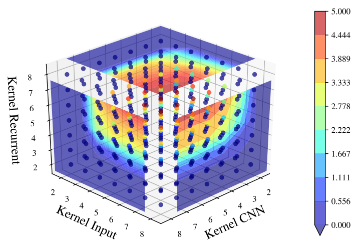
(c) Mixed Quantization (Sim1).



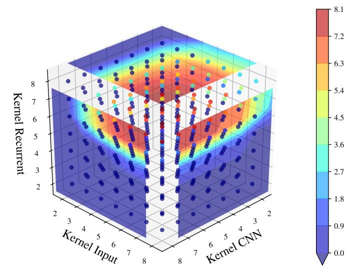
(d) Uniform Quantization (Sim2).



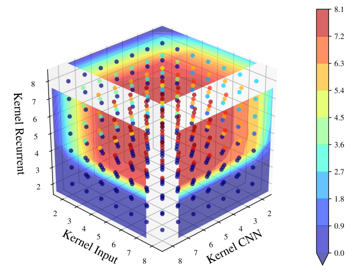
(e) APoT Original (Sim2).



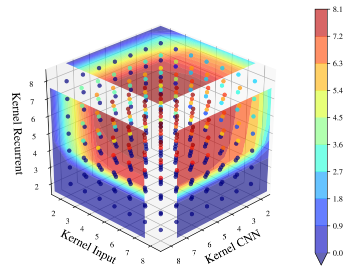
(f) Mixed Quantization (Sim2).



(g) Uniform Quantization (Exp).



(h) APoT Original (Exp).



(i) Mixed Quantization (Exp).

Fig. 4.19 Performance of the Post Training Quantization (Heterogeneous Approach).

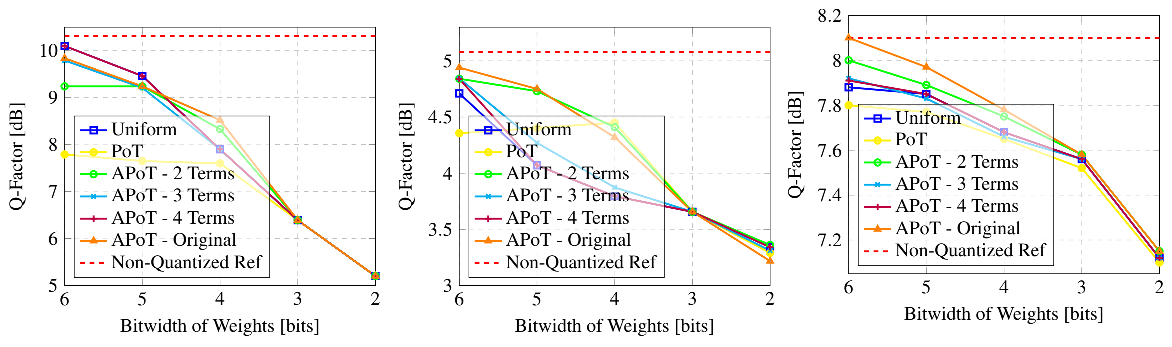
is quantized the input and CNN kernels with the original APoT and the recurrent kernel with the 2 terms APoT, compared to the case with all weights quantized with the original APoT.

QAT homogeneous approach

This work now evaluates the potential of implementing quantization during the training phase of the NN to mitigate the error introduced by the low bit precision of weights. Since QAT leads to at least as good performance as PTQ and the results depicted in Fig. 4.20 show that the optical performance is highly impacted when the bitwidth decreases below 6 bits, it will focus the QAT analysis in this region (between 6 and 2 bits).

Fig. 4.20 depicts the Q-factor as a function of the bit precision for the three considered transmission scenarios. The considered quantization techniques are the Uniform, PoT, APoT with 2, 3, and 4 additive terms and the original version of APoT that can be found in

Ref. [124]. In order to better illustrate the impact of QAT, it was compared to the results in Figs. 4.17 and 4.20. As an example, first consider that all weights are quantized equally with 4 bits. For Sim1, Fig. 4.17 (a) shows that the uniform quantization provided the best performance with a Q-factor close to 4 dB, whereas Fig. 4.20 (a) shows that using the APoT original and following a QAT strategy enables reaching Q-factor values close to 8.5 dB. Similar conclusions can be drawn in Sim2 and Exp transmission scenarios: Fig. 4.20 (b) and (c) show that optical performance is highly impacted when weights are quantized to 4 bits whereas, when implementing QAT, the original APoT provides Q-factor values close to 4.5 dB and 7.8 dB, respectively. These results demonstrate the huge positive impact of QAT on the compression of the NN model. Moreover, when further reducing the bitwidth, the optical performance degradation is not as drastic as in the case of PTQ.



(a) SC-DP 30GBd; 64QAM; 20×50km SSMF link (Sim1).

(b) SC-DP 34.4GBd; 64QAM; 9×110km SSMF link (Sim2).

(c) SC-DP 34.4GBd; 64QAM (PS-8bits/4D symbol); 9×110km SSMF link (Exp).

Fig. 4.20 Performance of the Quantization Aware Training (Homogeneous Approach).

The original APoT quantization technique leads to very good optical performance in most of the cases depicted in Fig. 4.20. This good performance is a direct consequence of performing quantization during the training phase. Indeed, in this case, the weights remaining after the pruning are no longer underrepresented, but rather adjusted to the non-uniform levels of quantization, leading to good performance. Nevertheless, the difference between PoT and APoT is no longer as significant as the one observed in Fig. 4.17. As a consequence, and as can be seen in Fig. 4.20, no universal conclusion can be drawn about which is the best quantization technique for QAT.

In general, the original APoT and the APoT with 2 terms were the two techniques that performed the best for the range of bitwidth studied and transmission scenarios. But, since the APOT with 2 terms uses only one adder and bit shift for each multiplication, it is probably the best choice in terms of a trade-off between optical performance and computational complexity.

It is important to stress that the training of such NN structures is unstable. Consequently, the model needs to be monitored during training. In this work, early stopping was not used for QAT. Instead, the quantized NN structure was trained for 5000 epochs, with the intermediate NN models leading to the best Q-factor being saved and used as the final NN. As described in Ref. [208], the training phase of the quantized model can suffer from learning problems (e.g., exploration vs. exploitation trade-offs). As suggested in that reference, this work also used large mini-batch sizes (≥ 4000), since “this shrinks the variance of the gradient distribution without changing the mean and concentrates more of the gradient distribution towards downhill directions, making the algorithm more greedy”. As a result, when performing the QAT, the training hyperparameters must be properly set, and the training will most likely require a higher number of epochs as the bitwidth of the weights is reduced.

QAT heterogeneous approach

In this compression technique, and as described in Sec. 4.3, the BO is used to determine the ideal bitwidth per layer as well as the type of quantization in each layer (and other hyperparameters, like the learning rate), seeking to improve the overall performance.

Similarly to the PTQ case, when going from the homogeneous to the heterogeneous approach, the bitwidth and quantization types employed can be different in different parts of the NN architecture. Like in the case of heterogeneous PTQ, the performance of the heterogeneous approach is evaluated by considering the different bit precision of the input kernel of the LSTM layer, the recurrent kernel of the LSTM layer, and the filter kernel of the CNN layer. The values obtained by the BO can be found in Table 4.7 for the considered transmission scenarios as well as the Q-factor achieved when i) the NN model is not quantized (w/o Quant.), ii) the model is only quantized (PTQ), and iii) the model is simultaneously quantized and fine-tuned (QAT). This table shows that, at the low levels of bit precision, the PTQ corrupted the NN model completely. Nevertheless, the QAT adapts the weights in such a way that only a small degradation in performance is observed.

Table 4.7 Results obtained using the Bayesian Optimizer. The performance results of both QAT and PTQ are depicted to highlight the benefit of QAT.

Scenario	b_w			Quant. Type			Learning	Mini-Batch	Q-factor	Q-factor	Q-factor
	Input	Recurrent	CNN	Input	Recurrent	CNN	Rate	Size	PTQ	QAT	w/o Quant.
Sim1	3	5	4	APOT Orig.	APOT Orig.	Uniform	0.00114	4347	0.18 dB	8.6 dB	10.31 dB
Sim2	4	3	4	APOT 2 Terms	APOT Orig.	APOT Orig.	0.00132	6568	0 dB	4.5 dB	5.1 dB
Exp	2	3	2	APOT Orig.	APOT Orig.	APOT Orig.	0.00085	5253	0 dB	7.41 dB	8.1 dB

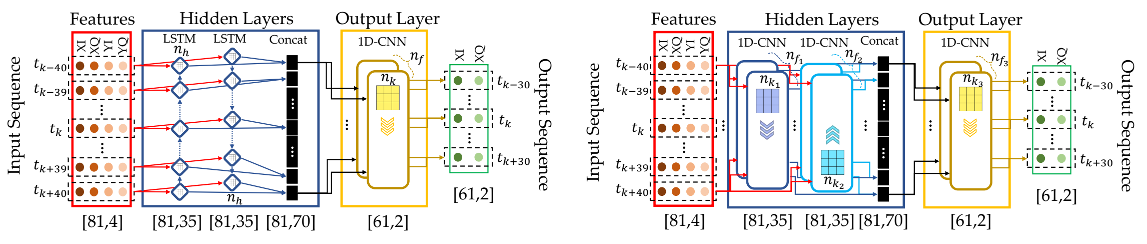
To conclude, here, it was evaluated the complexity of the different approaches. This was done in terms of NABS insofar as it was employed simultaneously different bit precision and quantization techniques (see 4.1). The reference complexity is the “traditional” uniform quantization with 8 bits in all layers, which was compared against the heterogeneous structures depicted in Table 4.7. For Sim1, the reference complexity is 28.6M NABSs while, for the heterogeneous architecture, the complexity is 10.9M NABSs, which translates into a complexity reduction of $\approx 62\%$. Similarly, for Sim2 and Exp, the reference complexity is $\approx$ 31M NABSs, whereas after heterogeneous QAT it is $\approx$ 7.9M and $\approx$ 7.4M NABSs, for the two cases, respectively, representing a reduction of $\approx 76\%$.

4.4 Brief discussion on the FPGA implementation of neural network-based equalizers

This section makes an important step forward in assessing the viability of NN-based equalizers for industrial applications by benchmarking i) their performance versus the 1-step-per-span (StpS) DBP using 2.3 samples/symbol (Sa/symbol) in experiments and ii) their computational complexity by comparing an FPGA implementation against the full electronic chromatic dispersion compensation (CDC) block (used, e.g., in standard DSP chain), which needs far fewer resources than 1 StpS DBP. In addition, for the first time, it is presented the FPGA implementation of a NN-based equalizer that employs the bidirectional recurrent layer with LSTM cells (biLSTM). By transmitting a 34 GBd single-channel, dual-polarization (SC-DP) 16QAM signal over 17×70 km of large-effective area fiber (LEAF) (both simulated and experimental), reporting $\approx$ 1.7 dB Q-factor improvement over a standard DSP chain while requiring only $\approx 2.5\times$ more FPGA resources than the CDC block implementation to achieve 400G transmission.

4.4.1 Converting Python NN models to FPGA designs

The two NN architectures for the equalizers investigated in this work are depicted in Figs. 4.21a and 4.21b for the biLSTM-based equalizer and the deep CNN-based equalizer, respectively. The shape of both architectures is similar, but the nature of the mathematical operations in each case is different: the biLSTM contains recursive connections, whereas the deep CNN is a feedforward network. A total of 81 symbols are used as input, allowing to simultaneously retrieve 61 symbols at the equalizer's output. Recovering 61 symbols in parallel at the equalizer output, allows the FPGA realization to have a higher throughput, which is one of the main desirable design features when building a DSP block for coherent transmission systems. In Fig. 4.21a, the hidden layer consists of a biLSTM layer with $n_h = 35$ hidden units. In Fig. 4.21b, the hidden layer is made up of a CNN layer with 70 filters ($n_{f_1} = n_{f_2} = 35$), with zero padding applied to retain the shape and the kernel



(a) biLSTM+CNN. (b) Deep CNN.

Fig. 4.21 Structures of NN-based equalizers taking 81 symbols as input to recover 61 symbols in parallel at the output: (a) the recurrent equalizer using a bidirectional LSTM layer containing 35 hidden units, and (b) the feedforward equalizer using a 1D-convolutional layer consisting of 70 filters ($n_{f_1} = n_{f_2} = 35$).

size $n_{k_1} = n_{k_2} = 11$. The output layer in both designs is a convolutional layer with $n_f = 2$ filters, a kernel size $n_k = 21$, and no padding. On the basis of a grid search analysis, these parameters were chosen to meet the hardware limitations, throughput requirements, and optical performance needed for this FPGA realization. The activation functions for both hidden layers were hyperbolic tangent ($tanh$), and the output layer is linear.

Focusing now on the biLSTM+CNN architecture implemented in this work, the mean square error (MSE) loss estimator and the classical Adam algorithm for the stochastic optimization step [201] were used when training the weights and bias of the NN. The training hyperparameters (mini-batch size equal to 2001 and learning rate equal to 0.0005) were found using the Bayesian optimization procedure described in [116]. The NN training was carried out by backpropagation for 30000 epochs with a fixed set of hyperparameters[19]. The BER is evaluated after each training epoch. For training, it was used a fixed dataset with 2^{20} symbols, and, at every epoch, it was picked 2^{18} random input symbols from this dataset. For the testing and validation, it was used a never-before-seen dataset with 2^{18} symbols. Both NN models were trained, validated, and tested using the same datasets. The weights were saved at the epoch at which the BER measured using the validation dataset was the lowest (early stopping).

The HLS method provides a design flow in which the desired function, i.e., an NN in the case considered, can be specified in C++ and then automatically converted to VHDL. This strategy is preferred because it separates the FPGA technology-specific implementation features, such as clocks and logic cell topology, from the NN's functionality. Working at a higher level of abstraction, the intended functionality can be the focus of attention and be described more readily in fewer lines of code [210].

[19]For applying the model to other launch power values, it was used transfer learning [209], which allowed utilizing less than 5 epochs to adjust the NN weights to the other launch powers.

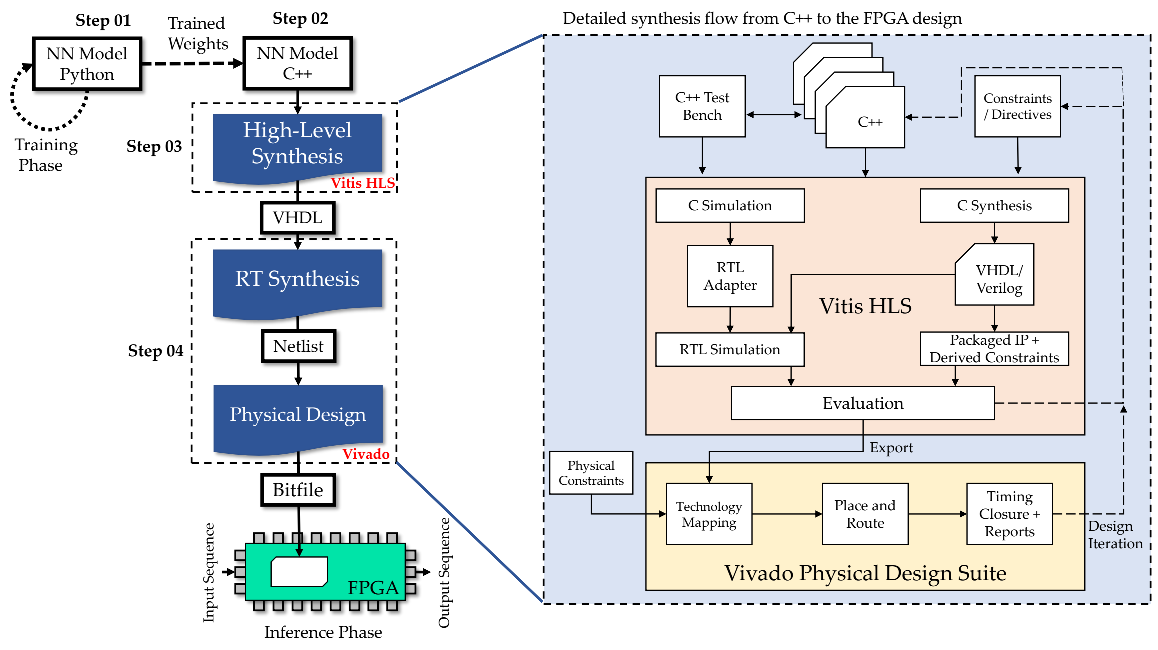


Fig. 4.22 High-level synthesis flow – from C++ to FPGA realization.

In addition, functional verification in C++ is much faster than functional simulation in VHDL. This makes it easy to test and debug the design. At this point, it is important to remember that the functions being described will be run on hardware. The HLS supports a substantial number of C++ syntax, but not all, because some cannot be implemented in an FPGA or ASIC. In this case, memory allocation is a key part of writing the C++ code that needs to be properly assessed. In the FPGA, the memory allocations are static and assigned during the mapping phase of the physical design flow. Dynamic memory allocations found in many C++ standard library functions cannot be supported and must be avoided or replaced with structures optimized for the implementation in an FPGA by using the libraries provided by the HLS tool supplier, in this study, Xilinx. Also, Operating System (OS) functions such as file read/write and date/time cannot be implemented in an FPGA; all data transmitted into and out of the FPGA must use input/output ports.

Consequently, two versions of the C++ codes were generated. The first is called here the test bench, while the second is the function to be implemented in FPGA. The test bench function reads the previously saved signal inputs and weights learned in Python and converts them to a fixed-point format (int32). These values are then incorporated into the function that describes the equalizers investigated in this study. The function is the C++ translation of the Python NN equalization architecture, using fixed-point arithmetic operations. After the equalization, the outputs of the function (the signal that has been equalized) are delivered to the test bench code that checks the MSE. Note that, here, it was not studied the impact of further reducing the quantization accuracy, since this would require some further work to overcome the quantization error caused by both the input signal and the weights. it was chosen the int32 format because, by using it, the implementation can take advantage of the simplified integer arithmetic while observing no significant performance reduction compared to the floating-point BER evaluated in Python.

However, when using the HLS, even though the conversion to VHDL is automated, some design intervention is still necessary, and engineering decisions must be taken to achieve the desired performance. HLS supports a set of directives, or pragmas, that can be used to modify the behavior of the HLS C++-synthesis stage to facilitate these interventions [211]. By utilizing pragmas, in order to discover the best implementation, it is useful to investigate several design structures without re-coding them. Although there are a variety of different pragmas, this study has primarily utilized those pertaining to pipelines, loops, and arrays.

Pipelines allow the parallel execution of operations within a function, lowering the number of clock cycles between commencing loop iterations; these clock cycles are referred to as the Iteration Interval (II). Each loop iteration does not need to end before the next one begins, i.e., the iterations can overlap. The number of pipeline stages can be controlled by setting the value of II using the HLS pipeline pragma. Setting the II to 1, as it is done in this project, enables each cycle to begin with a new iteration.

Loops can be unrolled and flattened. By default, loops within a function are left rolled, which means that the loop body is executed sequentially, utilizing a single set of logic resources. The minimum loop delay is then equal to the number of loop iterations. Unrolling generates several copies of the loop body logic, enabling parallel execution and reduced latency at the expense of additional size. Loops can be completely or partially unrolled, resulting in either one copy of the loop body every iteration for optimum throughput or in fewer copies for reduced area cost. Flattening transforms a hierarchy of nested loops into a single loop, which eliminates a clock cycle delay while traveling between higher and lower nested loops and can help with better optimization of the loop logic. In view of this, the loops in the feedforward NN layers can be flattened; however, the loops in the recurrent NN layers cannot be flattened due to their memory dependence and imperfect loops. Therefore, for the implementation of the recurrent NN layer, it was only flatten the loops relating to the matrix multiplication that occurs internally within each LSTM cell.

Arrays can be partitioned and reshaped. HLS will implement arrays in the C++ code as memory blocks in the VHDL description. This can cause a restriction in the design concurrency as FPGA memory blocks only have 2 access ports, which, if the designer has also used the previously mentioned unroll and pipeline pragmas, will need to be shared between all instances of the loop body. By reshaping and partitioning the array, the size and number of these memory blocks can be controlled. To enable the successful flattening of the loops in each NN architecture, the non-equalized signal (input signal) and the weights of the NN architecture are totally partitioned here. The final stage of the HLS step is to export the generated VHDL and derived constraints for use in the physical design step.

Here, it is emphasized that NNs are an excellent candidate for exploiting the benefits of HLS, as their nested architecture consisting of multiple layers and several multiply/accumulate functions can make good use of the loop and pipeline directives to investigate the trade-off between area and latency to meet the design requirements.

The area and timing reports generated by the HLS stage are still simply estimates of the final design performance based on the technology-specific data libraries for each FPGA; the actual performance cannot be determined until the physical implementation is complete. In this work, the Vivado Design suite from Xilinx is utilized. The physical implementation is performed by Vivado in three steps: technology mapping, placement and routing, and timing analysis.

Technology Mapping. Within this step, the VHDL source code is translated into primitive logic gates and boolean equations, followed by mapping these gates onto FPGA customizable logic blocks containing D-type FFs (DFFs) and RAM-based LUTs or more specialized functional cells, such as DSPs. During the technology mapping, the design is optimized and unnecessary logic is eliminated.

The next two steps constitute an iterative process executed automatically by the tool based on design constraints, such as a clock frequency. These limitations can be inherited from the HLS stage or defined in Vivado. The Vivado tool imposes sets of restrictions through established optimization strategies, which are discussed in detail in the vendor user manuals [212], and which the user can apply depending on the design goals. The optimal technique is determined by balancing computer runtime and outcomes. In [213] it can be found all potential pairings of synthesis and optimization procedures in Vivado using a high-speed pulse width modulation circuit as a target design; in this web blog, a comprehensive evaluation of the runtime versus performance of the different Vivado optimization methodologies is given. As the objective of this work was to maximize the throughput, it did not examine the solutions targeting chip area, power, or reduced runtime. Therefore, the Vivado configuration called "Performance ExtraTimingOpt" is adopted in this work, since it effectively optimizes the throughput by reducing timing slack [213].

Placement and Routing. This stage positions the logic blocks developed during the mapping phase, onto the specified elements of the FPGA cell array, and configures the signal routing channels between them. The placement algorithm starts from a random seed position and then moves functions to the cell array based on the degree of congestion for the parts of the die and the fanout of the driving function.

Time Analysis. This stage compares the design with the applied timing constraints to determine whether the overall performance requirement has been met. Timing analysis, in particular, requires a grasp of the FPGA structure and how the design has been mapped onto the array. It may be necessary to return to the HLS phase to apply more directives or adjust the function architecture to achieve timing closure. The time for a data (signal) to travel between two points is determined by a variety of factors, including DFF switching time, setup requirements (the time at which the signal must arrive at the destination before the capturing clock edge), logic and routing path delays, and clock edge uncertainty due to jitter and clock path skew. Here, it is pertinent to define the negative timing slack.

The negative timing slack indicates that the total delay in the data path between two DFFs is greater than the requested clock period. In this negative slack case, the NN has many nested loops, as discussed in the previous section; unrolling these loops would lead to a larger design consuming more logic area, but leaving large loops, i.e., the loops with a high number of iterations, can produce long logic multiplexer paths as the inputs to the loop logic are selected. Using Vivado timing analysis reports and annotated netlist viewer, it is possible to identify which paths can be the cause of the failing paths. In this case, the solution was to return to the C++ source and reorder the nested loops so that the outer loop, which was not unrolled, had fewer iterations; this approach reduced the size of the logic chain in the multiplexer path. This work has also decreased the clock frequency for each of the designed blocks to guarantee that a zero negative timing slack was achieved in all FPGA designs.

4.4.2 Simulation and experimental setups

At the transmitter, a DP-16QAM 34 Gbd symbol sequence was mapped out of the data bits generated by a Mersenne Twister algorithm. Then, a digital RRC filter with 0.1 roll-off was applied. The resulting filtered digital samples were resampled and uploaded to a digital-to-analog converter (DAC) operating at 88 GSamples/s. The output of the DAC was amplified by a four-channel electrical amplifier which drove a dual-polarization IQ Mach-Zehnder modulator, modulating the continuous waveform carrier produced by an external cavity laser at $\lambda = 1.55\,\mu m$. The resulting optical signal was transmitted over 17×70 km spans of LEAF, with the loss in each fiber span being fully compensated at its output using Erbium-doped fiber amplifiers (EDFAs). The EDFAs noise figure was in the 4.5 to 5 dB range. The parameters of the LEAF are: attenuation coefficient $\alpha = 0.225$ dB/km, chromatic dispersion coefficient $D = 4.2$ ps/(nm·km), and effective nonlinear coefficient $\gamma = 2$ (W·km)$^{-1}$. On the RX side, the optical signal was converted to the electrical domain using an integrated coherent receiver. The resulting signal was sampled at 80 Gsamples/s by a digital sampling oscilloscope and processed using the offline DSP described in [113]. Thereafter, the resulting soft symbols were used as input for the NN equalizers. Finally, the pre-FEC BER was evaluated from the signal at the NN output. Concerning the simulation, it mimics the experimental transmission setup. The signal propagation along the fiber was simulated by solving the Manakov equation. At the receiver, after full CDC and downsampling to the symbol rate, the received symbols were normalized to the transmitted ones. In addition, Gaussian noise was added to the data signal, representing additional transceiver distortions observed in the experiment. As a result, the Q-factor level of the simulated data matched the experimental one

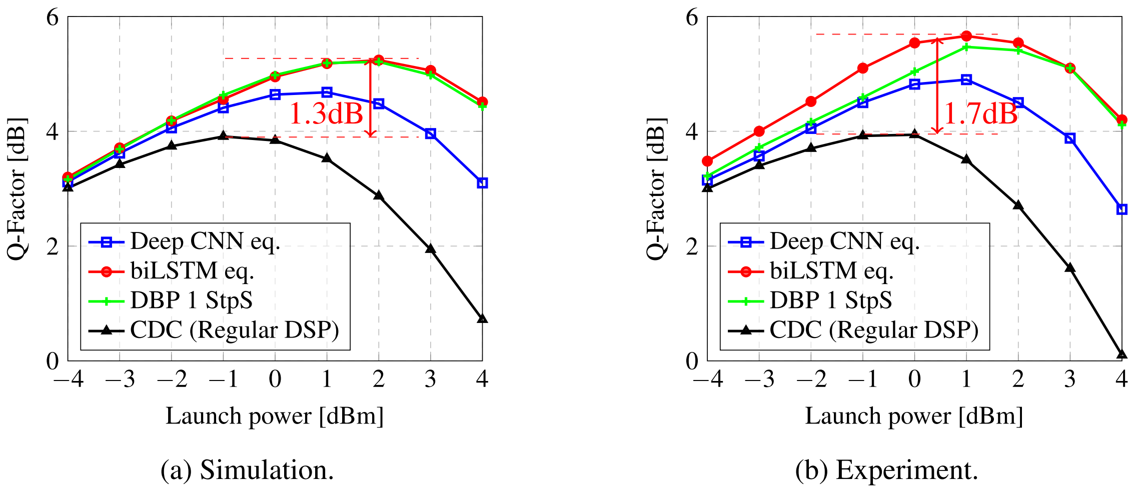
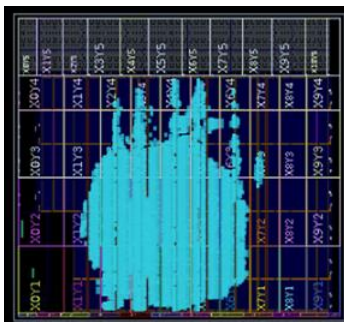
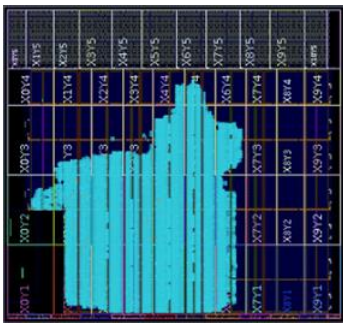


Fig. 4.23 Q-factor versus launch power for (a) simulation and (b) experiment corresponding to the transmission of a Single Channel DP-16QAM 34 GBd signal along 17×70km of LEAF. The difference between the CDC and the biLSTM equalizer's results is marked with red arrows. The case of the floating-point models' accuracy for the different types of NN equalizers (described in the legends), together with the 1StpS DBP and CDC performance curves.

4.4.3 Results and discussions

Fig. 4.23 summarizes the performance of the NN-based equalizers compared to the 1 StpS DBP and CDC over different launch powers for simulated and experimental data. The results referring to the simulated data are given in Fig. 4.23a. The biLSTM equalizer shows approximately the same performance as a 1 StpS DBP while improving the optimal power from −1 dBm to 2 dBm and the Q-factor by 1.3 dB with respect to the CDC. Regarding the deep CNN, it performs worse than the biLSTM and the 1-StpS DBP; the optimal power is improved from -1 dBm to 1 dBm and the Q-factor raises by 0.8 dB compared to the CDC. On the other hand, with the experimental data[20], it was observed in Fig. 4.23b that the biLSTM outperforms the 1-StpS DBP, especially in the noise-dominated region. For the 1-StpS DBP case with the experimental data, the Q-factor increases by 1.3 dB in the simulation and by 1.5 dB in the experiment. Compared to the simulation, the NN-based equalizers in the experiment also lead to a higher gain for the Q-factor; when having CDC as a baseline, the gain improves from 1.3 dB (simulation) to 1.7 dB (experiment) in the case of biLSTM equalizers, and from 0.8 dB (simulation) to 1 dB (experiment) for the deep CNN. The optimal power also shifts from 0 dBm to 1 dBm in both cases. This shows that the NN has the potential to reduce the effects of both the Kerr nonlinearity and the component-induced corruptions. Because all component impairments in the simulations were modeled with white noise, whereas in reality some of these component noises can be mitigated deterministically, this equalizer could enhance the Q-factor slightly further in the experimental case. In particular, the biLSTM equalizer beats the deep CNN equalizer because the biLSTM is a recurrent-based NN that benefits from temporal sequential data learning [4, 151].


(a) biLSTM eq.


(b) Deep CNN eq.

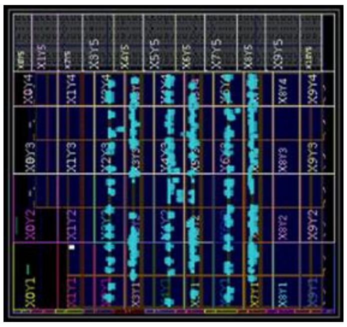

(c) CDC block

Fig. 4.24 Implementation in the EK-VCK190-G-ED Xilinx FPGA [8] of the (a) biLSTM eq., (b) Deep CNN eq., (c) CDC block.

Now that the Q-factor performance of these NN equalizers was presented, their computational complexity will be analyzed. Figs. 4.24a, 4.24b, and 4.24c show the real implementation and used chip areas of the biLSTM, the deep CNN equalizers, and the CDC, respectively, on the state-of-the-art EK-VCK190-G-ED Xilinx FPGA chip [8]. This chip is partitioned into 40 clock regions, with the blue areas in Fig. 4.24 representing the used chip resources. Table 4.8 summarizes the most important information on the VCK 190 implementation of the biLSTM, deep CNN equalizers, and the CDC, in terms of latency, clock frequency[8],

[20]The Q-factor obtained with the Python model and the FPGA implementation were virtually identical because it was not considered quantization.

resources required, the utilization of the resources, and throughput. The achieved throughput ($\mathrm{T_a}$) can be calculated as follows:

$$\mathrm{T_a} = \mathrm{clock} \times \log_2(\mathrm{QAM}) \times n_{\mathrm{out}}, \tag{4.69}$$

where clock is the clock frequency, QAM is the modulation format, $\log_2(\mathrm{QAM})$ is the number of bits per symbol, and n_{out}, is the number of parallel symbols it is recovered in the output; in this case, $n_{\mathrm{out}} = 61$.

Three important conclusions can be drawn from that figure. First, although the biLSTM renders a higher Q-factor improvement, due to the equalizer's recurrent structure its feedback loop connections cause a bottleneck in the design, resulting in higher latency (33 μs) and lower clock frequency (270 MHz). On the other hand, deep CNN and CDC can be parallelized more efficiently. The parallelizability brings about a reduction in their latency to 19.9 μs for the deep CNN, and 1.1 μs for the CDC. Due to the fact that the CDC has one filter, whereas the deep CNN has 70 filters, the parallelization is easier in the CDC implementation because of hardware restrictions, resulting in an operating frequency of 524 MHz for the CDC case, and 244 MHz for the deep CNN case. Fig. 4.24c clearly shows the CDC parallelization. A long latency increases the time required to process each time step, leading to slower overall processing times and reducing the speed of the network. This can be problematic in real-time applications where a fast response is necessary. To mitigate the impact of long latency, design optimization techniques can be applied to reduce the latency and increase the performance, such as reducing the size of the memory blocks, using more efficient algorithms, and implementing pipelining. In this work, due to the offline processing consideration, all the input is already in the memory, and the request of the sequence with 81 symbols as inputs are created so that the functioning of the NN-based equalizer parallelization works.

Second, regarding the FPGA utilization, the biLSTM equalizer is the only one using Block Random Access Memory (BRAM)[21] to store future/past recurrent states, while both CNN and CDC do not need such blocks. BRAM is used to store the hidden states of an LSTM, which can then be fed back into the network at the next time step to maintain its memory, while in the case of the feedforward structures, the special LUTs are used to store the coefficients. Note that the number shown in the tables is the number of BRAM blocks, which was the automatic result reported after the Synthesis step (Vivado). By using BRAM, the hidden state information can be stored in a dedicated memory block, separate from the

[21]BRAM is a type of memory in FPGAs that is used to store large amounts of data, typically used in FPGA designs to implement memory-intensive functions such as image and video processing, buffers, and large arrays. Unlike other memory elements in an FPGA, BRAM is a dedicated memory that is separate from the FPGA's general-purpose FFs and LUTs.

other resources in the FPGA. This can lead to improved performance, as memory accesses are optimized and dedicated resources are used for memory storage. However, the size of the BRAM blocks and the memory requirements for the recurrent connections should be carefully considered when designing an LSTM on an FPGA. The available BRAM resources may be limited, and it may be necessary to trade off memory size for performance, depending on the requirements of the specific LSTM design. The SRL (Shift Register Look Up Table) in Table 4.8 and Table 4.9 is a mode available in FPGAs whereby the LUT-RAM is configured as a shift register structure. This is more efficient, as it requires fewer cells and less routing than using individual DFFs to build shift registers. For the usage of DSP slices, LUT, and FF in each equalizer type, the biLSTM requires 64% DSP slices and 13% of LUT and FF, the deep CNN uses 30% DSP slices, 13% of LUT and 21% of FF, and the CDC needs 54% DSP slices and 1% of LUT and FF.

Third, in terms of throughput, the clock frequency is the maximum that each implementation can handle to comply with a zero-negative slack design. In this sense, the total throughput for the 16QAM modulation format is 66G, 60G, and 127G, for the biLSTM, deep CNN, and CDC block, respectively.

Type	Latency (μs)	Clock Frequency (MHz)	BRAM	SRL	DSP Slices	LUT	FF	Utilization (%) [DSP/LUT/FF]	Throughput ($Gbits/s$)	N^o FPGAs for 200G	N^o FPGAs for 400G [dual-carrier]	N^o FPGAs for 400G [56GBd]
biLSTM+CNN	33.4	270	164	109	1260	113532	224386	64.0/12.6/12.5	66	2.6	5.3	7.2
Deep CNN	19.9	244	0	125	582	118477	379829	29.6/13.2/21.1	60	1.4	2.7	3.7
CDC	1.1	524	0	1	1072	10441	5640	54.5/1.2/0.3	127	1.2	2.3	3.1

Table 4.8 VCK190 [8] Implementation.

Lastly, regarding the calculation of the number of equivalent FPGAs for a certain target throughput (T_{target}) from an experiment, it was considered the following equation:

$$N_{\text{FPGA}} = \frac{T_{\text{target}}}{T_{\text{a}}} * U_t, \tag{4.70}$$

where T_{a} is the throughput achieved after the NN design pipeline, and U_t is the maximum utilization after the NN design pipeline (both reported in Table 4.8). In the CNN+biLSTM case, for example, because the experiment was 16QAM single carrier transmission at both 34 GB per pol, the target throughput is 272 Gbit/200 G. So, considering the maximum utilization of 64% and the throughput achieved of 65.9Gbit by the NN equalizer, N_{FPGA} would be equal to 2.6 FPGAs.

In fact, the FPGA estimation was considered for three different cases:

1. 200G scenario: which is the scenario of the experiment in this section - 16QAM single carrier configurations at both 34GBd per pol (resulting in 272Gbit/200G).

2. 400G scenario: considering a dual carrier transmission instead of a single carrier. In this case, it just needs to scale the resources of 200G by a factor of 2.

3. 400G scenario: considering a 16QAM single carrier configurations with higher symbol rate equal to 56GBd per pol (resulting in 448Gbit, with 12% FEC overhead). In this case, it simply multiplies the 200G resources by $56^2/34^2 = 2.71$ because, given the increased symbol rate, the resources will grow approximately quadratically with the increase in symbol rate since the implementations were in the time domain.

For case 1, it was observed that 200G transmission can be achieved using an equivalent FPGA that has the same capacity as $\approx$ 3 FPGAs (VCK190) in the case of biLSTM, $\approx$ 2 FPGAs in a deep CNN case, and $\approx$ 1 FPGAs for CDC. For case 2, 400G with dual carrier transmission can be achieved using an equivalent FPGA that has the same capacity as $\approx$ 5 FPGAs (VCK190) in the case of biLSTM, $\approx$ 3 FPGAs in a deep CNN case, and $\approx$ 2 FPGAs for CDC. Finally, in case 3, because of the increase in symbol rate, much more hardware was needed. For the 400G with 56Gbd transmission case, one would need an equivalent FPGA that has the same capacity as $\approx$ 7 FPGAs (VCK190) in the case of biLSTM, $\approx$ 4 FPGAs in a deep CNN case, and $\approx$ 3 FPGAs for CDC. In all three cases, biLSTM used approximately 2.5 times more FPGA than required by the CDC implementation.

Unlike FPGAs, ASICs are application specific, and their digital circuitry contains permanently connected gates and FF in silicon; therefore, in ASIC design, there is no configurable block (such as DSP blocks). So, next, it is evaluated the approximations of the resource requirements and the performance in terms of throughput, clock frequency, and latency of the ASIC implementation by considering the VCK190 implementation without the usage of DSP slices. Table 4.9 contains information for the implementation of the biLSTM, the deep CNN, and the CDC equalizer on the FPGA with only LUT and FF. The biLSTM and CDC have a noticeably higher latency: 5.8 μs and 1.2 μs higher, respectively, compared to implementation with DSP slices detailed in Table 4.8. The lower clock frequency is also observed: 234 MHz for biLSTM and 246 MHz for the CDC, resulting in a lower throughput: 57G for biLSTM and 60G for the CDC. The degradation in throughput, latency, and clock frequency highlights the fact that DSP slices speed up the execution of signal processing functions. Especially in the CDC, when allowing implementation with DSP slices, the number of LUT and FF used is 40 times and 4.4 times less, respectively. Therefore, it is possible to see the degradation of performance in terms of the throughput of the CDC when the DSP slices are not used. However, in the case of deep CNN, the latency decreases by 2 μs, the clock frequency increases by 1 MHz, and the throughput remains unchanged. One can observe that in the deep CNN implementation with the DSP slices in Table 4.8, the number of DSP slices used is only about half that for the other two equalizers, and the number of LUT and FF is highest. Therefore, the removal of DSP slices did not affect the deep CNN because the processing time for LUT and FF was already the bottleneck in the previous implementation. Here, it is essential to recall that the clock frequency, which is essential to establishing the throughput, is chosen to guarantee zero negative timing slack. In this regard, since the routing and mapping are performed automatically by the Vivado platform, it can be observed that when restricting the software from using the DSP slices, longer paths are generated during the synthesis in the biLSTM and CDC cases. The longer paths cause the clock frequency to decrease to achieve the zero-negative slack level. In contrast, the deep CNN's paths when deploying the DSP slices are already long due to the logic implementation and synthesis, and, therefore, there is no significant variation in clock frequency after removing the DSP slice in that case.

Type	Latency (μs)	Clock Frequency (MHz)	BRAM	SRL	LUT	FF	Utilization (%) [LUT/FF]	Throughput ($Gbits/s$)	N^o FPGAs for 200G	N^o FPGAs for 400G [dual-carrier]	N^o FPGAs for 400G [56GBd]
biLSTM+CNN	39.2	234	164	163	566070	249763	62.9/13.9	57	3.0	6.0	8.1
Deep CNN	17	245	0	162	300426	399868	33.4/22.3	60	1.5	3.0	4.1
CDC	2.3	246	0	1	418534	24968	46.5/1.4	60	2.1	4.2	5.7

Table 4.9 Implementation where all multiplications are done using LUT and FF.

Regarding the utilization of LUT and FF: the biLSTM uses 62.9% of LUT and 13.9% of FF, the deep CNN uses 33.4% of LUT and 22.3% of FF, and the CDC uses 46.5% of LUT and 1.4% of FF. The utilization of LUT for all three equalizers is considerably increased compared to the previous case (standard FPGA implementation). As the number of LUTs and FFs increases, the equivalent number of FPGAs used to represent the biLSTM and the CDC equalization also grows.

For the same three cases discussed previously, it can be observed the following. For case 1, 200G transmission can be achieved using an equivalent FPGA that has the same capacity as $\approx$ 3 FPGAs (VCK190) in the case of biLSTM, $\approx$ 2 FPGAs in a deep CNN case, and $\approx$ 2 FPGAs for CDC. For case 2, 400G with dual carrier transmission can be achieved using an equivalent FPGA that has the same capacity as $\approx$ 6 FPGAs (VCK190) in the case of biLSTM, $\approx$ 3 FPGAs in a deep CNN case, and $\approx$ 4 FPGAs for CDC. Eventually, in case 3 with a 16 QAM 56Gbd transmission, one would need an equivalent FPGA that has the same capacity as $\approx$ 8 FPGAs (VCK190) in the case of biLSTM, $\approx$ 4 FPGAs in a deep CNN case, and $\approx$ 6 FPGAs for CDC. In all three cases, biLSTM used approximately 1.5 times more FPGA than required by the CDC implementation.

Finally, note that in this study, it was established the approximate resources and performance of the equalizers implemented on ASIC by excluding the DSP slices. However, this is still not an optimized realization: in ASIC implementation, the number of resources used needs to be further optimized to reduce the utilization, increase the clock frequency, and enable high-speed processing.

Chapter 5

The Flexibility Consideration for Designing Neural Networks-based Equalizers

This chapter presents an extremely alluring technical answer for the fast adaptability for varied transmission setups required in high-speed optical communications DSP-related solutions. One of the primary barriers to neural network equalizers being used in fiber-optic communication systems is their limited operational region. Here, it is proposed the transfer learning (TL) technique to solve this issue by substantially improving the performance of the neural network channel equalizers in terms of training complexity. With the proposed TL for mitigation of fiber nonlinear effects, the solution becomes flexible for different launch powers, modulation formats, and symbol rates after a short period of retraining.

5.1 Transfer learning in optical fiber communications

The TL can be defined as a system's ability to identify and adapt knowledge acquired in previous tasks to new tasks [214].

Recent publications on the application of TL in optical communication focus on optical network tools [215–220]. A few works also addressed the nonlinearity mitigation issue [221–223]. However, the TL in optical networks has been mainly used for optical signal-to-noise ratio (OSNR) monitoring. In [217], this application was introduced using an artificial NN-based TL approach to accurately predict the quality of the transmission of different optical networks without re-training NN models from scratch. In that study, the source domain was a 4×80 km (4 spans) large effective area fiber (LEAF) link using QPSK modulation. The target domain was the same system but with a different number of spans (propagation distance) and different modulations formats (4$\times$80 km LEAF with 16-QAM; 2$\times$80 km LEAF with 16-QAM; and 3$\times$80 km dispersion-shifted fiber with QPSK). The results showed that when using the TL, just 2% of the original training dataset size was enough to calibrate the NN for the new target domain. More recently, in Ref. [216], the experimental demonstration of the application of TL for joint OSNR monitoring and modulation format identification from 64-QAM signals was presented. It was shown that by implementing the TL from simulation to experiment, the number of training samples and epochs needed for the same prediction quality was reduced by 24.5% and 44.4%, respectively. Another recent application of TL was in the spectrum optimization problem for resource reservation [220]. To predict a spectrum defragmentation time, the pre-trained NN model for a source domain (having a 6-node topology) was transferred and trained again using the data from the target domain (the NSFNet with 14 nodes). It was shown that by using the TL technique, the proportion of affected services was reduced, the overall likelihood of resource reservation failure was diminished, and the spectrum resource utilization was improved.

Only a few works have addressed the topic of TL for nonlinearity mitigation, and they focus on short-haul direct detection systems. In [222] the successful transfer of the knowledge for the links with different bit rates and fiber lengths was demonstrated. Both feedforward and recurrent NNs were tested for the TL application: about 90% (feed-forward) and 87.5% (recurrent) reduction in epochs were achieved, and 62.5% (feed-forward) and 53.8% (recurrent) reduction in training symbols were demonstrated. Another work in direct detection, Ref. [221], applied the TL from 5 dBm launch power to other powers (ranging from -7 dBm to 9 dBm) and from one transmission distance (640 km) to other ones (from 80 km up to 800 km). The experimental results showed that the iterations of TL constitute approximately one-fourth of the full NN training iterations. Additionally, the TL did not result in a performance penalty in a five-channel transmission when transferring the learned features from training just the middle channel to the four other channels.

Finally, the TL in coherent optical systems was first investigated in Ref. [223]. In that work, the authors applied the TL for different launch powers but provided a very brief explanation of the technique. Note, however, that compared to the previous works, this chapter explicitly explains how one can successfully transfer the learned features from nonlinear optical channels using the NN equalizers, addressing the coherent transmission systems. Also, this chapter presents a novel and broad description of how the TL can be efficiently used to realize flexible NN equalizers for adaptation to changes in launch power, modulation format, symbol rate, and fiber setup.

5.2 Application of transfer learning to nonlinearity mitigation

First, it is important to identify the *domain* and the *task* notions for optical channel equalization. The domain consists of a feature space Y which is an array of time-domain vectors (the memory window vectors). Each window vector consists of the real and imaginary parts of the received symbol (in each polarization) at time-step k and its $2N$ neighboring (past and future) symbols. The task consists of two parts: i) the label space X, which is the set of real and imaginary parts of the transmitted symbols, and ii) the channel posterior function f, which defines the conditional probabilistic distribution $p(X|Y)$. Several machine learning models can be used to learn the function f. Herein, this work uses the proposed NN-based equalizer CRNN or the "CNN+biLSTM" equalizer; described in Sec. 3.2 at Fig. 3.9. This architecture was chosen because it delivers the best performance for impairments mitigation in long-haul coherent optical systems when compared to several alternative NN structures, provided that the computational complexity is not restricted, as shown in Chapters 3 and 4.

Having identified the domain and task in the sense of optical channel equalization, it is now time to explain “what to transfer”, i.e., which information can be transferred between different domains and tasks. The pass-averaged Manakov equation [96] describes the averaged evolution of the slowly varying complex-valued envelopes of the electric field in an optical fiber. From the Manakov equation, it can be stated that the channel likelihood, or the conditional probability of the received signal given the transmitted signal, $p(Y|X)$, is affected by the launch power, symbol rate, fiber type characterized by its attenuation coefficient α, the dispersion coefficient β_2, the Kerr nonlinear coefficient γ, and the link setup, characterized, e.g., by the number of spans N_s, the span length L, the amplifier noise figure NF. In a nutshell, the NN-model that learns f seeks to grasp the inverse correlation between the transmitted and received symbols, hence the aforementioned parameters are also important for determining the posterior function f.

As far as the propagation within both Task A (the source) and Task B (the target) is governed by the Manakov equation, the TL boosts the training performance of the Task B model from the Task A model. The TL strategy that fits this goal is the *inductive TL* [224]. In the inductive TL, the source and target tasks are different but related, and the TL seeks to strengthen the target task by exploiting the source domain’s inductive biases. The transductive and unsupervised TLs are the two other TL techniques [225]. The definition of unsupervised TL is analogous to inductive TL; however, since no labels are available in this case, it is infeasible to solve the purposed regression problem of channel equalization using this strategy. In the case of transductive TL, the source and target tasks must be identical but, as mentioned before, in this case, the task will change as the transmission parameters change. As a result of these constraints, transductive and unsupervised TLs are inapplicable to the current goal of nonlinearity mitigation in optical channel equalization.

From the model perspective, this TL uses the parameter control strategy [225]. In this case, the attribute priors, or probabilistic distribution parameters of the signal features, can be learned from the source domain and then be used to ease the learning of the target equalizer model. The parameters of a model reflect the knowledge learned by the model. As a consequence, the knowledge can be passed by sharing the parameters between different task models.

Note that the interpretability of composite neural network structures is still, largely, a matter of debate in the machine learning community [226–228]. For the CRNN-based equalizer (CNN+biLSTM) [151], both the LSTM layer and the CNN layer jointly contribute to the nonlinear mitigation, but they do this in a complementary manner under their respective "strengths". However, some insights can be readily earned when analyzing the TL technique application to the considered CNN+biLSTM equalizer. In this chapter, to save the training complexity during the transfer process, it did a standard procedure to test which layers are due to be retrained and which ones can be kept frozen. During the tests, it was observed that whether freezing/retraining a layer depends on the characteristics of the source and target domains. The implemented TL procedure is summarized in Fig. 5.1. First, train all the layers in the model using Domain/Task A (the source, top panel). It was marked the NN equalizer layers that were (re)trained with a dark gray color, and the layers with the fixed weights with a light gray color. Next, transfer the learned weights to the model for a new Domain/Task B (the target): three possible procedures for the transfer can be executed, as shown in the middle panel of Fig. 5.1. In the first case (case (a), the top inset in the middle panel), all the weights are learned again for the new Task B. Such an approach is recommended when there are considerable changes in nonlinearity and dispersion simultaneously (e.g., when having a change in both the symbol rate and power). In the second case marked with (b), the middle part of the inset in the middle panel, the convolutional layer was frozen, and only the weights in the biLSTM and output layers are trainable. This type of transfer (without losing the performance) obtains a reduction in training complexity when the channel memory changes noticeably, but the nonlinearity is still similar for both Tasks A and B. Such a scenario exists when, for instance, increasing or decreasing the symbol rate for Task B, but keeping the same optical launch power for both Tasks A and B. Finally, in the third case (c) – the lower part of the middle inset, – the convolutional layer is trainable, but the biLSTM and output layers are frozen. This strategy evidently reduces the training complexity as well. This TL type can be used when the memory of the system is similar for both Tasks (e.g., the symbol rate is kept

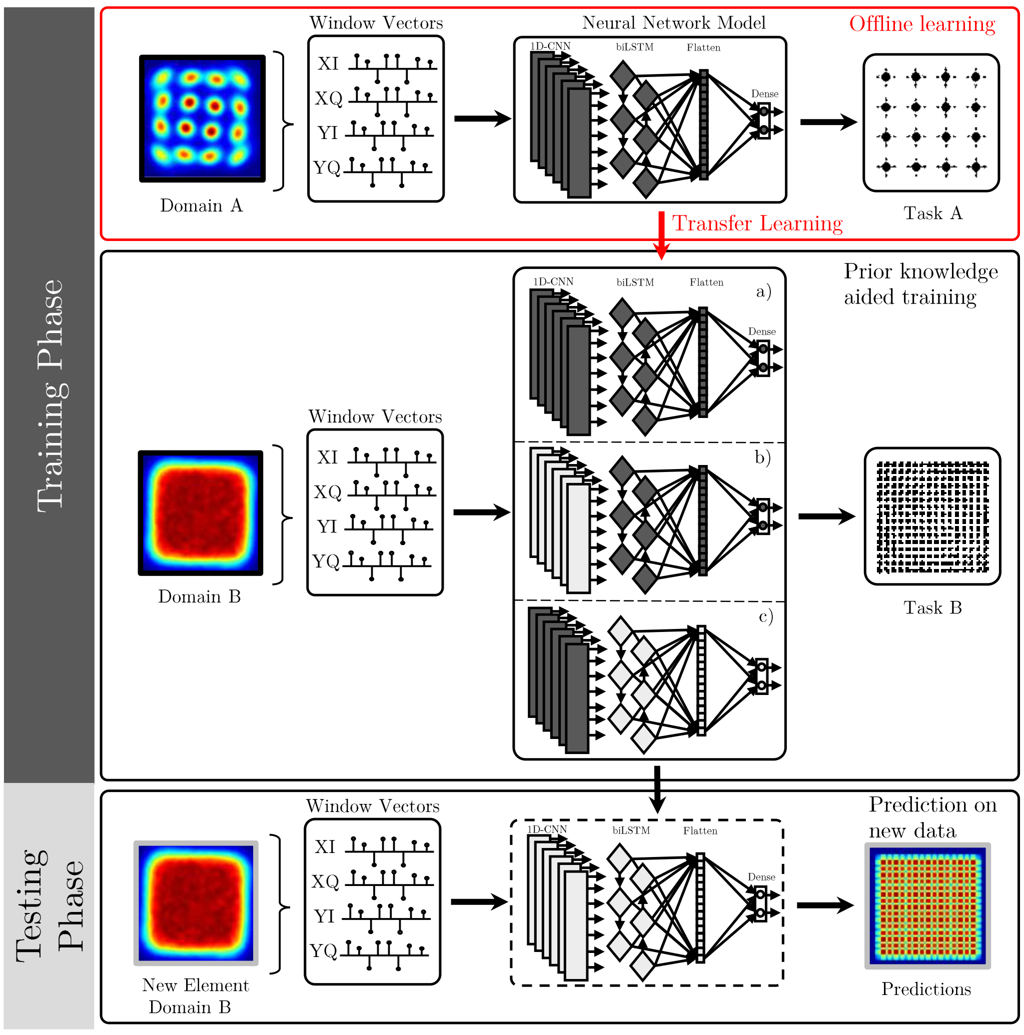


Fig. 5.1 Schematics of transferring the learning for the optical communication system with the NN-based equalizer (CNN+biLSTM). The leftmost subfigures display the received constellations, while the equalized constellations are shown in the rightmost ones. XI, XQ, YI, YQ refer to the I and Q components of X and Y polarizations. The NN elements that are getting trained/retrained are marked with dark gray, while the fixed NN elements are highlighted with light gray. The top panel represents the offline learning to train the NN model to a certain Domain/Task A. Three possible strategies of TL for the new Domain/Task B are shown in the middle panel: a) the model is retrained completely to the new Task B with the initial weights coming from the model trained on Task A; b) only the biLSTM and output layers are retrained to the new Task B and the convolutional layer is frozen with the weights coming from Task A; c) only the CNN layer is retrained to the new Task B and the biLSTM and output layers are frozen with the weights coming from Task A. In the lowest panel, it is evaluated the performance of the NN with a completely new dataset for Domain/Task B, keeping all layers frozen.

the same), but the nonlinearity for Task B changes, e.g., when changing the launch power. Finally, when evaluating the performance of the new model attributed to Task B, all weights were frozen: this case is indicated in the lower panel of Fig. 5.1, entitled "Testing Phase". With this in mind, it is now possible to unveil some physical interpretations/motivations for the choice of the layers to be retrained. The reason for retraining only the CNN layer when the power changes, is that the hidden weights connecting different cells in the LSTM layer should remain fixed since the correlation between symbols remains almost the same. On the other hand, the kernel weights of each filter in the CNN layer need to be updated to account for the difference in the intensity of the nonlinearity. A similar logic can be followed when the memory of the transmission system changes. In this case, only the LSTM needs to be retrained since the weights between cells must learn the new correlation of the domain while the CNN can be kept frozen. The next section will analyze how efficiently it is to transfer the learned network parameters from Domain/Task A to Domain/Task B for different modifications in the transmission parameters.

5.3 Numerical Setup and Neural Network model

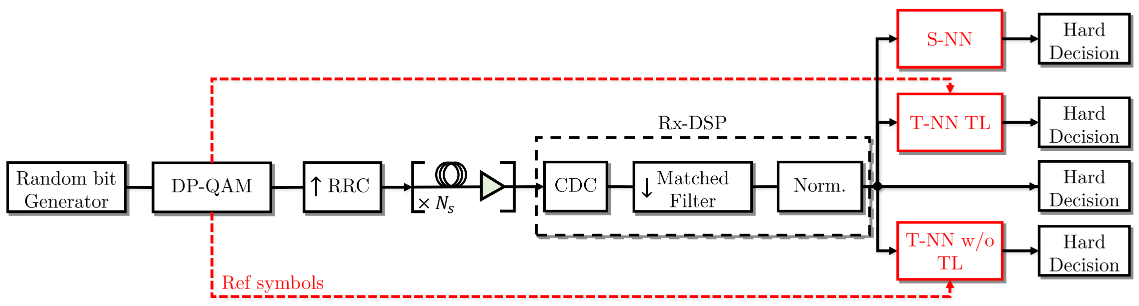


Fig. 5.2 Scheme of the numerical setup (single channel) considered in this chapter, where the red elements indicate the 3 possible different NN implementations evaluated. The red arrows show that it was used the transmitted symbols for the regression retraining but, since the S-NN is not retrained, it does not require using any transmitted symbols of the target domain.

This section explicitly evaluates the efficiency of the TL technique in terms of reducing the size of the training dataset and the number of epochs needed to reach acceptable accuracy of signal recovery. For comparison purposes, it was used four key reference curves. i) only linear equalization is applied, where the respective Q-factor level is independent of the number of epochs, as no training occurs. This case is labeled as "w/o NN" and plotted with an orange straight line. The efficiency of the NN equalizer is measured against this curve. ii) The next reference curve is used to demonstrate the impact of changes in transmission parameters on the performance of the NN-based equalizer. In this case, the purple curve (labeled as the source NN, S-NN) shows the Q-factor when the NN equalizer is trained with the source Domain/Task A data only and tested on the target Domain/Task B without retraining. iii) It also evaluated the impact of training the NN using the data from the target Domain/Task B without using TL. This curve is labeled as “T-NN w/o TL”, (T-NN means the target NN). In this case, the weights are initialized randomly, which corresponds to the traditional training of the NN equalizers. iv) Finally, this work included the approach proposed in this chapter, corresponding to transferring the learned parameters from the source Domain/Task A. It was denoted as “T-NN TL x %”, where the “x” value represents the percentage of data used to train when compared to the T-NN w/o TL.

The following sections address the details of the simulation setup and how the NN was trained. The results of the TL will also be shown for the changes in fiber type (using the TL of Fig. 5.1 (a)), symbol rate (using the TL of Fig. 5.1 (b)), and launch power and modulation format (using the TL of Fig. 5.1 (c)). This section ended with a short discussion of the TL limitations for this task.

To illustrate the effect of the application of the proposed TL to the NN-based optical channel equalizers, it was numerically simulated the dual-polarization (DP) transmission of a single-channel signal at 34.4 / 45 / 65 / 85 GBd. The signal is pre-shaped with a root-raised cosine (RRC) filter with 0.1 roll-off at a sampling rate of 8 samples per symbol. In addition, the signal had four possible modulation formats: 16 / 32 / 64 / 128-QAM. Modulation formats up to 128QAM were evaluated, since recent publications showed performance analysis for these modulation formats using simulation as well as experiment setups [111, 229, 230]. It was considered the following two test cases: (i) transmission over an optical link consisting of 9×50 km TWC spans; and (ii) transmission over 18×50 km SSMF spans. The optical signal propagation along the fiber was simulated by solving the Manakov equation via split-step Fourier method (SSFM) [231] with the resolution of 1 km per step. The considered parameters of the TWC fiber are: the attenuation parameter $\alpha = 0.23$ dB/km, the dispersion coefficient $D = 2.8$ ps/(nm·km), and the effective nonlinearity coefficient $\gamma = 2.5$ (W·km)$^{-1}$. The SSMF parameters are: $\alpha = 0.2$ dB/km, $D = 17$ ps/(nm·km), and $\gamma = 1.2$ (W·km)$^{-1}$. Every span was followed by an optical amplifier with the noise figure NF = 4.5 dB, which fully compensated fiber losses and added amplified spontaneous emission (ASE) noise. At the receiver, a standard Rx-DSP was used. It includes the full electronic chromatic dispersion compensation (CDC) using a frequency-domain equalizer, the application of a matched filter, and the downsampling to the symbol rate. Finally, the received symbols were normalized (by phase and amplitude) to the transmitted ones. No other transceiver distortions were considered.

After the Rx-DSP, the bit error rate (BER) was estimated using the transmitted symbols, received soft symbols, and hard decisions after equalization, addressing the four cases depicted on the right side of Fig. 5.2. The NN input mini-batch shape can be defined by three dimensions [151]: $(B, M, 4)$. B is the mini-batch size, M is the memory size defined through the number of neighbors N as $M = 2N + 1$, and 4 is the number of features for each symbol, referring to the real and imaginary parts of two polarization components. The output target is to recover the real and imaginary parts of the k-th symbol in one of the polarization, so the shape of the NN output batch can be expressed as $(B, 2)$.

In general, for the CNN+biLSTM NN considered in this section, it was incorporated the mean square error (MSE) loss estimator and the classical Adam algorithm for the stochastic optimization step with the default learning rate set equal to 0.001 [201]. The training was carried out for up to 200 epochs with a batch size of 1000, which has proven to be high enough to show the convergence for the transmission scenarios. Additionally, the total dataset used was composed of 2^{18} symbols for the training dataset and of 2^{18} independently generated symbols for the testing phase. The training dataset was shuffled at the beginning of every epoch to avoid overfitting caused by learning the connections between the neighboring training pairs [108]. All datasets were generated using the Mersenne twister generator [200] with different random seeds, which guarantees a cross-correlation below 0.004 between the training and testing datasets, meaning that the symbols are virtually independent.

Finally, since the goal of this chapter is to demonstrate the efficiency of the TL technique, it was used the same best-performing CNN+biLSTM architecture with 244 filters, kernel size 10, and 226 hidden units in the LSTM cell, as in Ref. [151]. Also, the number of taps used was $N = 40$: this is the maximal memory size estimation for all scenario changes investigated in this work. Note that the memory effect is important because, even though it is compensated by the chromatic dispersion compensator electronically, it is still needed to mitigate also the impact of the coupling between the nonlinearity and the chromatic dispersion along with optical fiber transmission. To unroll this coupling efficiently, it is needed the information from the neighboring symbols. Note that this is true in other perturbation techniques [1], where the triplets are used to enhance the signal after the chromatic dispersion compensation. This memory guarantees no artificial degradation of the NN performance for all the cases considered. Also, it is important to highlight that the objective of the approach proposed in this study was to produce a NN complex enough to deal with the different levels of nonlinearity but without needing to increase its number of hyperparameters (e.g., the number of layers, neurons, filters, etc.). As it was explained in Ref. [151], increasing the NN parameters' number above the NN's capacity would cause overfitting, therefore limiting the achievable performance improvement. Clearly, in the ideal scenario, the neural network topology/hyperparameters should be optimized to the new configuration when changing the transmission setup. However, in this work, the equalizer is kept unchanged, although this approach requires higher complexity to fully cope with a strong nonlinear scenario.

5.4 Transfer learning for different scenarios of launch power

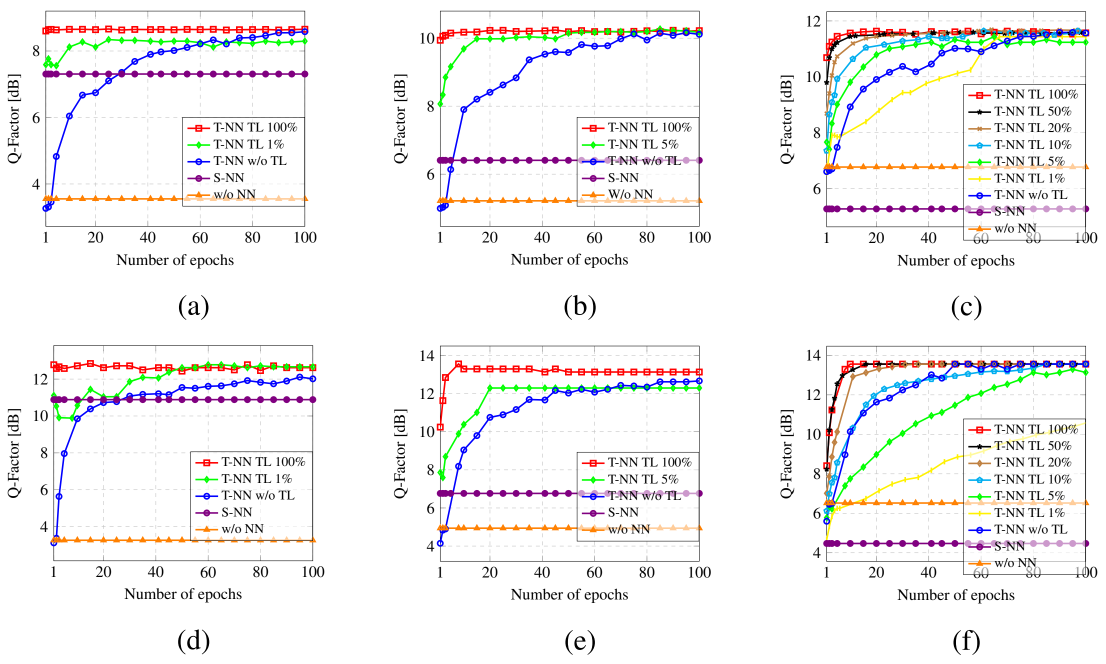


Fig. 5.3 Transfering the learning between the launch powers. Case I: from 8 dBm to (a) 7 dBm, (b) 6 dBm, (c) 5 dBm, using 18×50 km SSMF fiber link and DP-16-QAM 34.4 GBd. Case II: from 5 dBm to (d) 4 dBm, (e) 3 dBm, (f) 2 dBm, using 9×50 km TWC fiber link and DP-16-QAM 34.4 GBd.

This analysis begins by transferring the learned parameters from a system that has been trained with a specific launch power to a system that operates at different power levels. Both the SSMF and TWC fiber types are considered. Fig. 5.3 presents the results of TL between different powers for two cases, considering the SSMF and TWC separately. The first case (Case I) compares the system performance in terms of Q-factor when the source dataset consists of a 16-QAM signal with a launch power of 8 dBm to the three target datasets with different optical power: (a) 7 dBm, (b) 6 dBm, and (c) 5 dBm. The systems keep the same DP-16-QAM at 34.4 GBd, and the same transmission parameters (18×50 km SSMF fiber link). Some important conclusions can be drawn from this figure. First, as expected, when moving from the source power (8 dBm) to the target power (5 dBm), the S-NN's output degraded showing even worse performance than the reference case (when only the linear equalization was applied). Since the conditional probability defined by the models did not generalize for different powers, this observation reveals that the NNs, by default, are not flexible enough to be used when the launch power changes. In practice, such changes in launch power may be necessitated by the addition of neighboring channels to the network, which will result in a reduction in the optimum launch power owing to XPM effects. However, the TL usage allows the NN to quickly reconfigure, and the latter then provides an efficient output in the new scenario. By retraining only the convolutional layers, it is required 1 epoch, 4 epochs, and 10 epochs for the 7 dBm, 6 dBm, and 5 dBm scenarios, respectively, to achieve the best Q-factor. This translates into an approximate reduction in the number of epochs to 99%, 95%, and 88%, respectively, compared to achieving the same Q-factor when trained from scratch. Another advantage of the TL is that the size of the required training dataset can be reduced without compromising the equalizer's efficiency. Case I show that, depending on the difference between the launch power considered in the source and target links, it can be saved up to 99 % of the amount of training data. Note that this work considered the analysis when training with different dataset sizes (1/ 5/ 10/ 20/ 50/ 100%), as shown in Fig. 5.3 (c) and (f). However, for clarity, the rest of this chapter only reported the TL for the cases where it was saved the most epochs using 100% of the dataset, and the case in which it was trained with the least amount of data while achieving the same or better Q-factor as in the case without the TL.

Case II illustrates how general the findings are. It was checked a source dataset with DP-16-QAM 34.4 GBd considering a launch power of 5 dBm in a 9×50 km TWC link and transferred to the target sets of (d) 4 dB, (e) 3 dBm, and (f) 2 dBm launch powers, with the same setup. The use of TL was helpful in this case as well, and the results are given in Fig 5.3. When switching to (d) 4 dBm, (e) 3 dBm, and (f) 2 dBm, the number of epochs necessary to achieve the maximum Q-factor decreased approximately by 99%, 90%, and 80%, respectively. Also, the re-training process required fewer data: 99%, 95%, and 95%, respectively.

At this stage, it is pertinent to address three questions: i) what happens if the launch power is reduced further; ii) can the knowledge be transferred from lower to higher launch powers; and iii) would the TL still work in the presence of a transceiver noise. To answer the first question, it was tested the TL for a wide range of launch power levels (from 8 to -8 dBm) and found that it performs quite well as long as the NN-based equalizer still works, i.e., for the powers where it produces a non-zero improvement in symbol recovery. The TL works well in the nonlinear fiber transmission regime because the NN reverses the Kerr nonlinearity and uncompensated dispersion-related effects. However, the equalizer's effectiveness degrades in the linear regime. This is because the NN equalizer itself cannot recover impairments in the linear regime of the simulated data, since they come from a stochastic noise-related effect emerging from the amplifiers. Note that in the simulations, it was considered the ideal components, e.g., DAC/ADC, etc. Nonetheless, despite the fact that it used such high power, 7/ 6/ 5 dBm, the TL works efficiently in the whole range of powers down to the optimal launch power.

Table 5.1 Dependence of the TL performance on the transfer direction, for the case where only the launch powers were changed from the source to target datasets.

Fiber	Scenario	Max Q-factor	Epochs required
SSMF	TL 8 dBm $\rightarrow$ 5dBm	11.56	<10
	TL 2 dBm $\rightarrow$ 5dBm	11.56	>100
	w/o TL	11.56	>80
TWC	TL 5 dBm $\rightarrow$ 2dB	13.56	<10
	TL -1 dBm $\rightarrow$ 2dBm	13.56	>100
	w/o TL	13.56	>45

Question ii) is relevant because it stresses the need of comprehending the underlying physical effects. After analyzing the effectiveness of transferring the knowledge from smaller to higher launch powers and vice-versa, it was found that the *TL is more effective when the training is carried out at higher launch powers and the TL occurs from higher to lower launch powers*. The results regarding the TL direction effects are summarized in Table 5.1. The explanation for the TL direction dependence is that the NN equalizer reverses the nonlinear effects, and the latter intensifies with the growth of launch power. The source domain NN trained with higher launch power possesses a better "knowledge" of nonlinearity, the NN learns the nonlinear channel function well. Therefore, the high power regime has more useful information about the channel function, which can be readily adapted for target systems with lower launch powers, compared to the situation when the TL is employed in the opposite direction. To illustrate this phenomenon, consider the case where a polynomial Volterra equalizer is used. It is evident that if training it for a higher nonlinearity scenario, it will need more coefficients to get a satisfactory result than if training it for a lower nonlinearity [232]. The TL follows the same logic whereby training the source NN with the worst-case scenario, it is easier for NN architectures to discard some of the source NN's elements that are not meaningful in the lower nonlinear regime than it is to learn new ones from scratch.

Finally, Fig. 5.4 was added to answer the third question of whether the TL would still work with the addition of component-generated noise. To generate the data with noise, it was

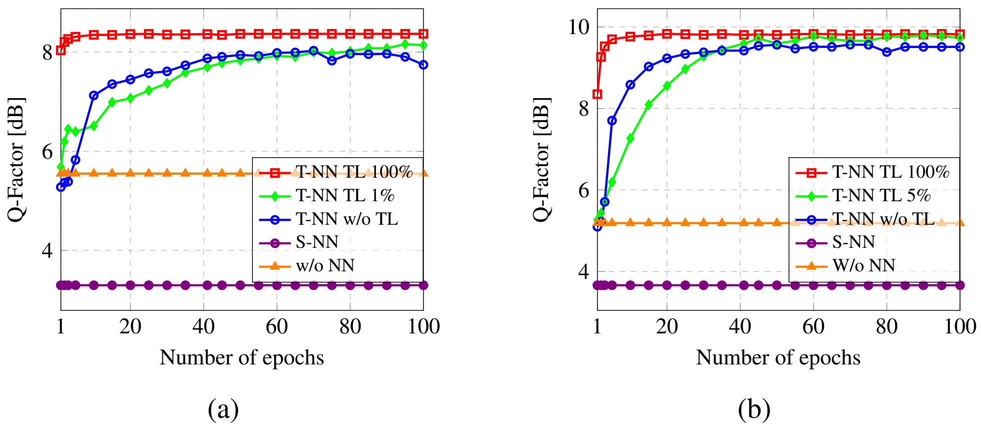


Fig. 5.4 Launch power transfer learning from a dataset without transmitter noise to the dataset with transmitter noise. (a) SSMF (from 8 dBm to 5 dBm); (b) TWC (from 5 dBm to 2 dBm).

assumed realistic transceivers affected by electrical noise with back-to-back SNR given by:

$$\mathrm{SNR[dB]} = -0.175R + 30, \tag{5.1}$$

where R is the symbol rate[1]. This equation is an approximate fit to the experimentally measured values described in [233] and the noise, modeled as an additive white Gaussian, is contributed equally by the transmitter and receiver. Fig. 5.4 (a) refers to the case of 18×50 km SSMF transferring from the source 8 dBm (without transceiver noise) to the target 5 dBm (with transceiver noise); Fig. 5.4 (b) refers to the 9×50 km TWC transferring from the source 5 dBm (without transceiver noise) to the target 2 dBm (with transceiver noise). The analysis of the results given in Fig. 5.4 reveals that the system performance was slightly impacted by the increased noise level, but the TL continued to work with the same effectiveness. In Fig. 5.4 (a), the reduction of 90% in epochs and 99% in the training dataset was observed. By the same token, Fig. 5.4 (b) shows a decrease of 90% in epochs and 95% of the training dataset. Note that the case depicted in Fig. 5.4 was selected to highlight that the transmitter noise in the target domain does not harm the TL performance. It was also tested by adding noise to both the source and target domains, with the TL still showing quite good performance in this case as well.

[1]This equation was derived by the authors of Ref. [233] for a reference system and distributed within TRANSNET Project members.

5.5 Transfer learning for different modulation formats

This section analyzes the impact of changing the modulation format on the NN equalizer's performance. The source dataset modulation format is 16-QAM, whereas the target modulation formats are 32-QAM, 64-QAM, and 128-QAM. First, the same launch power is kept independent of the selected modulation format. Fig. 5.5 shows the results for the two cases studied where only the modulation format changes: Case I (Fig. 5.5-a, b, c) with the SSMF setup at a launch power of 4 dBm, and Case II (Fig. 5.5-d, e, f) with the TWC setup at a launch power of 2 dBm.

In this scenario, only the convolutional layers were retrained in the case of T-NN with TL. From the results obtained, one can infer that both the T-NN with TL and S-NN can be successfully used with different modulation formats, as can be readily seen from Fig. 5.5. This means that it is not needed to retrain the model if changing only the constellation size of the modulation format. These results also demonstrate that the nonlinear propagation channel law is almost unaffected by changing the modulation format (from the 16-QAM to a higher-order one) when the power level remains the same. Thus, the NN equalizer, which is reverting the channel nonlinear effects, continues to function well for other modulation formats. This is in stark contrast to the case of classification equalizers (classifiers) [4, 76, 234], because the latter incorporates the decision boundaries in the NN structure itself. For the classifiers, the S-NN will not work with the new target task since its output stage does not capture the different symbol alphabet. For the NN structure used in this work (the CNN+biLSTM with regression), the channel reversion capability of the equalizer is independent of the modulation format. In contrast, for the classification task, the number of neurons in the last layer is defined by the number of constellation points. So, for the classification model, the output layer with an updated dimensionality should be retrained to correctly identify the new constellation points. Note that the small performance deviation of the T-NN with and without TL, shown in Fig. 5.5, is a consequence of the particular weight initialization. Finally, TL direction in the case of the modulation format modification is irrelevant to the TL performance inasmuch as the regression-based NN functionality is not affected by the format changes.

The performance of TL when it simultaneously changes the modulation format and launch power is also evaluated in Fig. 5.5. The source dataset in Case III was the transmission of 16-QAM signals with a launch power of 8 dBm in the SSMF link, and transferring the learned parameters to the target having 4 dBm launch power and (g) 32-QAM, (h) 64-QAM, (i) 128-QAM modulation formats. The source dataset of Case IV was the transmission of 16-QAM signals with a launch power of 5 dBm in the TWC fiber link, and transferring the learned parameters to the target with 2 dBm launch power and (j) 32-QAM, (k) 64-QAM, (l)

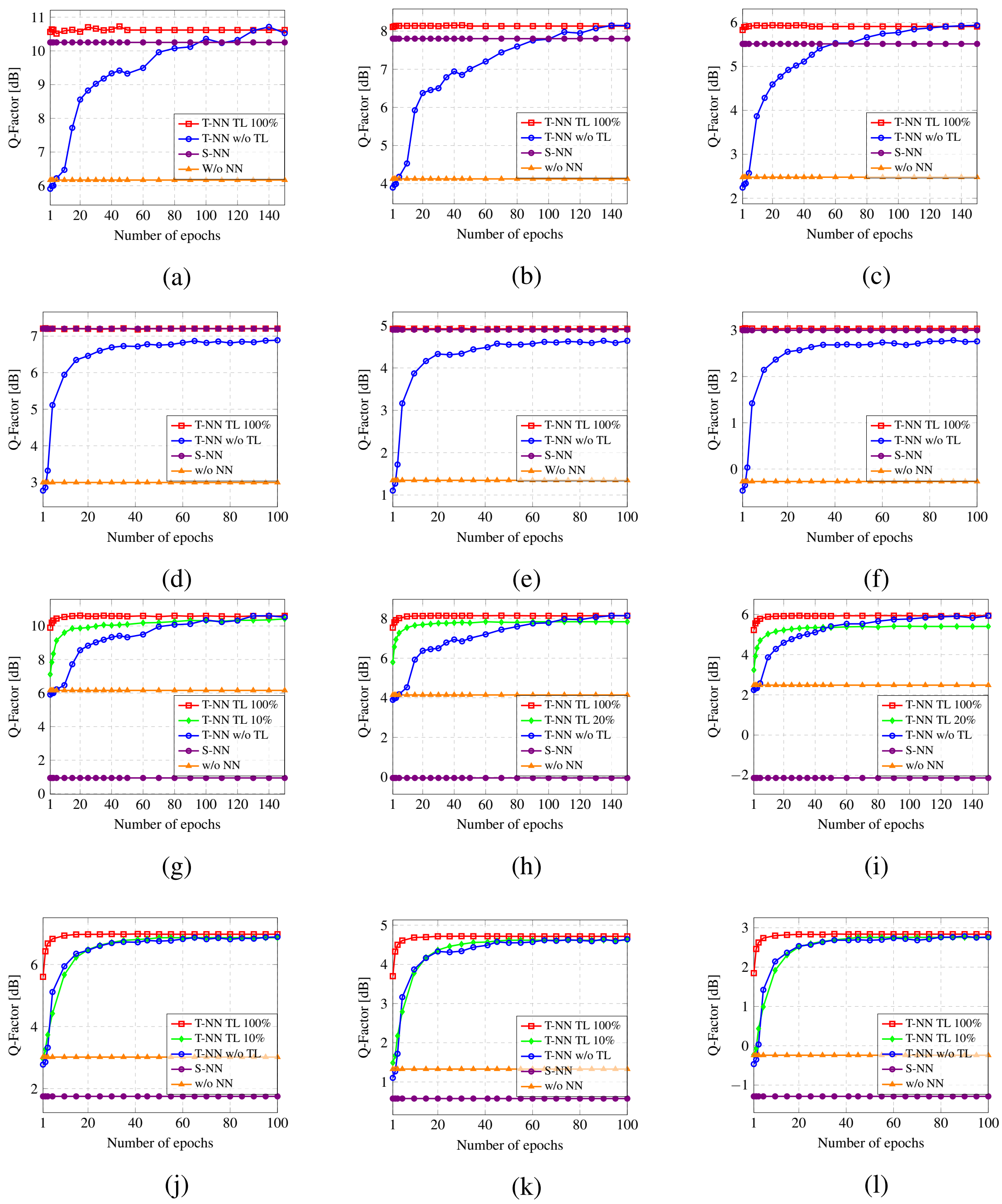


Fig. 5.5 Transferring the learning between modulation formats. Case I: from 16-QAM to (a) 32-QAM, (b) 64-QAM, (c) 128-QAM, using 18×50 km SSMF fiber link and 4 dBm 34.4 GBd signals. Case II: from 16-QAM to (d) 32-QAM, (e) 64-QAM, (f) 128-QAM, using 9×50 km TWC fiber link and 2 dBm at 34.4 GBd signals. Case III: from 8 dBm / 16-QAM to (g) 4 dBm / 32-QAM, (h) 4 dBm / 64-QAM, (i) 4 dBm / 128-QAM, using 18×50 km SSMF fiber link and 34.4 GBd. Case IV: 5 dBm / 16-QAM to (j) 2 dBm / 32-QAM, (k) 2 dBm / 64-QAM, (l) 2 dBm / 128-QAM, using 9×50 km TWC fiber link and 34.4 GBd.

128-QAM modulation formats. From the analysis of Cases III and IV in Fig. 5.5, one can notice a reduction of up to 95% in epochs and 90% in the training dataset for the SSMF case and a decrease of up to 85% in epochs and 90% in the training dataset for the TWC case. As expected, these figures are close to the ones obtained when evaluating the TL between different launch powers, Subsec. 5.4.

5.6 Transfer learning for different symbol rates

This section evaluates the functionality of TL when only the symbol rate is changed. By changing the symbol rate and keeping the remaining system parameters constant, it effectively changes how the neighboring symbols interact with each other. In other words, this change impacts the channel memory. As the channel memory is primarily handled by the biLSTM part of the CNN+biLSTM equalizer, in this section it is used the TL defined in Fig. 5.1, middle panel, inset (b). It retrained the biLSTM and output weights, but keep the convolutional weights frozen. Note that when changing the symbol rate, the effective nonlinearity level is also affected. Nonetheless, it is observed that no retraining is needed in the CNN layer because the features that were extracted by it (embedded in the values of its kernels), do represent the case of higher nonlinearity, and those can be fine-tuned to the lower nonlinearity by just adjusting the LSTM layer input weights. On the other hand, the weights between the LSTM cells, which cater to the memory effects among other possible representations, had to be retrained because the memory effect changes when the symbol rate does. The source dataset is the 34.4 GBd signal, and the target symbol rates are 45 GBd, 65 GBd, and 85 GBd. It was considered the SSMF and TWC link cases with 16-QAM modulation format and 6 dBm and 2 dBm launch powers, respectively. Fig. 5.6 depicts the results for the SSMF and TWC fiber links, referred to as Case I and Case II, respectively. The analysis of this figure shows that, by significantly changing the symbol rate with respect to the source, the S-NN performance can degrade below the reference system (w/o NN).

In Case I, when moving to (a) 45 GBd, (b) 65 GBd, and (c) 85 GBd, the number of necessary epochs decreased by 99%, 95%, and 81%, respectively, for the SSMF link case. Furthermore, the re-training process needs much fewer data, with a reduction of up to 99% of the required training data for the three symbol rates considered. The number of required epochs for the TWC fiber link (Case II), decreased by 92%, 73%, and 75%, respectively, when switching to (d) 45 GBd, (e) 65 GBd, and (f) 85 GBd. So, the retraining phase requires fewer data: 95%, 95%, and 90%, respectively. This demonstrates the potential of TL when adjusting the NN to different symbol rates, which is a very important feature when considering the current commercial transponders which can operate in a very wide

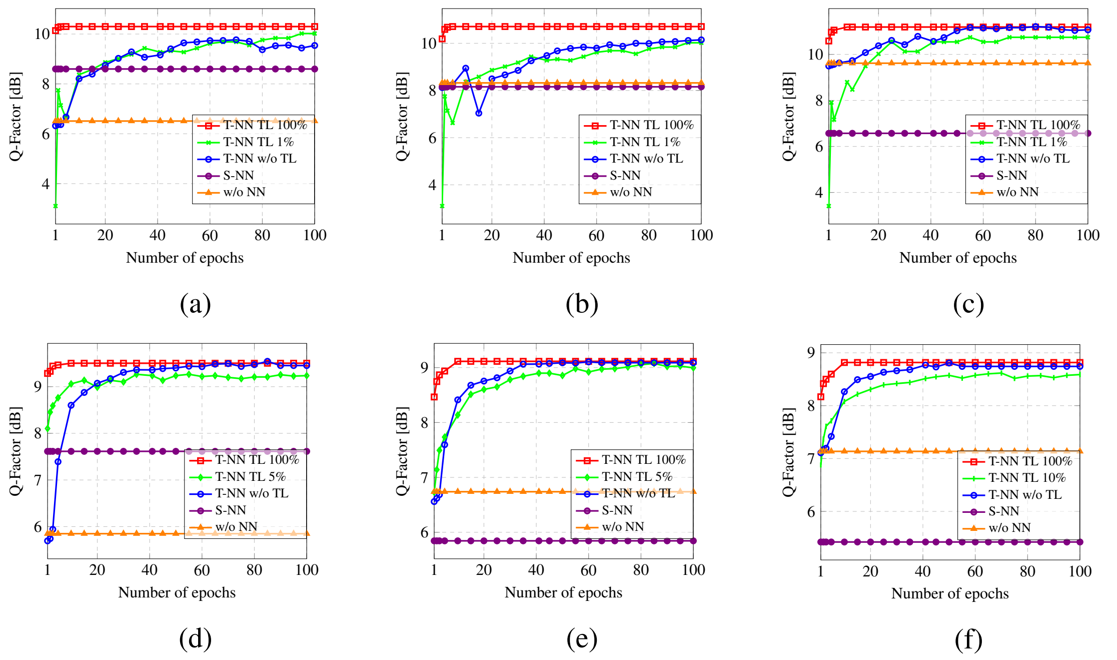


Fig. 5.6 Transfer learning between symbol rates. Case I: from 34.4 GBd to (a) 45 GBd, (b) 65 GBd, (c) 85 GBd, in a 18×50 km SSMF link using 16-QAM and 6 dBm of launch power. Case II: from 34.4 GBd to (d) 45 GBd, (e) 65 GBd, (f) 85 GBd, in a 9×50 km TWC fiber link using 16-QAM and 2 dBm of launch power.

range of symbol rates. In some cases, when retraining with the least possible training data percentage (green curve), a negative transfer (slight performance degradation) occurs at early epochs. This is a situation whereby the randomly selected portion of the training data has a distribution that deviates from the distribution of test data. But it can be seen that this negative transfer is resolved after just a few epochs (typically less than 10). This effect is attributed to the difficulty in training recurrent layers [235]. For example, when the training dataset is reduced to 1% of its original size, this corresponds to updating the weights approximately 99% less time per epoch compared to the training with the full dataset. Having such a small amount of updates per epoch for the RNN can lead to instability in the training. But, again, this effect can be sorted out by using several epochs.

Finally, just as it was done with the TL for the change in the launch power, now it is explained why TL from the lower to the higher symbol rate scenarios is used. The studies considering SSMF and TWC fiber are summarized in Table. II. The analysis of Table. II shows that transferring the learned features from a lower symbol rate to a higher one results in fewer epochs being required to train the NN-based equalizer, which indicates an increase in the TL effectiveness. This behavior is a consequence of reducing the impact of nonlinear effects that occur in optical fiber transmission by increasing the symbol rate while maintaining

Table II Dependence of the TL performance on the transfer direction, for the case where we change only the symbol rates from the source to target datasets.

Fiber	Scenario	Max Q-factor	Epochs required
SSMF	TL 34.4GBd → 45GBd	10.30	<5
	TL 64GBd → 45GBd	10.30	>100
	w/o TL	10.30	>100
TWC	TL 34.4GBd → 45GBd	9.50	<10
	TL 64GBd → 45GBd	9.50	>40
	w/o TL	9.50	>60

the same launch power. Just like when changing the launch power, here it is also important to choose the direction from a higher nonlinearity scenario to a lower one to improve the effectiveness of the TL.

5.7 The operational limits of transfer learning for NN-based equalizers

The effectiveness of TL when multiple changes simultaneously occur in the transmission setup is discussed in this last section. In this complicated case, it is transferred the learning from the source and re-train all layers of the NN model, because now the memory size and the intensity of nonlinearity change simultaneously.

To demonstrate the performance rendered by the TL, the following case is addressed. For SSMF, it was transferred the learning from the source domain, i.e., from the setup operating at 8 dBm power, with 34.4 GBd symbol rate, with 16QAM (we can represent the domain parameters as {8 dBm, 34.4 GBd, 16QAM}), to {6 dBm, 65 GBd, 64QAM}; the second case SSMF studied is the transfer from {4 dBm, 34.4 GBd, 16QAM}), to {8 dBm, 65 GBd, 16QAM}. For the TWC, it was studied the TL from {5 dBm, 34.4 GBd, 16QAM} towards {3 dBm, 65 GBd, 64QAM}, and from {8 dBm, 65 GBd, 16QAM}), to {4 dBm, 34.4 GBd, 16QAM}. These results are summarized in Table III, rows 7 to 10. TL is still useful even for such drastic scenario changes, and it can have variable but nonzero savings percentages in both the number of epochs and training dataset size, but the improvement can be much less pronounced as compared to the single parameter change scenarios.

When increasing the symbol rate while keeping the power and transmission setup the same, the power spectral density reduces. A reduced power spectral density implies a reduction in the nonlinear effects, and it was analyzed in the previous section on the TL of symbol rates. However, in test cases 8 and 10 from Table III, it was addressed the cases where the source and the targets had approximately the same power spectral density because of changing the symbol rate and the launch power proportionally. With this, it was achieved some non-trivial results regarding the direction of the transfer. In test case 8, when using the TWC fiber which is characterized by a chromatic dispersion parameter that is 6 times smaller and by a nonlinear coefficient that is almost twice the one of SSMF, respectively, the best performance occurred when following the “rule of power” (i.e., going from a higher launch power to a lower one). However, in test case 10, considering the use of SSMF, memory is the key factor. Thus, in this case, the best performance was achieved by following the “rule of symbol rate” (from low to high).

Table III Summary of the TL effectiveness for the different scenario changes addressed. Rows Test 1 to 6 depict the main results from sections 5.4 (the launch power change), 5.5 (the modulation format change), and 5.6 (the symbol rate change). Rows Test 7 to 12 highlight the results of section 5.7, where multiple changes in the transmission configuration, including the change of the fiber type, are analyzed. For computing the epochs saving ratio with the TL, we used 100% of the training dataset. The red color highlights the particular changes in each test case.

Test	**Scenarios of TL**				**Evaluation**		
	Fiber	Power [dBm]	Symbol rate [GBd]	Mod. Format [QAM]	Max Q-factor[dB]	Epochs Saved w TL	Dataset Saved w TL
1	TWC → TWC	5 → 3	34.4 → 34.4	16 → 16	12.66	90%	94%
2	SSMF → SSMF	8 → 6	34.4 → 34.4	16 → 16	10.16	94%	94%
3	TWC → TWC	5 → 2	34.4 → 34.4	16 → 64	4.64	84%	90%
4	SSMF → SSMF	8 → 4	34.4 → 34.4	16 → 64	8.14	94%	80%
5	TWC → TWC	2 → 2	34.4 → 45	16 → 16	9.54	90%	94%
6	SSMF → SSMF	6 → 6	34.4 → 45	16 → 16	9.76	98%	98%
7	TWC → TWC	5 → 3	34.4 → 65	16 → 64	8.66	30%	50%
8	TWC → TWC	8 → 4	65 → 34.4	16 → 16	10.08	78%	90%
9	SSMF → SSMF	8 → 6	34.4 → 65	16 → 64	5.75	55%	50%
10	SSMF → SSMF	4 → 8	34.4 → 65	16 → 16	7.24	94%	99%
11	TWC → SSMF	5 → 5	34.4 → 34.4	16 → 16	11.49	10%	50%
12	SSMF → TWC	5 → 5	34.4 → 34.4	16 → 16	10.73	0%	50%

Finally, it was studied the possible performance limits of the TL technique and discussed the prospect of transferring the learned features between different fiber setups. For this analysis, it was transferred the knowledge from the 18×50 km SSMF link to the 9×50 km TWC fiber link and vice versa, keeping the other transmission parameters the same: {5 dBm, 34.4 GBd, 16QAM}. Table III, rows 11 and 12, depicts the results for such a TL approach. This can be seen even when changing the fiber plant, and this is *the largest change in channel function considered in this study*, the TL can still provide a 50% reduction in the re-training dataset size, thus demonstrating the potential of the TL approach. However, it was not possible to identify a decrease in the number of necessary epochs for such a TL case. In fact, such a decrease in the TL effectiveness can be expected as the TL is well-tailored to different but still related source and target channel function [224, 225, 236, 237]. However, when the fiber type changes, the resulting channel posterior functions may become quite distinct, reducing the TL effectiveness.

Chapter 6

The Pitfalls in Neural Networks-based Optical Channel Equalizers

Despite the acknowledged advantages and benefits of neural networks, as explained in the preceding chapters of this dissertation, there are still difficulties and pitfalls that prevent the deployment of such solutions, particularly in optical transmission-related problems. This chapter will present the common misunderstandings and misinterpretations that arise when ML techniques are used for channel equalization in coherent optical communications.

Before discussing the pitfalls themselves, it is necessary to clarify why such issues emerge in the first place when neural networks are applied to optical communications. In optical transmission-related problems, 99% accuracy — when deciding which bit was transmitted — is typically the minimum required by cutting-edge transponders operating at pre-forward error correction (FEC). However, when working with neural networks in more known fields, such as image recognition, such ultra-high levels of precision are uncommon; hence, it can be anticipated the emergence of unexpected phenomena when employing neural networks in such high-precision systems. Therefore, stricter requirements must be imposed on such NN architectures in the case of optical communications, since the learning quality must be greater than in other fields.

In addition, this chapter focuses on the training challenges that arise when dealing with optical channel equalization. Overfitting and local minima difficulties are undoubtedly the most significant factors limiting the equalization capabilities of the NN-based DSP elements. The following sections will present details on how these two points may occur when employing approaches based on machine learning for impairment compensation in coherent optical communication systems.

6.1 The problem of overestimation in some quality of transmission metrics

Several performance metrics can be used to evaluate the quality of M-ary QAM transmission systems, including bit error rate (BER), mutual information (MI), Q-factor, signal-to-noise ratio (SNR), effective SNR, and error vector magnitude (EVM), to name a few.

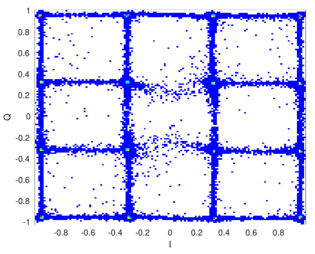
In digital communications, the post-FEC bit error rate (BER) is the ultimate quality of transmission (QoT) metric, as all actual transmissions have post-FEC BER values near 0. In addition to the post-FEC BER information, the MI can be a highly valuable metric as well, because it estimates the achievable spectral efficiency. When the decoder is unavailable (such as when assessing channel equalization using received soft symbols), the post-FEC metric is sometimes disregarded in favor of the pre-FEC metric, which is a typical performance statistic for uncoded systems.

After soft symbol equalization, the pre-FEC BER can vary based on the decision technique (hard or soft decision; HD and SD) employed. This metric predicts exactly the post-FEC BER for HD-FEC with appropriate interleaving [238]. Adopting such a metric, however, can lead to an inaccurate estimate of spectral efficiency, which is particularly apparent at low code rates (see [238] for more). However, when dealing with NN equalizers, the pre-FEC BER/Q is commonly used because obtaining the post-FEC BER/Q would require integrating the equalized symbols into the rest of the DSP chain, which is not worth the work for initial performance evaluations.

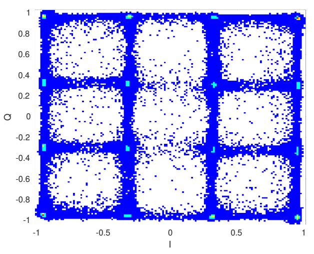
The SNR and EVM are the least accurate QoT measurements. However, these two measurements can also be utilized to provide a qualitative comparison of different transmission regimes. It is worth noting that these metrics are related to channels with known statistics, such as the additive white Gaussian noise (AWGN) channel. The correlations between the different QoT metrics in nonlinear channels, on the other hand, are not always known, and adopting extrapolations based on linear theories should be done with caution.

Although the pre-FEC BER is an important QoT metric, it does require the transmission of a known pattern, such as a training sequence, through the system for continuous performance monitoring[1]. As a result, the effective SNR (ESNR) and EVM gained popularity insofar as they lent themselves well to the study of unknown symbol sequences. Furthermore, as previously stated, these QoT metrics can still provide an accurate estimation of the BER when the system errors are primarily caused by optical AWGN, i.e., when fiber transmission is close to the linear regime and nonlinear effects (arising in TX/RX components and non-

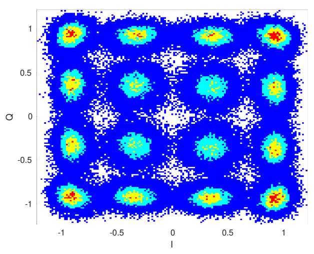
[1]Note that in practical terms, the pre-FEC BER is usually derived from the post-FEC BER assuming that the FEC can correct all errors.



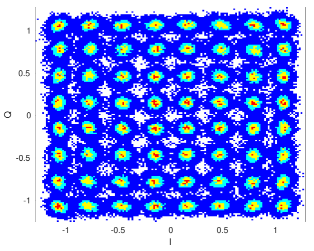
(a) $9.4\%/3.12\frac{\text{bits}}{\text{symbol}}/7.87\text{dB}$



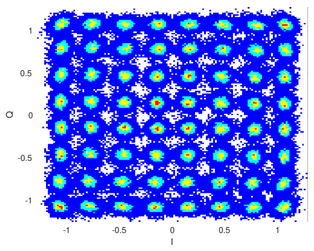
(b) $9.8\%/3.25\frac{\text{bits}}{\text{symbol}}/7.89\text{dB}$



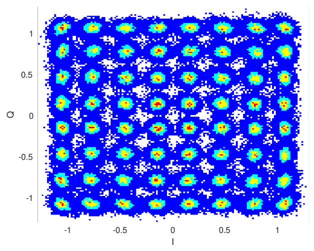
(c) $17.3\%/3.9\frac{\text{bits}}{\text{symbol}}/7.84\text{dB}$



(d) $7.9\%/5.91\frac{\text{bits}}{\text{symbol}}/8.94\text{dB}$



(e) $7.93\%/5.9\frac{\text{bits}}{\text{symbol}}/8.95\text{dB}$



(f) $7.92\%/5.92\frac{\text{bits}}{\text{symbol}}/8.94\text{dB}$

Fig. 6.1 Constellation diagrams after the MLP-based equalization, corresponding to the best performance when using the different QoT metrics. (a) and (d) lowest EVM; (b) and (e) best Q-factor; (c) and (f) highest MI. The simulated transmission scenarios are: 15 × 100 km SSMF, 4 dBm, 28 GBd 16-QAM (a)–(c); 10 × 60 km SSMF, 3 dBm, 30 GBd, 64-QAM (d)–(f). In each constellation, the QoT metrics were presented as (EVM/ MI/ Q-factor) in the caption.

additive noise) are almost negligible. For such metrics, it is inherently assumed that the reception is non-data-aided and that a quadratic-QAM signal constellation is used. The ESNR, EVM, and Q-factor can be calculated using the following expressions that are valid for a Gaussian-distributed signal [239–242]:

$$\text{EVM}_{RMS} = \left[\frac{1/N\sum_{n=1}^{N}|y_n - x_n|^2}{1/N\sum_{n=1}^{N}|x_n|^2}\right]^{\frac{1}{2}}, \tag{6.1}$$

$$\text{SNR} \approx \left[\frac{1}{\text{EVM}_{RMS}}\right]^2, \tag{6.2}$$

$$Q = \sqrt{2}\,\text{erfc}^{-1}(2 * \text{BER}), \tag{6.3}$$

where y_n is the normalized n-th symbol in the stream of measured symbols, x_n is the ideal normalized constellation point of the n-th symbol (i.e., a symbol from the M-QAM alphabet), N is the total number of symbols in the constellation, and erfc^{-1} is the inverse complementary error function. These metrics are typically considered in dB, using the relation: $T[\mathrm{dB}] = 20\log_{10}(T)$, where T is the QoT metric under investigation.

From a statistical perspective, the BER depends on the particular decision mechanism utilized at the receiver side, whereas the MI gives the effective transmission capacity regardless of the decision process. However, because the received soft symbols are used in the equalization, the MI cannot be conveyed explicitly. One option to derive the MI value is to estimate it by assuming a single-input single-output AWGN channel, which yields a suboptimal estimate giving out the MI's lower bound. This lower bound to the MI, $I(X;Y)$, can be expressed as [243–245]:

$$I(X;Y) = \mathbb{E}\left[\log_2\left(\frac{p(y|x_k)}{\sum_{i=0}^{MF-1} p(i)p(y|x_k)}\right)\right], \tag{6.4}$$

where $p(i)$ is the probability distribution of each k-th QAM alphabet symbol, and $p(y|x_k)$ defines the conditional probability of the received constellations given the k-th QAM input symbol. Then, the multivariate Gaussian distribution estimator [246, 247] can be used to calculate $p(y|x_k)$ of the transmitted-received complex symbols, which, ultimately, gives as the lower bound for the MI via Eq.(6.4).

Now that the foundation for the most pertinent QoT metrics were defined and discussed the assumptions underlying their computation, it is time to examine how these metrics may be used to assess the performance of the NN equalizer. This section's primary objective is to raise awareness of the fact that, depending on the modulation format, the NN structure, and the level of noise, the NN can produce the "jail window" pattern by squeezing the QAM constellations. Notice that numerous other works [40, 248–254] have also reported the "jail window" type constellations after the NN-based equalization, often without paying attention to the true meaning and consequences of this phenomenon. This effect is elucidated and any potential misinterpretations associated with it are discussed. The "jail window" effect originates from the regression task performed by NN equalizers based on MSE loss; the effect occurs because the ultimate objective of such NNs is to reduce the Euclidean distance between recovered and transmitted symbols. Note, however, that the "jail window" constellation violates the Gaussian channel assumption employed in the derivation of several of the aforementioned QoT metrics. In reality, this effect diminishes the precision of all Gaussian-assumption-based metrics (e.g., the ESNR) other than the Q-factor calculated directly from the BER obtained from direct error counting. Consequently, when utilizing these imprecise Gaussian-assumption metrics, it is possible to generate deceptive findings indicating that the NN performs well, whilst the true benefit supplied by the "jail window" constellation is grossly overstated. The "jail window" effect can also be explained by the mismatch between the true transmission performance metric (the BER/MI) and the metric that is minimized by the neural network training (the MSE loss), resulting in a disagreement between the objective function and the actual NN result. However, the BER itself cannot be used as a NN loss function because it is not differentiable. The effects of the "jail window" are examined in greater detail later in Section 6.5.

To illustrate the metric-related problem, two types of equalizers were tested, 1 hidden layer biLSTM and 3 hidden layers MLP, in four different numerical transmission setups: i) 20×80 km SSMF when transmitting 8-QAM at 34.4 GBd; ii) 15×100 km SSMF when transmitting 16-QAM at 28 GBd; iii) 30×50 km SSMF when transmitting 16-QAM at 64 GBd; iv) 10×60 km SSMF when transmitting 64-QAM at 30 GBd; the launch power was set to 3 dB higher than the power level for the best performance without NN equalizer. Clearly, the optimal launch power should increase when a nonlinear equalizer is utilized. The scenarios and equalization results for different metrics are summarized in Table. 6.1.

Table 6.1 Summary of the best performance after 1000 training epochs for different QoT metrics and transmission setups. (Simulation results of single-channel transmission)

Scenarios	MI [bits/symbol]			EVM [%]			Q-factor [dB]		
	Ref	biLSTM	MLP	Ref	biLSTM	MLP	Ref	biLSTM	MLP
Case i) 20×80km; SSMF; 5dBm; 8-QAM; 34.4GBd	2.93	2.99	2.90	22.7	5.9	8.3	7.82	10.72	8.37
Case ii) 15×100km; SSMF; 4dBm; 16-QAM; 28GBd	3.83	3.99	3.89	19.7	7.4	9.4	7.42	10.48	7.89
Case iii) 30×50km; SSMF; 5dBm; 16-QAM; 64GBd	3.84	3.97	3.87	19.4	8.1	10	7.48	9.15	7.65
Case iv) 10×60km; SSMF; 3dBm; 64-QAM; 30GBd	5.68	5.98	5.92	10.4	5	7.9	7.1	12.75	8.95

Here, note that Table 6.1 shows the best value of each QoT metric after running the process over 1000 epochs, but the best value did not occur at the same epoch number for all metrics. For example, in case ii), 15×100 km SSMF with 16-QAM at 28 GBd, it can be observed that for the epoch corresponding to the lowest EVM (9.4%), the MI and Q-factor values were, respectively, 3.12 bits/symbol and 7.87 dB; for the epoch corresponding to the highest MI (3.89 bits/symbol), the EVM and Q-factor values were, respectively, 17.27% and 7.84 dB; and for the epoch leading to the highest Q-factor (7.89 dB), the MI and EVM values were, respectively, 3.25 bits/symbol and 9.8%. This result clearly highlights that the particular QoT metric selected for performance optimization *does impact the eventual result for the system with an equalizer,* and that the differences can be even more accentuated if another transmission setup/ NN architecture is used. Indeed, depending on the quality

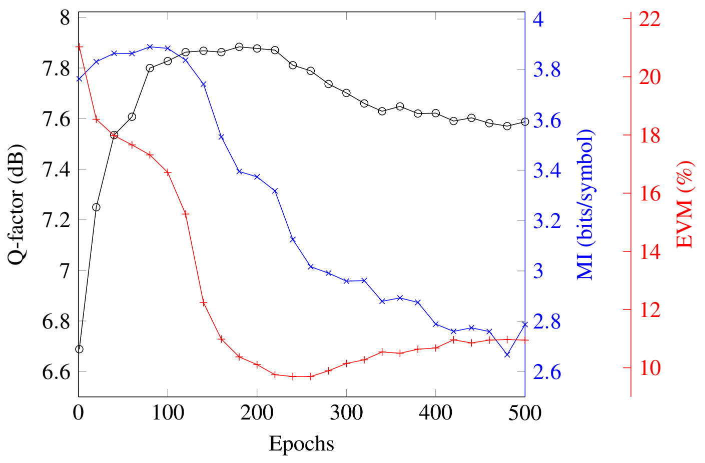


Fig. 6.2 Evolution of Q-factor (simulations), MI, and EVM over the training epochs when using the MLP equalizer in the 15 × 100 km, 4 dBm, 28 GBd 16-QAM system.

metric, the constellation after the equalization changes. Fig. 6.1 shows the constellation corresponding to the maximum of each metric for cases ii) and iv), where, in the first case, the "jail window" pattern does appear, but in the second it does not. For case ii), it can be seen the "jail window" with thin lines connecting the constellation points for the best EVM. The same "jail window", but with somewhat thicker lines, can be observed in the case leading to the best Q-factor. Contrarily, the traditional Gaussian-type constellations are observed in the case leading to the best MI. For case iv), the NN did not generate the "jail window" pattern. In this case, the minimum in the EVM and the maximum in the MI and Q-factor were achieved at approximately the same epoch. It is important to note that the Q-factor in all three cases did not change significantly, but the improvement in the EVM with the "jail window" caused a high decrease in the MI estimation from 3.89 bits/symbol (the epoch leading to the best MI) down to 3.12 bits/symbol (the epoch leading to the best EVM). This result is the consequence of the non-Gaussian shape of the constellation obtained, which violates the applicability conditions of Eq. (6.4) used to measure the MI. Furthermore, Fig. 6.1 points to one of the repercussions of the "jail window" pattern: the NN continued to minimize the Euclidean distance between the prediction and the labels, but this process no longer decreases the BER, and so it departed from the main goal of equalization. After the epoch of the highest MI (Fig. 6.1 (c)), the NN began to converge to the "jail window" and stopped further improving the Q-factor, which is why Fig. 6.1 (a) and (c) have roughly the same BER / Q-factor but Fig. 6.1 (a) has a much smaller EVM than Fig. 6.1 (c).

This effect shows that the usage of QoT metrics mentioned above for the regression task can be misleading, resulting in an underestimation of the true achievable MI and converging to a non-optimal equalizer's structure. Ultimately, Fig 6.2 presents the evolution of those three metrics over the epochs for case ii), to visualize the previous statements, where one can see that the overall behavior and the best values are different for different metrics.

Table 6.1 also demonstrates that the NN equalizer may lead to significant improvement of the EVM when compared to the linear equalization only, but without rendering the same corresponding improvement in the Q-factor. To better illustrate this effect, the following expression can be used to estimate the BER from EVM after the equalization [255]:

$$\mathrm{BER} = \kappa \frac{1-M^{-1/2}}{1/2\log_2(M)} \operatorname{erfc}\left[\sqrt{\frac{3/2}{(M-1)\mathrm{EVM}_{RMS}^2}}\right], \tag{6.5}$$

where M is the cardinality of the modulation format and κ is the correction factor. Now, it was estimated how much the result obtained through Eq. (6.5) deviates from the one calculated through the direct bit error counting. Starting by using the EVM and BER before the NN to calculate the correction factor κ: for case ii), the correction factor is $\kappa = 1.076$, which means that the BER calculated via (6.5) using the reference value of the EVM (before the NN equalization) is a suitable QoT estimator, almost matching the true BER value after the respective conversion. However, this equation shows how overestimated the EVM can be if the "jail window" is present after the NN equalization. For the MLP equalizer, by using Eq. (6.5) for case ii), it was obtained a Q-factor estimate of 13.62 dB while, in reality, it is only 7.89 dB. Additionally, in case iii), the Q-factor estimated using Eq. (6.5) is 13 dB, whereas the true Q-factor is just 7.65 dB. However, in case iv), where the "jail window" is absent (see Fig. 6.1) the Q-factor estimated through the EVM is equal to 9.4 dB, whereas the true one is 9.2 dB, showing a good match. Interestingly, even in cases where the "jail window" is absent, the Q-factor calculated through the EVM can be considerably overestimated. For instance, in case ii) and using the biLSTM equalizer, the Q-factor estimated through the EVM is 15.6 dB, whereas the true value is 10.7 dB.

Finally, the data in Table 6.1 also indicate that the "jail window" phenomenon happens less frequently when the biLSTM equalization is utilized as opposed to the MLP equalizer. Nonetheless, it may continue for the biLSTM in some circumstances, as will be discussed in section 6.5. In order to avoid an overestimation of the system's performance, the findings should be stated in terms of BER or Q-factor calculated from Eq. (6.3). If not, the constellations following equalization must be evaluated to ensure that they are still approximately Gaussian so that the other QoT measures provide a good representation of the actual performance.

6.2 PRBS Order Impact in Systems' Performance

There has been a plethora of empirical evidence that deep enough NNs can memorize random labels, even when using considerably large datasets [256]. Thus, in principle, by sufficiently increasing the size of a NN, one can always reduce the training error to small enough values, even though the task of learning a completely random sequence is meaningless.

Unfortunately, when transmitting commonly-used PRBS (for example, of orders 7, 9, 11, 15, 20, and 23), the benefits of NN-based equalization may be overestimated due to the phenomena described in the previous paragraph. In fact, the NN may learn the PRBS generation rules themselves, rather than estimating the inverse of the transmission channel model, and this naturally results in sharp performance degradation of the equalizer when truly random data (obtained from live traffic, for instance) is transmitted. This overfitting effect is especially relevant because, even in experiments, NN-based equalizers are frequently trained using PRBS-based datasets.

When dealing with NNs, two key PRBS-related issues must be addressed. First, the PRBS has a periodicity that is determined in terms of symbols by the PRBS order and the modulation format cardinality. The order 20 PRBS, for example, has a period of $2^{20} - 1$ bits, and if it used a 64-QAM signal with 6 bits per symbol representation, the symbol periodicity will be around 174k symbols[2]. To avoid NN learning such a pattern in this circumstance, the size of the training dataset used to train the NN must be less than the aforementioned quantity. This topic was investigated in Ref. [108], where an MLP classifier was used to check the overestimation of PRBSs of orders 7 and 15 in the 4-pulse amplitude modulation (PAM) transmission system using a large training dataset of size 2^{19}. According to the findings of that work, the NN was able to produce a reasonable result when tested on another PRBS sequence, but when tested on a fully random signal, the system's performance degraded dramatically because the NN did not learn the channel equalization but instead learned the PRBS periodicity.

The second important aspect relates to the PRBS generator's rules. A very simple and popular approach for the generation of PRBS is to use a linear feedback shift register (LFSR) with particular initialization and feedback. This approach can lead to a serious problem because if the NN's input is broad enough to catch the whole set of inputs used by the LFSR to construct the current symbol, the NN will be able to learn the PRBS model generator instead of performing the genuine nonlinearity mitigation task. More recently, in Ref. [257],

[2]The true period of symbols will be 6 times longer in this case, because this is when the bit sequence is aligned to the symbol boundaries. However, because the NN maps the input symbols onto a multidimensional space, the NN may also trace the information on bit periodicity. As a result, symbol periodicity is defined as the number of symbols that contain the entire bit periodicity.

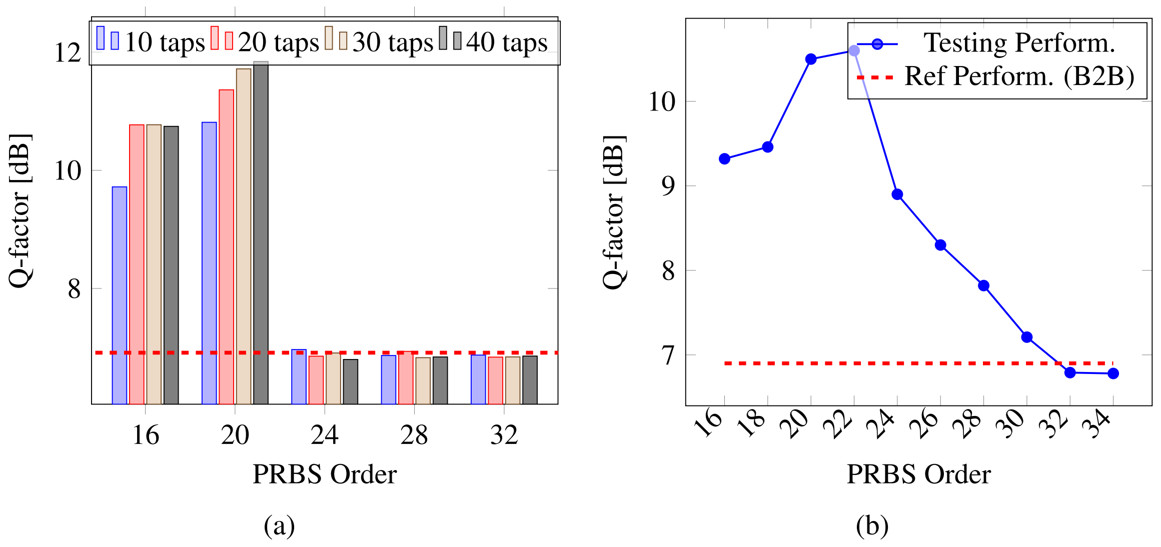


Fig. 6.3 Q-factor's dependencies on the PRBS order for (a) the different values of input memory (the different number of taps is highlighted with different bar's color) and different random seeds for training and testing with dataset sizes of 2^{18} symbols; (b) for the same random seed for training and testing, but taken from different chunks, with using 40 taps and different training dataset sizes per PRBS order as indicated in Table 6.2. The red dashed lines for both panels show the threshold, over which we arrive at the overestimation issue. (Simulation results)

this issue was mathematically studied and tested for an MLP classifier for both on-off keying (OOK) and 4-PAM transmission. In that Ref. [257], the authors used the PRBS of order 20 and generated 2^{19} bits, a sequence that is shorter than the generator periodicity. The bits were converted into symbols and fed into the NN classifier. The authors then created an input with the neighbor symbols but removed the current symbol to be classified. If the current symbol cannot be learned from adjacent symbols, the NN cannot correctly decide which bit was transmitted, and this situation corresponds to BER $= 0.5$. Otherwise, the BER should be significantly lower than 0.5, indicating that the NN can understand the symbol generation and mapping rules. In Ref. [257], the authors found that, in the case of OOK signals, the input memory length of 17 was enough to recover the symbols. In the case of transmission of 4-PAM signals, the same behavior was observed once the input had 9 taps. In the case of 4-PAM, some fewer taps were needed because each symbol carries 2 bits, whereas the OOK signals carry only 1 bit per symbol. In order to avoid the issues resulting from the limited PRBS length, instead of LSFR, the Mersenne Twister random sequence (MTRS) generator should be used, which can provide the sequences with a much longer period than that of the LFSR.

This section, different from the two previous works [108, 257], evaluated both aforementioned issues using the recurrent equalizer (biLSTM). The biLSTM's likelihood of learning the deterministic time correlation due to the PRBS order is by far superior to that of the MLP equalizer. Additionally, it was considered the transmission of a 64-QAM modulation format, which increases the number of bits per symbol, therefore enhancing the PRBS-related problems in the NN-based equalization.

Table 6.2 Summary of parameters such as periodicity and training/testing dataset size per PRBS order used in this study with 64-QAM.

PRBS Order	Periodicity Bits	Periodicity Symbols	Training Dataset Size	Testing Dataset Size
16	$2^{16}-1$	$\approx 10k$	$5k$	$5k$
18	$2^{18}-1$	$\approx 43k$	$20k$	$20k$
20	$2^{20}-1$	$\approx 174k$	$87k$	$87k$
22	$2^{22}-1$	$\approx 699k$	$262k$	$262k$
24	$2^{24}-1$	$\approx 2.79M$	$1M$	$1M$
26	$2^{26}-1$	$\approx 11.18M$	$1M$	$1M$
28	$2^{28}-1$	$\approx 44.7M$	$1M$	$1M$
30	$2^{30}-1$	$\approx 178.9M$	$1M$	$1M$
32	$2^{32}-1$	$\approx 715M$	$1M$	$1M$
34	$2^{34}-1$	$\approx 2.8B$	$1M$	$1M$

This study used generated data from 10 transmission runs with PRBS orders 16, 18, 20, 22, 24, 26, 28, 30, 32, and 34. AWGN was added to the RX input in a back-to-back scenario so that all data sets after the hard decision had the same Q-factor (6.9 dB). Since the only source of signal degradation is a random noise coming from the AWGN added, the NN should not provide any performance improvement, as the NN is a nonlinear deterministic function. Hence, any Q-factor improvement when employing the NN results from the NN's learning of the PRBS generation rule.

It was performed two tests to evaluate the impact of the PRBS order on the performance of NN. First, it was studied the impact of the training data set and PRBS symbol periodicity. Fig. 6.3 shows the Q-factor as a function of the PRBS order when 2^{18} symbols are used to train the NN. A different seed is used to generate another sequence of 2^{18} symbols (with the same PRBS order), which are used to test the NN. The analysis of Fig. 6.3 shows a clear improvement in the Q-factor for PRBS orders 16 and 20. This result confirms that the NN can learn the PRBS periodicity when the training data sets (2^{18} symbols) are larger than the symbol periodicity. Moreover, Fig. 6.3 also shows that increasing the number of taps enables achieving an even better Q-factor (the NN could train faster); this behavior was also observed in Ref. [257]. For the PRBS orders higher than 24, the training data set becomes smaller than the symbol periodicity, and, consequently, the NN is no longer capable of learning the PRBS generation rules: the values drop below the threshold (the dashed line).

At this stage, it is important to consider the dataset size for which the NN would learn the data generation rule for the higher-order PRBS. In order to answer this problem, it was conducted a second round of testing, in which it was trained the neural network using up to 1M symbols for PRBS orders ranging from 16 to 34. The main results of this study are reported in Fig. 6.3(b). Note that to guarantee that the NN learns the PRBS generation rule instead of the symbol periodicity, the training and testing datasets were generated with the same seed, but for training and testing, different data blocks with lengths shorter than the PRBS periodicity were selected. Table 6.2 provides the periodicity for each PRBS order, and the training/testing data set sizes for this study. The number of taps was set equal to 40 and the NN equalizer was trained for more than 5000 epochs. As before, if the Q-factor increases above the reference (red line in Fig. 6.3 (b) is the B2B level), this indicates that the NN was able to learn the data generation rule. Fig. 6.3 (b) shows that the Q-factor increases with PRBS order up to 22. This behavior can be explained by the fact that PRBS orders 16 and 18 were trained with 5k and 20k symbols, respectively. This training data set is too small to fully train the NN, but one cannot increase it further because of the periodicity constraint given in the third column of Table 6.2.

When the PRBS order is increased, the amount of training data required to learn the respective PRBS generation rule also increases. However, increasing the size of the training data beyond $1M$ becomes impractical (the training takes too long). Thus, the Q-factor curve in Fig. 6.3 (b) starts decreasing for PRBS orders above 22, indicating that the NN's capacity to recover the PRBS generation rule becomes progressively worse. In this case, the $1M$ training data size is insufficient (and/or the NN complexity is insufficient to learn the PRBS generation rule of the highest order). Thus, when increasing the PRBS order from 24 to 30, it was observed a decrease in the Q-factor until the point where the NN completely ceases to learn the PRBS generation rule (for the PRBS orders equal to or higher than 32). To conclude, this result underlines that the MTRS should always be employed in simulations, rather than the LSFR because it produces PRBS lengths that are virtually infinite. Therefore, an undesired overestimation of system performance can be prevented. However, when this is not possible, even bigger PRBS orders (e.g. ≥ 32) can be utilized with caution when working with experimental data since, depending on the size of the training dataset, modulation format, and input memory, overestimation can still occur.

6.3 DAC/ADC Memory Impact on Training NN Equalizers

As described in the previous section, the two most influential factors related to the data quality are the dataset size and its variability (the absence of spurious periodicity, bias, etc.). Fig. 6.4 shows the typical layout of a lab-style optical transmitter and a coherent receiver.

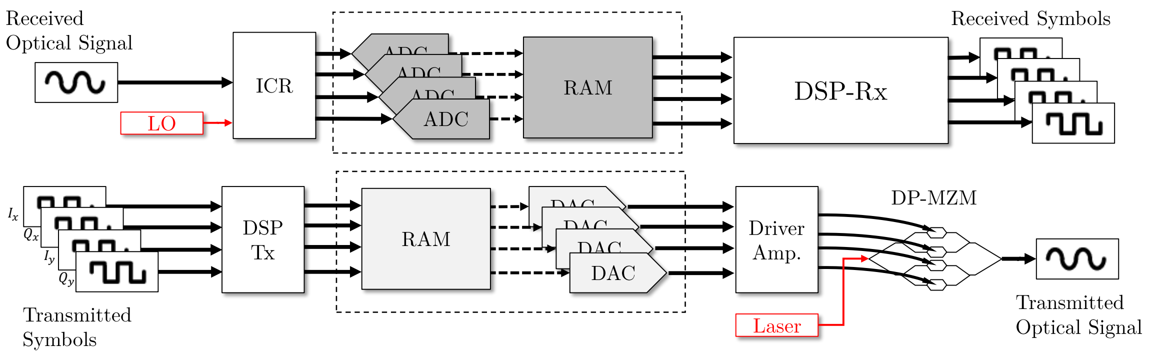


Fig. 6.4 A simplified block diagram illustrating the components of a typical transmitter and receiver. The DAC/ADC RAM memory is highlighted to emphasize its role in the transmission chain.

The Tx-DSP writes the signal samples into the DAC memory, and the signal is transmitted cyclically. On the Rx side, the output of the ADC is written into memory and processed subsequently by the Rx-DSP. When dealing with experimental data, there are several factors to take into account in order to ensure proper data quality for the NN training. First, the DAC and ADC memory is clearly limited in size, which puts an upper limit on the length of the signal to be transmitted. This applies primarily to capturing the buffers built into real-time transponders, where the memory resource is very precious and, hence, the available memory is usually rather limited. However, the same argument applies, in principle, to lab equipment, where, however, the limits are not so stringent. The other factors to consider are the sampling frequency of DAC and ADC devices and the symbol rate of the transmitted signal. For the NN training, we are interested in the sequences of symbols or bits; for a given DAC memory, the number of symbols the DAC can hold depends on the sampling frequency and symbol rate. Finally, an additional constraint can emerge due to the Rx DSP architecture. Often, to achieve synchronization, the DSP implementation assumes that the data are composed of frames of equal length. Furthermore, all frames may be assumed to contain the same payload data to facilitate the alignment of received and reference sequences. In this case, only the data obtained from a single received frame can be taken up for the NN's training and testing. This section will report on how the aforementioned properties impact the training of NN-based equalizers and data variability quality.

Consider an exemplary scenario where having a DAC / ADC with a memory equal to 512k samples per channel, operating at the sampling frequency of 80 GSample/s. Assume that the DSP requires about 10 frames holding identical data to achieve proper synchronization and evaluate BER. Then, the number of samples per frame is around 52k samples. Furthermore, it was assumed that the transmission symbol rate is 34.4 GBd. This leads to the number of effective symbols 52k/(80/34.4) $\approx$ 22k, which are available for NN training. This can be easily verified by applying the autocorrelation function for the received symbols: Fig. 6.5 shows the autocorrelation of the experimental data that was produced with a DAC specification close to the ones mentioned above. As can be seen, the difference in the peaks is around 22k, which is the same value that was calculated.

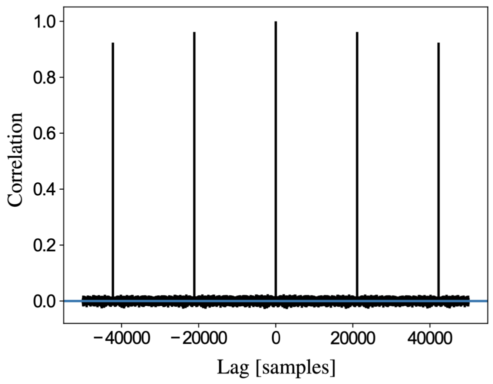
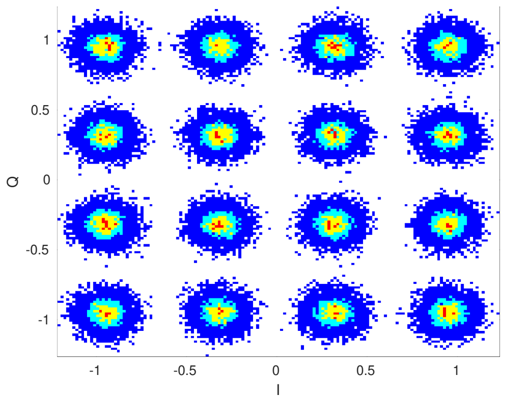
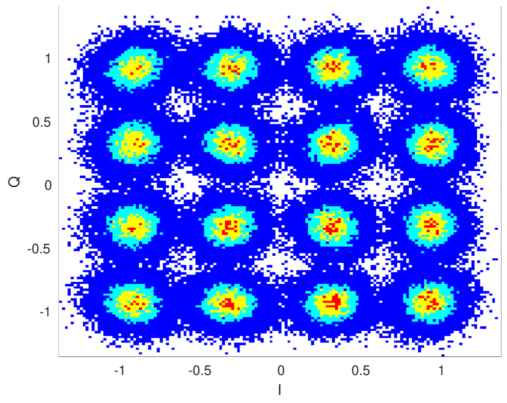


Fig. 6.5 The auto-correlation diagram of the received experimental signal shows the impact of the DAC memory on the signal's periodicity.

With this information, two main concerns can be raised in terms of the use of machine learning in systems employing DAC/ADC. First, having a system that is unintentionally biased towards one subset of symbols can result in poor model performance when the latter is validated on a different subset. This fact can ultimately lead to the overfitting of the NN-based equalizer. Second, as shown in Fig. 6.5, even when it was used a PRBS of order 32 or even completely random data, the periodicity could not be removed from the data since the DAC will repeat just a portion of the PRBS. Because of this, *the same transmission trace cannot be used for training and testing* even if it is selected non-overlapping chunks. Fig. 6.6 shows the constellations after NN equalization using different chunks of the same transmission trace, Fig. 6.6(a), and when using the transmission traces with different random seeds, Fig. 6.6(b). From these constellations, it is evident that using the same random seed to train and test the NN provides a constellation of outstanding quality (the respective Q-factor is equal to 13 dB). Nevertheless, the NN trained with the same random seed (the one that produced the picture in Fig. 6.6(a)) cannot generalize to perform the channel equalization: when using it with another random seed, it is seen that the resulting equalized constellation is noticeably degraded, and the NN's true performance in terms of Q-factor is just 8.66 dB, well below the previous overestimated value. This fact clearly demonstrates that, in this scenario, the system has overfitted over the training sequence pattern. Therefore, when reporting results using the same random seed for training and testing, you may overestimate the performance of your equalizer.



(a) Same random seed



(b) Different random seed

Fig. 6.6 16-QAM signal constellation after the NN-based equalization when (a) training and testing are carried out using different datasets but generated with the same random seed, and when (b) training and testing are run using datasets generated with different random seeds.

To avoid overfitting, the NN inputs were generated with the following procedure. First, it was recorded 60 measurement traces, each with 2^{18} symbols, using a different random seed to generate each trace. After that, the entire dataset was split into two parts: 50 traces were taken for training purposes and the remaining 10 traces were saved for testing. From these traces, it was generated the input of window vectors for each symbol to be recovered. Afterward, all these window vectors were concatenated to generate the training and testing datasets. Finally, after having $\approx$ 13M window vectors in the training and $\approx$ 2M window vectors in the testing datasets, 2^{20} random input vectors were selected from the overall $13M$ in each epoch while training the NN. To show the impact of not having enough variability in the training dataset, the NN equalizer performance trained was compared using the aforementioned dataset generation solution with the case where was trained the NN with 2^{20} vectors generated by just one random seed. To evaluate the performance of the equalizers, both trained models were tested with 2^{18} input vectors taken from $2M$ (the testing set) that were never used in the training.

The results of the comparison are summarized in Fig. 6.7, and several conclusions can be readily drawn from that figure. First, for the case where we trained the NN with only

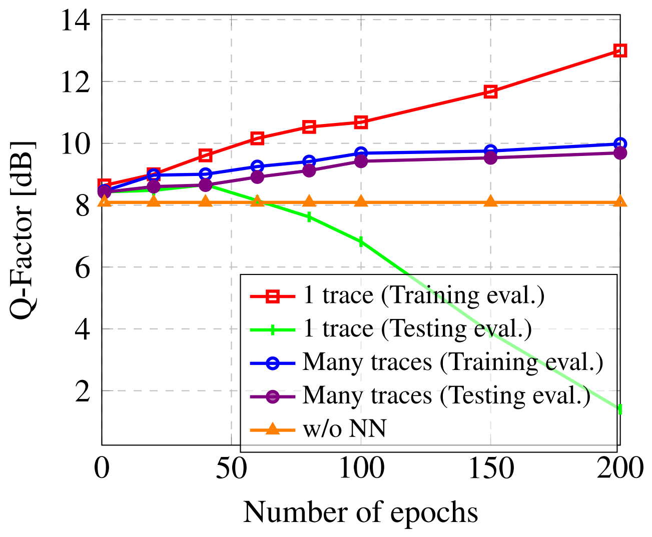


Fig. 6.7 Q-factor versus training epochs for the experiment with 16-QAM 5×50 km SSMF, 34.4 GBd, 6 dBm power, using the biLSTM equalizer. These curves refer to the training and testing performance of the NN when using one trace or when using multiple traces (our solution is described in the main text) for training.

one trace, the traditional overfitting appears: the training curve keeps growing while the testing curve bends down after some point as the model does not generalize. Because the model's variability was only for 22k points, it overfitted quickly, and the maximum Q-factor of the testing dataset was just 8.66 dB. On the other hand, when using the multiple trace training solution, it was possible to see that both the training and testing datasets' curves grow simultaneously, indicating the generalization capability of the equalizer. Using the proposed method, it was possible to reach a maximum Q-factor of 9.69 dB using the testing dataset, which is almost 1 dB higher than the value obtained when training the NN with one single trace. Note that this solution uses the different parts of the training dataset, which benefits not only from the diversity of different symbols picked from different random seeds but also from the fact that noise (which is different for each trace) adds diversity to the dataset as well. Heuristically, this noise is expected to "smear out" each data point, making it difficult for the network to properly match individual data points, and therefore reducing the overfitting [258]. Moreover, as it has been observed in several previous works [259–265], the noise injection to various parts of the NN during the back-propagation training can remarkably improve NN's generalization capability, and the latter observation fully complies with the result achieved with this solution, Fig. 6.7.

6.4 Regression or Classification (Soft Demapping) NN Equalizers: The Design Dilemma

As discussed in Sec. 2.1, equalization is the task of recovering the transmitted data X_n from the received data Y_n. It maps Y_n to the most likely transmitted data, $\hat{X}_n = f(Y_n;\Theta)$, for a given mapping function $f(\cdot)$ and the set of trainable parameters Θ, optimized according to some likelihood measure. In this case of NN-based equalization, $f(\cdot;\Theta)$ denotes the NN itself, and Θ denotes its trainable weights and biases. Here, X_n and Y_n can denote either a single sample of transmitted and received data or the sequences of samples for either one or both of them. For simplicity of presentation, it was assumed the single sample representation, and that X_n is chosen from a constellation alphabet $\{c_1, c_2, ..., c_m\}$ with $c_i \in \mathbb{C}$, the complex space.

The most popular and, perhaps, simple approach to improving the channel capacity is the receiver-based equalizer that is typically designed and optimized based on the minimum-mean-squared-error (MMSE) criteria. The usefulness of MMSE equalizers is stipulated for several reasons, including: (i) the MSE minimization is an optimal condition for the transmission over the additive white Gaussian noise (AWGN) channel; (ii) the MMSE is quite convenient for mathematical optimization because of convexity and differentiability of its objective function; and (iii) the MMSE equalizer is usually optimized independently of the underlying waveform or modulation format. In this section, this category of NN models based on MMSE is called a regression equalizer (Reg.), but one can term them as a dual-stage soft-demapping (i.e., the NN MMSE post-equalizer followed by the AWGN demapper).

Another approach is to design a model incorporating the decision step, which corresponds to the classifier in the machine learning literature, see, e.g., [266, Chapter 6]. Such a device is convenient insofar as the transmitted signal in digital communications is usually generated from a discrete finite-size constellation, e.g., quadrature amplitude modulation, QAM. This approach has received more attention recently [4, 76, 267–270], in view of the following reasons: (i) the classifier is optimized for the specific in-use modulation format; (ii) it directly maximizes the information rate, the main objective of the channel equalization, and outputs the likelihoods for each received symbol, a more suitable metric for the subsequent forward error correction; (iii) and, even more importantly, it can adapt itself to the correct statistical channel characteristics. In this section, this category of a predictive model is called a multi-class classifier (MC Class.), and in this communication study case, one can see them as single-stage soft demapping, in which it is used the categorical CEL in the learning process.

Given that there are two techniques to equalize a channel, it is only logical to wonder which of the two is the best or most appropriate under specific conditions. In this section, the performance of the multi-class CEL classifier and regression MMSE predictive models are examined, highlighting the potential drawbacks of each task for the NN-based long-haul optical channel soft-demapping problem.

A fair comparison between regression and classification is quite challenging, as these two produce different output variable types: discrete versus continuous, respectively. Such comparison studies have been carried out in theoretical machine learning-related work [271–273]. The desire to determine the optimal approach between regression- and classification-based NN models is a common concern in various fields. This motivates research into the use of these models in the development of NN-based soft-demappers for coherent optical systems. For the optical channel demapping, this comparison may be made more evident by contrasting the multi-class classifier output to that obtained with the regression, in terms of bit error rate (BER) (i.e., using a hard decision metric) or with respect to the mutual information (MI), i.e., using a soft decision metric.

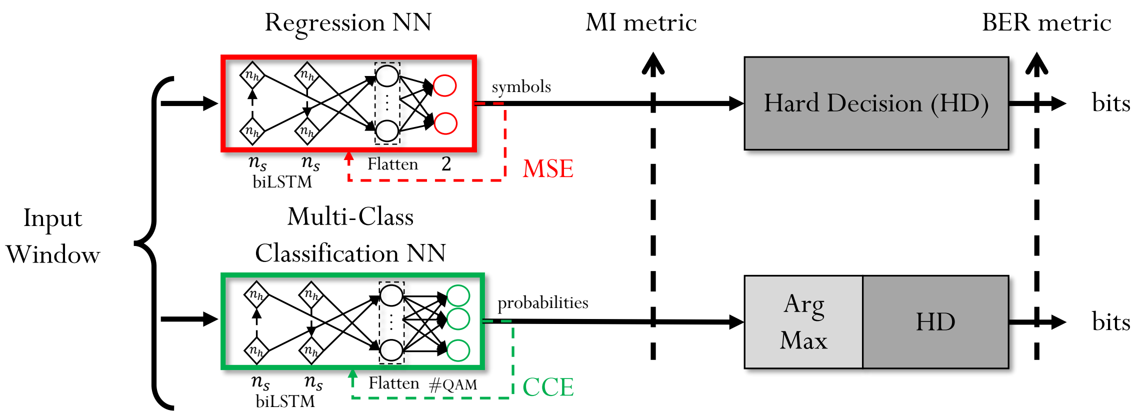


Fig. 6.8 Scheme of the two classic configurations of NN models, using the LSTM as an exemplary NN core: regression (top) and multi-class classification (bottom), in the context of channel equalization and soft demapping in communications.

As it can be seen in Fig.6.8, to guarantee a fair comparison, both regression NN and MC. Class NN has the same hidden structure composed of a biLSTM layer with n_s cells and n_h hidden units, and the only differences between using each architecture for regression or classification tasks occur in i) the structure of the output layer, as shown in Fig 6.8, and ii) in the loss function type used for each task. In the case of regression, the output layer has two linear neurons referring to the real and imaginary parts of the recovered symbol, and the loss function used is the MSE. For multi-class classification, the number of neurons in the output layer is determined by the modulation format cardinality (#QAM), and the NN structure ends with the softmax layer, while the loss function is the categorical CEL. It is important to point out once again that besides these two differences, the regression and classifier models that it was compared here, share the same number of inputs, hidden layers, and the training and test datasets are also the same.

Moreover, in the regression-based equalization, $f(Y_n;\Theta)$ is relaxed to $f_{\text{reg}}(Y_n;\Theta)$ outputting any complex value, and its likelihood maximization boils down to the minimization of MSE, i.e., to find the specific set of parameters Θ^*_{reg}:

$$\Theta^*_{\text{reg}} = \text{argmin}_\Theta \left\{ \mathbb{E}_{X_n,Y_n} \left[|X_n - f_{\text{reg}}(Y_n;\Theta)|^2 \right] \right\}. \tag{6.6}$$

The above expectation $\mathbb{E}_{X_n,Y_n}$ is taken over the samples of transmitted data and the corresponding received data. Those samples are, in fact, distributed according to $P(X_n,Y_n) = P(X_n)P(Y_n|X_n)$, where $P(X_n)$ is the transmitted signal distribution and $P(Y_n|X_n)$ describes how likely the channel output Y_n is upon the transmission of X_n. For regression, the MSE loss function can be treated as a simplification of the true likelihood measurement for the case of optical channel equalization: using the MSE optimization, the channel noise distribution is assumed as additive Gaussian; if the noise distribution is non-Gaussian or signal-dependent, then the MSE-based optimization fails to capture all information in the distorted sequences [274], and may not learn the model parameters that optimally maximize the negative log-likelihood that is the cross-entropy between the empirical distribution defined by the training set and the probability distribution defined by the model Ref. [266, Chapter 5.5]. In practice, when using MSE-based regression, the model tries to recover the deterministic part by assuming that the stochastic part has a Gaussian distribution with a signal-independent variance. This approximation is often reasonable, as in many systems the noise-signal interaction is smaller than the transmitter-induced and additive optical amplifier signal-independent noises, and the latter two can often be well approximated as a Gaussian process [275].

In contrast, the classification-based equalization combines the regression with soft demapping into a single NN, so it has as outputs: $\hat{X}_n \in \{c_1, c_2, ..., c_m\}$. In this case, $f(Y_n;\Theta)$ is relaxed to $f_{\text{cl}}(Y_n;\Theta)$ outputting a vector of posterior probabilities $(q_1, ..., q_m)$, where $q_k := Q(X_n = c_k|Y_n;\Theta)$, showing how likely $X_n = c_k$ are, given receiving Y_n. Then, $\hat{X}_n$ is equal to the c_k that has the largest posterior probability. It turns out that the "maximum likelihood" estimation (the best effort of the model $f_{\text{cl}}(\cdot)$) is obtained if the following categorical CEL of the actual posterior $P(X_n|Y_n)$ and $Q(X_n|Y_n;\Theta)$ is minimized:

$$\mathscr{X}(P,Q;\Theta) = -\mathbb{E}_{Y_n} \left[\sum_{k=1}^{m} P(c_k|Y_n) \log_2(Q(c_k|Y_n;\Theta)) \right]. \tag{6.7}$$

Equivalently, one can instead maximize

$$I_{\Theta}(X_n;\hat{X}_n) = \mathbb{E}_X[\log_2(P(X_n))] - \mathscr{X}(P,Q;\Theta), \tag{6.8}$$

where $I_{\Theta}(X_n;\hat{X}_n)$ is the mutual information for the mismatched decoding rule $Q(X_n|Y_n;\Theta)$ [276, Def. 12], [277, Theorem 2] [3]. As a result, the classification-based equalization is optimized for the following set of parameters[4]:

$$\Theta^*_{\mathrm{cl}} = \operatorname{argmax}_{\Theta}\left\{I_{\Theta}(X_n;\hat{X}_n)\right\}. \tag{6.9}$$

Therefore, for the classification, the cross-entropy loss [279] is the most suitable loss function concerning its meaning in information theory [280], effectively representing any type of noise statistics. However, there are two major drawbacks associated with this loss function, emerging specifically from the machine learning-related perspective, which can make the training of such a classifier a troublesome task. First, regardless of the corresponding inaccuracy in the target space, the CEL penalizes the misclassification between the two classes (i.e., between any two constellation points in such a problem) with the same "cost" value: the penalization ignores the spatial proximity of the labels, reflecting only the fact that the constellation point has been misclassified. However, typically, the account of the misclassification "type", i.e., when the cost of all errors is not equal, can be (and typically is) quite beneficial for the efficient NN's training. The cost of making a mistake can be determined by the projected and actual classes of an example [281, 282]. Each class represents a distinct notion that can be identified using a NN in the conventional classification task, for example, when one classifies different kinds of coordinate objects. However, when it comes to the problem of optical equalization, each class contains more information than just a label. In other words, the different classes correspond to different point positions in the constellation, and each class has its own nonlinear distortion level. The additional information about each label can be obtained by using the physical nature of each nonlinear level, i.e., the aforementioned distortion level that depends on the constellation point's power.

To better understand this question, consider the task of classifying symbols in, say a 16-QAM constellation. In this problem, the misclassification between classes that share the same decision boundary should cost less than the misclassification of the ones that do not

[3]In the case of regression-based equalization, the MI cannot be expressed directly from the optimization cost, Eq. (6.6), as it is possible in the case of classification. Instead, it was computed an MI using the Gaussian approximation of conditional probabilities (a mismatched distribution) as it was already introduced in Sec. 6.1 Eq.(6.4) [276, 278].

[4]Note that $I_{\Theta}(X_n;\hat{X}_n) \leq I(X;Y)$, the true mutual information (MI) of the channel, and the equality holds if $Q(X_n|Y_n;\Theta^*_{\mathrm{cl}}) = P(X_n|Y_n)$ for all (X_n,Y_n).

share any decision boundary: in the first case, one naturally shares some symbols due to the noise-induced clouds' spreading and overlap, while the second case is a "serious error". However, using the CEL will only capture errors in the target class: it discards any notion of errors that you might consider "false positives" and does not care how predicted probabilities are distributed other than the predicted probability of the true class (since one-hot encoded vectors are used), implying that only the predicted probability associated with the label influences the value of the CEL. Thus, from the standpoint of machine learning, one can say that there is a natural order among the labels of the target variable. The training difficulties that the standard CEL can bring in classification with natural ordering problems are discussed in more detail in Ref. [283].

The second classification disadvantage is the magnitude of the gradients that arise during the training process with the CEL. The CEL surfaces, according to Ref. [273], present fewer local minima than the square error-based losses (SEL), the type to which the MSE loss belongs. The CEL, on the other hand, has stronger gradients than the SEL, which leads to a stronger tendency of overfitting in CEL-trained systems, leading to the SEL's having a better generalization property in almost all scenarios examined in Ref. [273]. This result was explained thereby assuming that the CEL loss surface is more prone to sharp minima (narrow valleys) than the SEL's surface, making the overfitting considerably "easier" to happen for the former systems. Additionally, it was also shown that such classification-based systems can suffer from the gradient vanishing problem. For instance, Ref. [284] shows the gradient vanishing when using the softmax with categorical-CEL, and the same for the sigmoid with the binary-CEL was reported in Ref. [285]. It was observed the same tendency in the coherent channel equalization problem, when the MSE-based system generalized better than the categorical CEL systems, due to overfitting, sharp local minimal, and ultimately the gradient vanishing problem that is observed in the CEL-based learning.

Finally, one final disadvantage of classification-based equalizers is more relevant to their use in real transmission systems. When utilizing the classification, the equalization must have a predefined number of outputs that correspond to the constellation's cardinality. This means that the classifier model's operation is dependent on the modulation format on which it was trained. In other words, a classifier's practical implementation (e.g., in hardware) would limit the device's applicability to a particular modulation format only, which greatly reduces the device's flexibility and adaptability. However, as shown in Refs. [209], when using regression, the model can easily adapt itself to work with different modulation formats. As a result, regression-based models are much more flexible (reconfigurable) than classification models, allowing the former to be used under conditions different from the ones in which the regression models were trained.

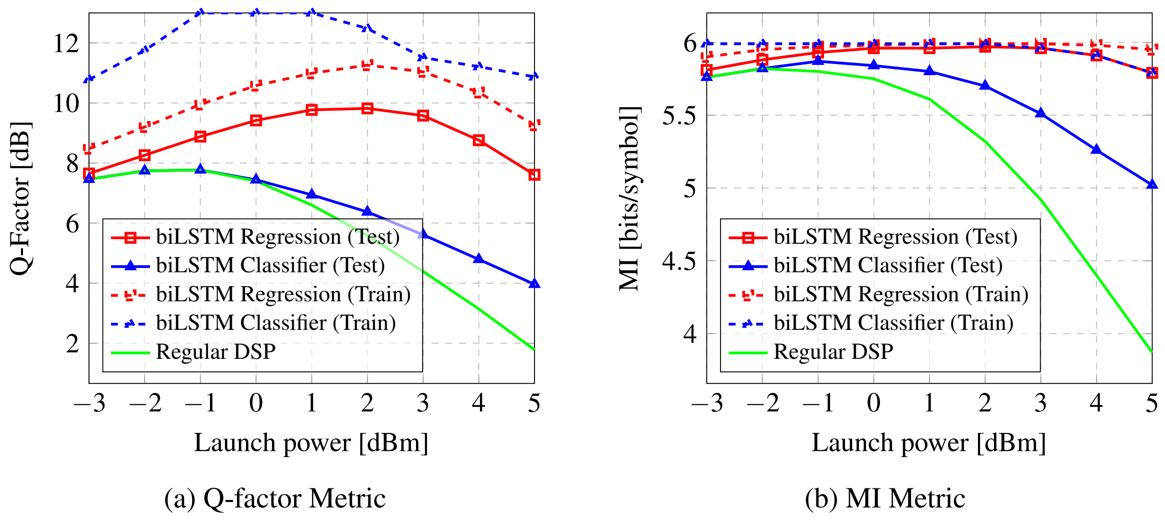


Fig. 6.9 Performance metrics' comparison for the regression- and classification-based equalizers, showing the impact of overfitting that can be seen when comparing the training and testing learning curves. The analyzed setup considered a 20×50 km SSMF link and a single-carrier-DP 64-QAM signal with 30 GBd and 0.1 RRC pulse shape (simulation results).

Now, since the key points related to the regression and classification tasks were covered, this work will focus on how an equalizer (or a soft-demapper) with the same architecture (applied to the data from the same transmission setup) performs, and how the result depends on whether the regression or classification task is employed. In this test, a single channel DP signal of 30 GBd is transmitted over 20×50 km employing 64-QAM (used in Figs. 6.9 and 6.10). The optical launch power is varied from -3 dBm to 5 dBm. A biLSTM equalizer was used with n_h =208 hidden units and n_s = 133 number of cells. Note that, the mini-batch size and learning rate were optimized for each equalizer type individually because it was observed that using the same ones in regression and classification was causing even stronger overfitting after a few epochs for the latter, which is one of the classification drawbacks.

Fig. 6.9 shows the Q-factor and MI dependencies for training and test datasets when the same biLSTM equalizer is used for regression and classification (of course, the output layers of NN structures are different for the two tasks). The Q-factor curves for the training and testing of the classifiers exhibit a considerable discrepancy, as shown in Fig. 6.9a, indicating that the classification model is overfitted. This noticeable difference in the classification task persisted even after optimizing the learning rate and mini-batch size. However, in the case of regression, the training and testing output curves behave almost identically, in contrast to the classifier's results. It follows that for the test scenarios, the regression model based on the MSE generalizes considerably better, which is consistent with the findings from Ref. [273], where the strong gradients in the classification task loss function impact learning further: the learning state gets "trapped" in a sharp local minimum of loss landscape. Furthermore, after the equalization, the regression equalizers resulted in a higher Q-factor than the achieved value for the classifiers, owing to the better generalization of the regression-based NN structures on the optical datasets. The maximum regression-based NN's Q-factor on the testing dataset was 9.82 dB at 2 dBm, while the maximum Q-factor for the classification was 7.74 dB at -1 dBm. Besides the Q-factor, it is also instructive to show the MI for both tasks as well. The reason for this is that the classification loss function, the CEL, is directly related to the MI metric, therefore, displaying the Q-factor after a hard decision can limit the entire potential of the model [209]. Looking at the MI values in Fig. 6.9b, one can see that the classifier's training performance was overfitted, yielding nearly the maximum MI attainable for each power when the training dataset was used. However, in the case of regression, the training and testing curves followed the same trend, indicating a much better generalization capability of the equalizers. The maximum MI on the testing dataset of the regression NN was 5.97 bits/symbol (note that this value is only a MI's lower bound), and the one for the classification was 5.87 bits/symbol (but this MI value is exact). So, it is possible to conclude that in typical optical transmission tasks, the machine learning drawbacks associated with conventional classification "overpower" the regression-related shortcomings related to statistical limitations.

Aside from the overfitting, it is also important to highlight the consequence of the second classification drawback, regarding the gradient vanishing in the training process. According to Ref. [266, Chapter 8.2.2], a practical way to demonstrate that the local minimum is potentially the cause of the learning problem is to verify that the gradient norm shrinks to some "insignificant" values along with the training. As in Ref. [286], it was depicted in Fig. 6.10 the gradient norm of the last layer for the biLSTM equalizer over the epochs to show the effects of the loss function on the behavior of the gradient norm. Unlike Ref. [286] that presented this plot for the MLP architecture, this study now shows it for the biLSTM equalizer, basically observing the same trend as reported for the MLP equalizer in a completely different transmission setup. From Fig. 6.10, it is clear that after just 200 epochs, the gradient norm for the categorical-CEL case drops from 1.25 to 7×10^{-5} and, with continued training, goes even lower to 8×10^{-7}. For the regression case using MSE, the gradient continues to stably decrease over the epochs, reaching a minimum of around 0.003. Therefore, it can be affirmed that the categorical CEL potentially has sharp local minima for the optical channel equalization task since, as described in Ref. [266, Chapter 8.2.2], the gradient norm shrinks to a negligible value (on the order of 10^{-7}) during training.

Finally, a brief discussion of the loss function study in the task of optical channel equalization/soft demapping is presented in this section. Although the MSE is a good fit

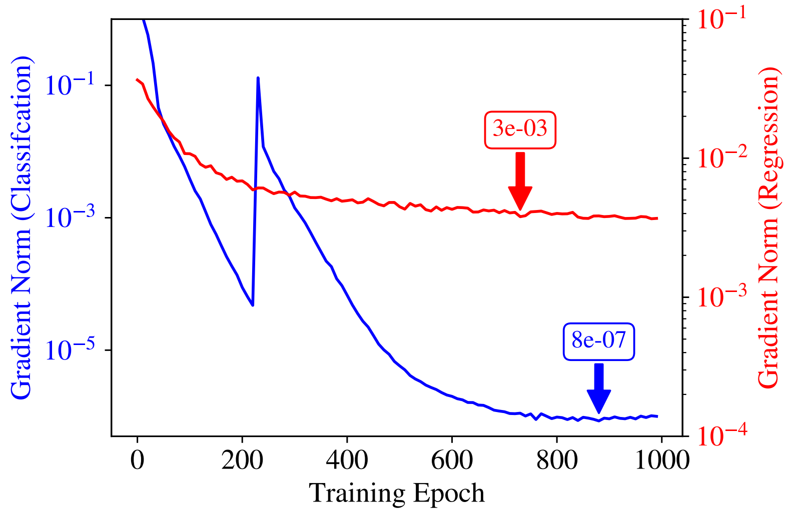


Fig. 6.10 Gradient norm over the training epochs for the Regression model (red) and Classifier model (blue), to highlight the gradient vanishing problem and sharp loss-landscape local minima.

when dealing with a Gaussian noise model and makes the NN learn effectively, delivering attractive Q-factor gains, in some scenarios, the MSE continues to decrease with the stochastic gradient descent (SGD) algorithm, but this does not translate into the BER improvement, resulting in falling into a local minimum. The ”jail window” constellation pattern visually indicates a mismatch between the MSE loss and the BER metric, and it is believed to appear for local minima in learning, producing seemingly good BER results, but not the ultimate equalization. In the next section, a scenario is presented where the model’s final Q-factor performance increases when the “jail window” disappears due to mini-batch size optimization. The suitability of CEL as the most suitable loss function for communication applications can be claimed since minimizing cross-entropy maximizes the mutual information of the system, matching the QoT metric of the transmission and the loss function of the learning algorithm, and therefore, no mismatch between the loss function and the QoT metric appears. However, other machine learning-related problems attributed to the learning process, namely SGD learning, were observed in this section. It is noted that the landscape of such a loss function has very sharp local minima, and the gradients tend to vanish in the early learning stage due to high accuracy requirements in optical channel equalization tasks. Therefore, even though the CEL does not have a statistical limitation for the noise likelihood, it poses many learning difficulties that can ultimately make its performance worse than that of the regression NNs based on MSE. It is also stressed here that several works have acknowledged that both regression and classification have disadvantages [287–293].

It can be confidently said that for optical communication, the ultimate answer to the question of what the best loss function would be is still not available, as each candidate has its drawbacks. It is believed that further investigation is required by looking at different fields to potentially find some other possible candidates. It has been observed that in computer vision, the majority of the pixels in an image describe the background, while only a few pixels express the objects in the image. This has resulted in inefficient training, as most parts of the image correspond to an easy prediction, which offers little relevant learning. To address this issue, Facebook A.I. developed a new modified approach named focal loss (FL) [293] by adding a weighting factor to the CEL function. The FL gives a higher weight to cases that are hardly misclassified, which corresponds to cases that have been misclassified after the HD process in communications. It is believed that the difficulty of dataset imbalance exists in virtually all high-accuracy communication-related equalization/demapping problems. For instance, in a system where an initial SER after HD is equal to 10^{-3}, only 100 symbols (0.1%) in a dataset of 100K symbols correspond to errors that persist after HD and that the model needs to learn from them to mitigate impairments. A similar focal loss function as in computer vision could be created for communications applications to improve the performance of classifiers. This could be an interesting topic for future research in the fields of optical channel equalization and efficient NN-based soft symbol demapping.

6.5 Interrelation Between Batch Size, Constellation Cardinality, and Equalization Quality

The NN training presupposes the use of optimization algorithms, such as SGD or Adam [294], to update the NN parameters (weights) based on the value of the loss function. Traditionally, two training methods are utilized: batch training, in which the algorithm updates the weights after the entire training dataset, and online learning, which is executed after each training sample.

In practice, however, stochastic optimization methods use mini-batches, i.e., the portions of all training examples with a size greater than one, but less than the entire training dataset, to compute the gradients using less memory and update the parameters after each mini-batch [266]. One reason for this is that modern NNs demand such a large amount of data that the computer dynamic memory cannot hold the entire dataset. Notably, the size of a mini-batch is regarded as one of the most important hyperparameters to consider when training an NN. Large mini-batches are well known in deep learning for their ability to significantly speed up the training process, while also providing an accurate approximation of the gradient [266]. Small mini-batches, on the other hand, have been shown to have a regularizing impact, preventing overfitting [266, 295]. The latter phenomenon can be explained by the fact that, while the large mini-batches can indicate the gradient's direction, the algorithm cannot forecast how long this trend would continue, usually causing the training process to take a large update step forward. This leads to unstable learning or falling into local minima. However, small mini-batches, while carrying added noise due to their smaller sampling across the entire dataset, result in small steps and the system can easily converge to a true direction [295]. Several studies have been carried out to investigate the effects of mini-batches size on the NNs performing traditional deep learning tasks. Ref. [296] proposed using small mini-batches to introduce noise in the gradient estimation and push the gradient away from sharp local minima; the authors of that Ref. [296] also demonstrated that the optimal batch size for the CIFAR-10 dataset is 80. In Ref. [297], it was empirically demonstrated that the large mini-batch sizes lead to the convergence at sharp local minima, resulting in poor NN generalization, whereas the smaller mini-batches lead to flatter local minima, allowing the NN to achieve a better generalization. However, no research has been conducted to study the effect of mini-batch size on the performance of NN-based regression equalizers in optical transmission.

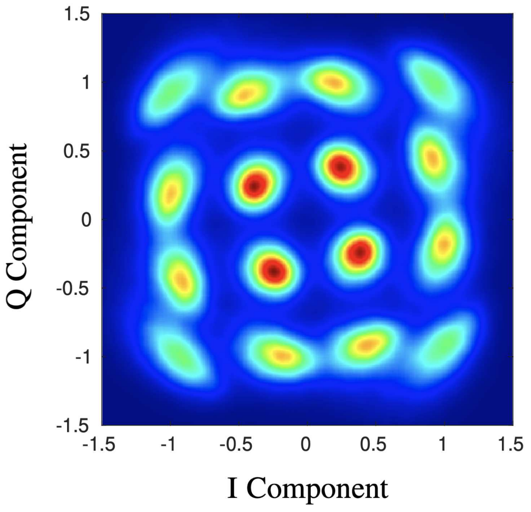


Fig. 6.11 Signal constellation of 16-QAM after metro-link (450 km) simulated optical transmission using a Nonzero Dispersion Fiber [9]. We used this highly-nonlinear exemplary system to visually demonstrate the distinct distortion levels at different constellation points: the constellation points at the outer layer are clearly more distorted than the ones closer to the origin.

The motivation for this study is that the mini-batch in the NN performing optical channel equalization has more physical meaning than it does in other "traditional" machine learning tasks, such as computer vision. Recall that different transmitted symbols in the constellation correspond to different optical field intensities, depending on the modulation format. This can be seen by looking at the 16-QAM received constellation diagram, Fig. 6.11, where the outermost points are the most distorted ones. This happens because the fiber nonlinearity is proportional to the cube of the optical field amplitude, so the most distant points (having the largest amplitude) experience the most nonlinear effects. Because the NN's parameters are updated after each mini-batch, the training process can be more efficient if each mini-batch contains training samples that cover the whole range of possible constellation amplitudes, to encompass different distortion levels. If this hypothesis is correct, there should be a relationship between the mini-batch size and the modulation format's cardinality, also affecting the NN's performance. This section reports tests and simulations to explain the aforementioned connections.

Table 6.3 Summary of biLSTM and MLP performance dependence on the mini-batch size for the different constellation cardinalities. Simulation results.

QAM	Q-factor Ref	MLP Equalizer Q-factor (Different Batch Sizes)						biLSTM Equalizer Q-factor (Different Batch Sizes)					
		8	16	32	64	128	2048	8	16	32	64	128	2048
8	8.65 dB	8.62 dB	8.70 dB	8.65 dB	8.74 dB	8.82 dB	9.44 dB	10.08 dB	10.25 dB	10.51 dB	10.71 dB	11.28 dB	13 dB
16	7.22 dB	6.91 dB	7.02 dB	7.09 dB	7.17 dB	7.28 dB	7.72 dB	8.58 dB	9.14 dB	9.72 dB	10.10 dB	10.43 dB	11.57 dB
32	5.05 dB	4.47 dB	4.57 dB	4.69 dB	4.80 dB	5.01 dB	5.58 dB	7.02 dB	7.66 dB	8.36 dB	8.69 dB	8.75 dB	8.79 dB
64	3.15 dB	2.67 dB	2.73 dB	2.95 dB	2.96 dB	3.06 dB	3.79 dB	5.78 dB	5.87 dB	6.29 dB	6.47 dB	6.51 dB	6.78 dB
128	1.53 dB	0.77 dB	0.90 dB	1.10 dB	1.30 dB	1.36 dB	2.09 dB	3.54 dB	3.92 dB	4.09 dB	4.13 dB	4.14 dB	4.27 dB

Table 6.3 shows the Q-factor of optical signals equalized by the MLP and biLSTM MSE-regression equalizers for a range of modulation formats: 8-, 16-, 32-, 64, 128-QAM[5] and mini-batch sizes: size 8, 16, 32, 64, 128, and 2048. In this section, simulations were performed for the SSMF 16×60 km link at 5 dBm launch power and 34.4 GBd symbol rate. When the size of the mini-batch increases, it can be observed a progressive growth in the Q-factor after equalization. Considering the case of biLSTM, which gives the best equalization quality. When the mini-batch size is increased for the 8- and 16-QAM, the Q-factor after the biLSTM-based equalization dramatically improves. For example, comparing the post-equalization Q-factors in the 16-QAM scenario, for mini-batch sizes 8 and 2048, a 3 dB improvement can be seen for the largest mini-batch over the smallest one.

This behavior of mini-batch vs. Q-factor gain for biLSTM is illustrated in Fig. 6.12 for 8-QAM and 16-QAM. Note that increasing the mini-batch size above 2048 does not provide any further improvement: the Q-factor reaches some kind of plateau (saturates) after the batch size of approximately 1000. When using a higher constellation cardinality, 128-QAM,

[5]Note that to generate the 8-QAM constellation it was used the standard Matlab function `qammod`; the same constellation shape was used in Refs. [298–301].

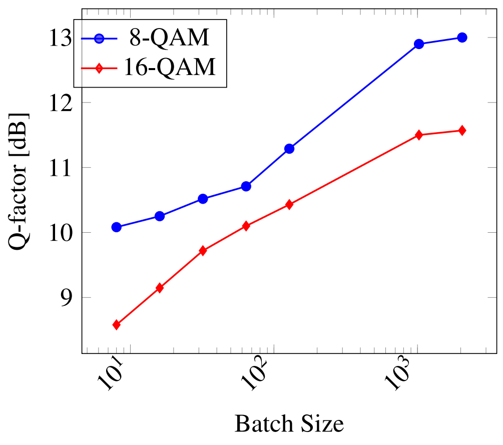


Fig. 6.12 Q-factor of 8- and 16-QAM simulated signals equalized by the biLSTM trained with different mini-batch sizes: 8, 16, 32, 64, 128, 1024, and 2048. The simulated transmission setup is described in the main text.

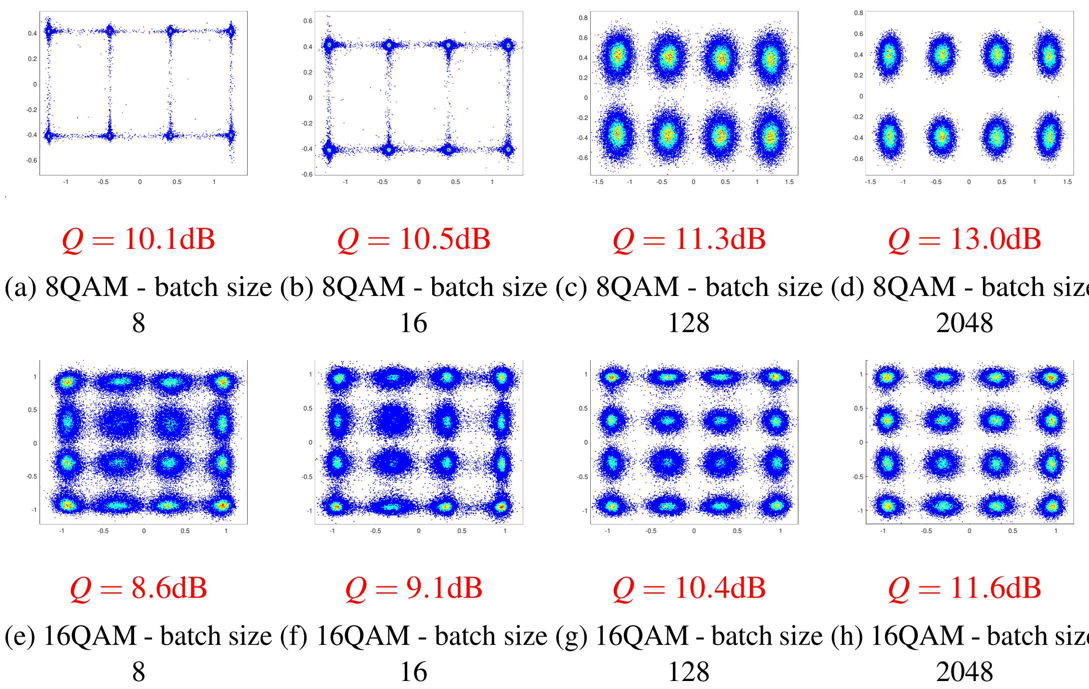


Fig. 6.13 Signal constellations (simulation results) for different batch sizes after equalization using the biLSTM equalizer for 8- and 16-QAM. The transmission system is described in the text. The Q-factor of each constellation is also shown to highlight the improvement in QoT.

the improvement rendered by the biLSTM equalizer increases from 3.54 to 4.27 dB when changing from the mini-batch size from 8 to 2048. We note that, from Table 6.3, for the 128-QAM with the MLP equalization, the difference between the Q-factors corresponding to the lowest and highest mini-batch sizes is approximately 1.3 dB, which is bigger than the difference for the biLSTM. The increase in Q-factor backs up the claim that the NNs can learn more efficiently when given training samples covering all nonlinearity levels in the constellation.

Together with the improvement in Q-factor (up to some limiting value) following the growth of the mini-batch size, one can see the interrelation between the constellation distribution after equalization and the mini-batch size used in training. The constellations for the 8- and 16-QAM systems for different mini-batch sizes utilizing the biLSTM equalization, are given in Fig. 6.13. In this figure, it can be observed that utilizing a small batch size for 8-QAM drives the NN-based equalizer to fall into the previously discussed “jail window” pattern rather than into the Gaussian-type distribution, but the latter can be obtained when using larger mini-batch sizes. In the 16-QAM case, the signal constellations after NNs with small mini-batches become noticeably distorted, and the “jail window” elements are clearly seen. At the same time, for the larger mini-batches, the 16-QAM constellations show little distortion, with clear concentrations at the center of each constellation point.

The degradation of constellations from circular clusters into the “jail window” pattern when training with small mini-batch sizes, is actually a new and intriguing phenomenon: so far, there have been no studies relating the equalizer’s performance deterioration to the size of the training mini-batches. Generally, as already seen, several factors can contribute to the patterning effect of the “jail window”. In this section, it is addressed the specific situation, where the “jail window” occurred solely due to the batch size effect, so it is possible to understand how and why the batch size relates to the “jail window” patterning. In a more general case, different contributions amalgamate, resulting in the “jail window” pattern; the latter is always an indication that something is not good with the training or with the NN system itself. This is actually evident from the results in Fig. 6.13 (a) to (d). In case (a), where the learned weights resulted in the “jail window”, the Q-factor is around 10 dB, while in case (d), where the learned weights did not generate the “jail window”, it was measured the Q-factor to be around 13 dB. Therefore, it is expected that when the “jail window” pattern is present, even though the performance metric value can be rated as “satisfactory”, in the training, it has most likely reached just a local minimum, and some better equalization results can be achieved with the more appropriate training or by using a modification of the NN architecture.

To explain the degradation of signal constellations further, when training the biLSTM equalizer with small mini-batch sizes, it was investigated the weight distribution inside the NNs. More precisely, it was compared the weight distributions in the last linear layer (output layer) of the biLSTM, and the distribution in the forward LSTM layer, for two values of the training mini-batch size: 16 and 2048, using an 8-QAM system as an example. Fig. 6.14

shows that the weights, when training with small mini-batches, range over a considerably larger interval than when training the NN with larger mini-batches. From Fig. 6.14b, it can be seen that the final layer weights were significantly saturated when the NNs were trained with small mini-batches[6]. The saturation is indicated by the presence of large value weights and typically degrades the performance of the NN. At this point, one can hypothesize that the presence of large weights makes the output of the NN strongly dominated by a few weights with a large value. This is a widely known phenomenon in deep learning when training NNs to perform regression tasks [266, 302]. The NNs featuring large weights tend to have the regression output converging to a few values that are hard-coded from the input. In the case of optical signal equalization, this effect deteriorates the signal constellations into a "jail window".

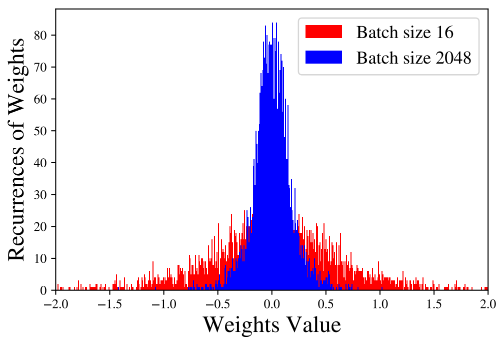
To carry out the analysis further, it was examined the number of outlier weights in the linear layer. As can be seen in Fig. 6.14, weights greater than one can be considered outliers. It was observed that in the case of 8-QAM, the real parts of the recovered symbols have twice as many outliers (larger than 1) weights as the imaginary parts (298 for the latter case in comparison with 153 for the former)[7] When increasing the outlier weight threshold value criteria from 1 to 4, it found just four weights on the neuron representing the real value of the constellation, while the neuron for the imaginary part had just two. This is the exact ratio between the number of continuous straight "lines" (each "line" is made up of equalized constellation points) in the amplitudes of the in-phase and quadrature, as seen in Fig. 6.13a.

We propose the following explanation for the aforementioned patterning effects. When employing small mini-batch sizes with a small cardinality constellation, there exists a noticeable probability that many points in the batch belong to just a single amplitude level. As a result, the NN tends to learn the inverse channel function at this particular nonlinearity level and is more likely to hard-code some inputs to this obvious amplitude output. This issue is less likely to occur with high-cardinality modulation formats since it is less probable that the batch will contain a lot of samples belonging to the same amplitude level. However, even in high modulation formats, it was possible to see the "jail window" pattern when using the MLP equalizer, but now with less intensity[8]. One possible explanation for why the optical signals recovered by the MLP equalizer are more prone to constellation degeneration than in the case of the biLSTM equalizer is that the feed-forward structure (e.g., the MLP)

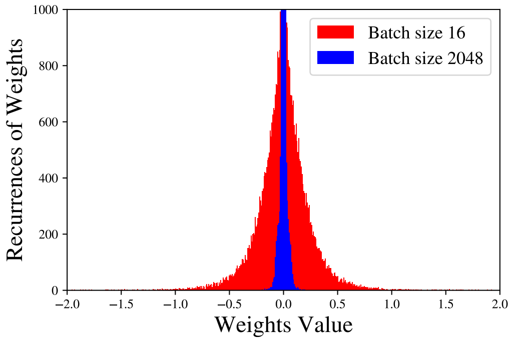
[6]Well-trained NNs usually have the weights' distribution being close to the Gaussian distribution with a small variance. When this variance is too large, say more than one, as shown in Fig. 6.14, it usually indicates saturation of weights, which, in the NN terms, means disregarding some important input features.

[7]This effect persists in the case of symmetric 16-QAM and other constellations when the "jail window" occurs, such that it is not relevant to the asymmetry of the 8-QAM constellations used in this study.

[8]It was observed that the "jail window" pattern is modulation format dependent. An interesting discussion on the fact that non-linear MSE equalizers realize a stair function can be found in Ref. [280]



(a) Weight distribution at the LSTM layer



(b) Weight distribution at the final linear layer

Fig. 6.14 Weight distributions of the biLSTM equalizer's forward and output layers, corresponding to the mini-batch size of 16 (red) and 2048 (blue).

makes it easier to hard-code input data to the output. At the same time, the recurrent-type structures (e.g., the LSTM type) allow parameters to be shared across the NN model [266], making the hard-coding effect more difficult to emerge. Here, however, one can remark on the performance of the MLP and LSTM equalizers. Both Sections 6.1 and 6.5 demonstrated that the LSTM equalization produces a better result than the MLP equalizer. The LSTM, which can be deemed as a nonlinear infinite impulse response (IIR) filter, can represent the inverse of a nonlinear channel more accurately than equalizers that use an MLP structure; the latter is actually a non-linear finite impulse response (FIR) filter [303]. IIR filters, unlike FIR filters, do not require that the system has finite memory, giving the LSTM-type equalization an advantage. Theoretically, the MLP may also achieve such a high degree of performance when the input accounts for large enough memory; but, due to the overfitting of the MLP structure[9], which turns out to be extremely difficult to achieve. On the other hand, because the large-memory handling LSTM structures are simpler than MLPs, the overfitting there does not occur, or at least, is much less pronounced.

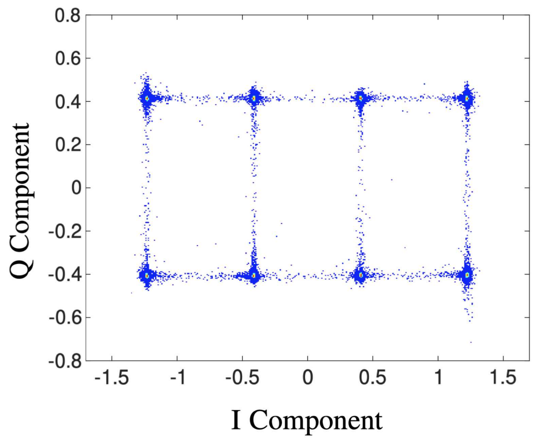
After considering the deteriorating effects attributed to the presence of large-value weights, we advocate the use of a well-known regularization technique, L^2 regularization, as a feasible way of minimizing the degradation in the equalized signal constellations. The idea behind L^2 regularization is to penalize large weights and favor smaller weights throughout the model [302]. From a Bayesian statistics standpoint, the addition of L^2 regularization is equivalent to performing a maximum a posteriori estimation with a Gaussian prior; the traditional MSE loss function corresponds to a maximum likelihood estimation when the

[9]To recover the nonlinear channel, the MLP would need a complicated structure with large memory. However, such a redundant architecture is equally susceptible to learning the training dataset, which leads to overfitting [151].

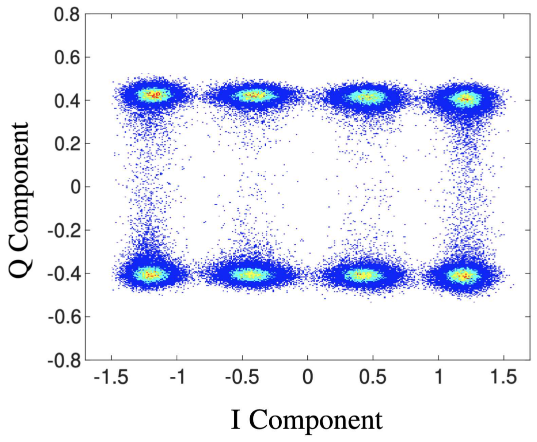
likelihood has a Gaussian distribution. This means that after the learning, the equalization transfer function (likelihood) has been re-weighted to reflect a prior, making it much more difficult to have hard-coded correlations between the input and output (recall that the hard-coding is one of the reasons causing the "jail window" pattern). The following equation represents the L^2 contribution to the regularized loss function $L(y, \hat{y})$:

$$L(y, \hat{y}) = \mathrm{MSE}(y, \hat{y}) + \lambda \sum_{i=1}^{n} w_i^2, \tag{6.10}$$

where the first addend is the initial MSE loss function, λ is the regularization parameter that must be optimized[10], n is the total number of weights in the NN topology, and w_i is the i-th weight in the NN topology.



(a) Without regularisation



(b) With regularization

Fig. 6.15 8-QAM signal constellations after the NN-based simulated channel equalization when training with mini-batch size equal to 16 without (a) and with (b) L^2 regularization.

Fig. 6.15 shows the signal constellations of the 8-QAM signal equalized by the NNs trained with the mini-batch size of 16, with and without regularization. It was observed that when using regularization, the degradation of signal constellations can be prevented. It is worth pointing out that although regularization can render better constellations, the equalizing performance is still badly affected by the small mini-batch sizes: even with regularization, it cannot match the performance of the NNs trained with large mini-batches. Thus, while the large weights contribute to the degeneration of constellations, this is not the only detrimental effect caused by a small mini-batch size. This observation suggests that the most important

[10]The regularization parameter was tested from the range between 0.01 and 0.0001, but no drastic difference was observed in terms of Q-factor improvement. In the plots presented in this section, it used the parameter value $\lambda = 0.001$.

problem with small mini-batch sizes can relate to the insufficient number of unique amplitude levels that are present in each mini-batch, as suggested above.

Fig. 6.15a and 6.15b demonstrate the signal constellations at the epoch with the highest Q-factor. It was observed that when training the NNs after this level, performance degeneration took place even with the regularization. Although most of the weights after the regularization turned out to have a relatively low value, there still exist significantly larger outliers. One possibility for this is that these outliers contribute to the distortion of the signal's constellations. Interestingly, using large mini-batch results in a narrower distribution of the weights, which does not feature any significant outliers even when regularization is not applied. This observation strongly supports the claim that training with an insufficiently large mini-batch brings about the performance degradation of NN-based coherent optical channel equalizers.

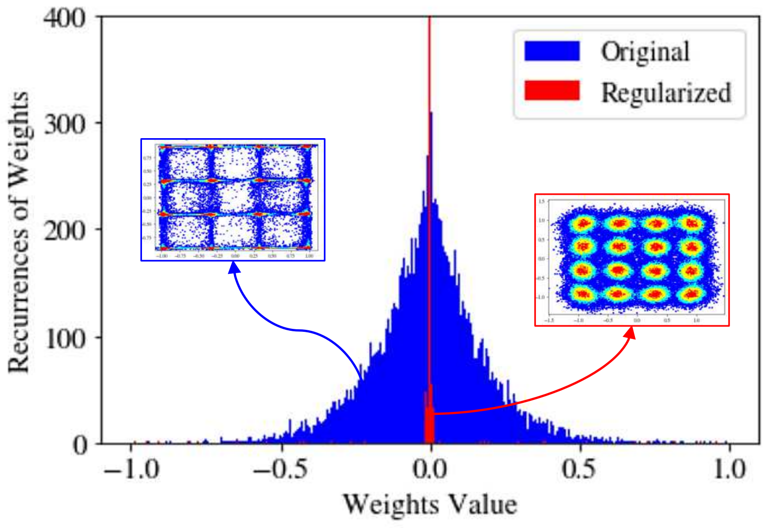


Fig. 6.16 Distribution of MLP weights for the entire model: non-regularized (Original) and regularized cases. The insets show the respective equalized constellations for each case.

Finally, it was applied the L^2 regularization to all hidden layers of the MLP equalizer in another transmission case where the "jail window" was really strong: it was the case (ii) described in Section 6.1 and given in Fig. 6.1 (b). The weight distribution of the entire model for the MSE (marked as "Original") and for the MSE with the L^2 regularization (marked as "Regularized") is presented in Fig. 6.16. From this result, it can be seen that, this time, for a higher modulation format (16-QAM), the regularization makes the original "jail window" constellation again a set of "Gaussian-like" clusters, see the corresponding insets in Fig. 6.16. However, even though the regularization produced a visually different constellation output, the original and regularized model revealed almost identical performance in terms of the eventual Q-factor: the original one gave 7.89 dB, regularized – 7.61 dB. This fact shows that regularization is not the solution to make the model learn further[11], but it rather helped us to identify which element in the NN-model was responsible for the “jail window” pattern.

In conclusion, the challenges and common misconceptions that often arise during the development of NN-based equalizers for high-speed coherent optical communication systems include poor model generalization, dependence on dataset characteristics (periodicity), overfitting behavior and indications, performance overestimation, and inaccurate estimation of computational complexity. Table 6.4 summarizes some findings and recommendations. Although this exploration of pitfalls is undoubtedly incomplete, the identified issues represent some of the most common problems that pose a significant risk to developing efficient NN-based equalizers and other machine-learning structures used in optical transmission lines.

Table 6.4 Overview of Main Issues in Devising Efficient NN-based Equalizers

Topic	Problem	Solution
QoT Metrics	The “jail window” effect in the NN-based equalizers. The use of the metrics assuming Gaussian channel statistics, such as EVM and SNR can lead to overestimated results in terms of BER.	To avoid overestimation it is recommended to always report the results in terms of BER or Q-factor calculated directly from BER.
PRBS Order	Depending on the PRBS order, the NN can learn the PRBS periodicity or the generation rule of the PRBS sequence that is based on its linear feedback shift register.	The training and testing datasets must be different fragments within the PRBS periodicity range. This periodicity depends on the PRBS order and the modulation format. Also, high PRBS orders should be used, ($\geq$32) , and the NNs are to be tested with a B2B system to guarantee the absence of overestimation and overfitting effects.
DAC Memory	Depending on the DSP implementation, the DAC memory limits the data variability to a few thousand samples, which makes the NN to overfit quickly into the training dataset statistics.	Increase the variability by, concatenating different traces generated with different random seeds to create a large training dataset of more than 13M symbols, and train the NN with different random 1M symbols from this training dataset for each epoch.
Regression vs Classification	The regression and classification have some statistical and machine learning drawbacks that can impact the learning in the field of nonlinearity mitigation in coherent optical channels.	In all tests in this study, regression has performed better than classification. However, we mention that this check always needs to be done. For fairness reasons, take into account that mini-batch size, learning rate, and the number of epochs need to be individually tuned for each configuration since the gradients are different when using CEL or MSE.
Batch Size Study	The batch size is an important hyperparameter to guarantee a proper learning process of channel equalization. Using small mini-batch sizes, in the regression can make the NN move the learning in a direction that is not the best.	We observed that using a large mini-batch size ($\geq$1000) is good for generalization when using regression for NN-based equalizers. We see that when using small mini-batches, the NN was not generalizing for the different levels of nonlinearity for points in the QAM constellation
Computational Complexity	The computational complexity can be confused with the number of NN parameters. Also, it can be mistakenly underestimated if one uses memory to save some common multiplications. The underestimation can take place as well if quantization and pruning are not accounted for accurately.	We recommend reporting the computational complexity in terms of real multiplications per recovered symbol. In the same way, when using the quantization and pruning, the CC should be reported in terms of BoPs or NABS. Also, when reporting the CC values considering that some computations can be saved to be used later, it needs to be clearly stated that latency is not a constraint.

[11]Using the L^2 regularizer can also make the NN model be a single point attractor in the space of its weights, where the attractor is located at the origin. In such a case, any information that was inserted into the NN model dies out exponentially fast. As described in Ref. [235], this can hide long-term memory traces that impact the learning process in this task.

Chapter 7

Conclusion and future work

It is expected that a capacity bottleneck will occur in the near future because fiber-optic communication, the backbone of the global data network, is under pressure to carry more data every day. Now, more than ever, it is crucial to study alternative information transfer strategies in order to make optimal use of the existing fiber infrastructure. WDM, SDM, PDM, etc., have all been tried in recent decades to boost data rates, each with its own advantages and disadvantages. To combat the signal degradation introduced by fiber, especially the phenomenon known as Kerr nonlinearity, many DSP-based methods have been developed. Fiber channels, unlike more common linear ones like wireless and copper, are distinguished by their nonlinearity. To adapt to the new fiber channel, it is important to rethink what had been created to lessen only the negative effects of the linear channel. This is not a simple task because one must take into account the physics governing fiber distortions as described by the well-known NLSE. This thesis shows the steps to designing a neural network-based equalizer to mitigate the nonlinearity in the optical channel. It started by showing an analysis of the performance pillar, comparing the performance of such NN-based equalizers with the standard DBP method. Then, the computational complexity pillar is presented, where equations to measure the computational complexity of NN structures from a software to a hardware perspective are introduced. After that, a trade-off study between the performance versus the complexity of the most famous NN structures in the literature is produced. Next, the performance of the different compression techniques is presented considering three different transmission setups, comparing the Q-factor versus computational complexity in all scenarios. As the most relevant result, it was observed that when using weight clustering and pruning, as a nonuniform quantization step, for the Sim1 transmission, it presented a Q-factor gain of 1.6 dB compared to the CDC in the case of 3 clustered weights with the cost of increasing the complexity by 182%, and a Q-factor gain of 0.6 dB at the expense of a 61% increase in complexity in the case of 2 clustered weights. This result represents a big step forward in reaching commercial implementation, since it is approaching the computational complexity of the existing CDC block in the DSP chain. Also, here a detailed study of the design of NN-based optical equalizers was carried out, addressing the steps from Python realization to FPGA implementation. The complexity, performance, and resources required for the approximations have been evaluated. The proposed realization showed that the biLSTM requires only ≈ 2.5 times more FPGA resources than the CDC, while still outperforming the 1-StpS DBP in Q-factor.

In the case of flexibility, the applicability of transfer learning methods for the adaptation and reconfiguration of NN-based equalizers in coherent optical systems was investigated. In particular, the potential of TL to reduce the number of training epochs and the training dataset without impacting the equalizer's performance was assessed. Furthermore, the transfer direction was evaluated when the system's launch power, modulation format, and symbol rate are changed. Overall, it could be observed that a reduction of up to 99% in terms of training epochs (required to achieve the best performance in the equalization) and 99% in terms of training dataset size can be achieved without affecting the performance quality of the retrained equalizer.

In the last technical chapter of this thesis, important hidden caveats and pitfalls when dealing with the applications of machine learning methods and, particularly, the NNs for non-linear impairments' compensation in coherent optical communications are presented. This chapter underlines the challenges and common misconception errors that often occur in the development of NN-based equalizers applied in high-speed coherent optical communication systems: a poor model generalization, dependent dataset characteristics (periodicity), overfitting behavior and indications, performance overestimation, and inaccuracy in estimating the computational complexity.

Certainly, for consistency, it is now important to reiterate at the end of this thesis what was stated in the beginning, namely that the NN-based equalizer provides a noticeable enhancement in channel capacity, as indicated by the increase in mutual information. As previously discussed, Table 6.1 presents the results of four different transmission setups, wherein the fourth configuration (i.e., 10×60km; SSMF; 3dBm; 64-QAM; 30 GBd) yielded a mutual information increase of 0.3 bits per symbol relative to the CDC mutual information baseline. Additionally, Fig. 6.9 (b) shows that for a transmission over a 20×50 km SSMF link, utilizing an SC-64-QAM modulation scheme operating at 30 GBd and 3 dBm, the mutual information improvement over CDC is approximately 1 bit per symbol. This outcome clearly demonstrates the considerable potential of NN-based equalizers to enhance the channel capacity, leading to an estimated throughput increase of approximately 21% in the latter case.

Finally, this thesis is concluded by describing some open problems in the design of low complexity NN-based equalizers that were not investigated in this thesis, with the aim of spurring more research effort on advancing the design of machine learning solutions in optical communication systems.

1) **The parallelization problem**. Training and evaluating each node can be very time-consuming in large NNs. This is unquestionably a bottleneck in the development of a high-speed NN design suitable for optical communication applications. A possible solution is to parallelize such models when implementing them in hardware. This topic was partially covered in Ref. [304] for feed-forward layers. If recurrent layers, like the LSTM layer proposed in this work, are in demand for future industrial applications, it would be interesting to investigate how to parallelize them in hardware implementations.

2) **Knowledge distillation**. This is another possible type of compression that was not investigated in this work, but that is receiving increasing attention from the community [305]. The idea behind knowledge distillation is to train a distilled NN model that has many layers and is truly computationally complex, and then use it to train a more compact NN model. An interesting direction for future work would be to figure out how this technique could be used to make NN-based equalizers work better.

3) **Meta Learning Based Compression**. In Ref. [306], the authors have jointly considered network pruning and quantization in an end-to-end meta-learning framework. Other works, e.g., Ref. [307, 308], used meta-learning to learn how to quantize or how to prune the NN structure. This can potentially be a good alternative to the BO technique used in this thesis, which, perhaps, could provide the same or even better solutions, but in a faster manner.

4) **Stabilization of the quantization training** for different transmission scenarios. The effectiveness of quantization, as mentioned in this work, is highly dependent on the difficulty of the transmission equalization task and the learning process. Several authors have already discussed the challenges and possible solutions of the training process for the quantized NN models [208, 309, 310]. To fully understand this application for future industrial applications, e.g., in the optical communications field, a deeper investigation into how to make this training process more stable and faster, independently of the transmission setup, is required.

5) **Probabilistic neural networks**. This can be a potential solution for the issue of signal corruption due to stochastic noise that was not investigated in this thesis.

6) **Time-frequency representation in neural networks**. In this thesis, all the investigation was considered as an input of the NN a memory vector of the complex symbols to be equalized. It is important as well to perform a comparative study with different types of

inputs (e.g., spectrum, triplets, images of the received constellations, etc.) so that one can identify if the performances can be further improved and complexity reduced.

7) **Further flexibility analysis to avoid the need to retrain the hardware NN implementation**. Even though this thesis presented the transfer learning technique, there is still room for investigation on how to improve the NN design for multiple transmission scenarios with no or very limited retraining.

8) **Improvements in the FPGA Realization** In this thesis, it was presented an FPGA implementation using an int32 quantization. So an open question is to push such FPGA to int8 or less, if possible, by using heterogeneous quantization together with quantization-aware training.

9) **New loss function for optical communications**. Investigation of new loss functions that will produce a better learning/training phase of neural network equalizer in the context of using datasets from long-haul optical transmission systems

References

[1] Shaoliang Zhang, Fatih Yaman, Kohei Nakamura, Takanori Inoue, Valey Kamalov, Ljupcho Jovanovski, Vijay Vusirikala, Eduardo Mateo, Yoshihisa Inada, and Ting Wang. Field and lab experimental demonstration of nonlinear impairment compensation using neural networks. *Nature Communications*, 10(1):3033, 2019.

[2] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. BOA: The Bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference GECCO-99*, volume 1, pages 525–532, 1999.

[3] Oleg Sergeevich Sidelnikov, Aleksei Aleksandrovich Redyuk, Stylianos Sygletos, and Mikhail Petrovich Fedoruk. Methods for compensation of nonlinear effects in multichannel data transfer systems based on dynamic neural networks. *Quantum Electronics*, 49(12):1154, 2019.

[4] S. Deligiannidis, A. Bogris, C. Mesaritakis, and Y. Kopsinis. Compensation of fiber nonlinearities in digital coherent systems leveraging long short-term memory neural networks. *Journal of Lightwave Technology*, 38(21):5991–5999, 2020.

[5] Chien-Yu Lin, Antonio Napoli, Bernhard Spinnler, Vincent Sleiffer, Danish Rafique, Maxim Kuschnerov, Marc Bohn, and Bernhard Schmauss. Adaptive digital backpropagation for optical communication systems. In *Optical Fiber Communication Conference*, pages M3C–4. Optical Society of America, 2014.

[6] Dominika Przewlocka-Rus, Syed Shakib Sarwar, H Ekin Sumbul, Yuecheng Li, and Barbara De Salvo. Power-of-two quantization for low bitwidth and hardware compliant neural networks. *arXiv preprint arXiv:2203.05025*, 2022.

[7] E. Ip and J. M. Kahn. Compensation of dispersion and nonlinear impairments using digital backpropagation. *J. Lightw. Technol.*, 26(20):3416–3425, October 2008.

[8] Vck190 evaluation board, 2022. URL https://www.xilinx.com/content/dam/xilinx/support/documents/boards_and_kits/vck190/ug1366-vck190-eval-bd.pdf.

[9] M. Wandel, P. Kristensen, T. Veng, Y. Qian, Q. Le, and L. Gruner-Nielsen. Dispersion compensating fibers for non-zero dispersion fibers. In *Optical Fiber Communication Conference and Exhibit*, pages 327–329, 2002. doi: 10.1109/OFC.2002.1036395.

[10] Ian Giles, Asiri Obeysekara, Rongsheng Chen, Dean Giles, Francesco Poletti, and David Richardson. Fiber LPG mode converters and mode selection technique for multimode SDM. *IEEE Photonics Technology Letters*, 24(21):1922–1925, 2012.

[11] Gabriele Liga, Alex Alvarado, Erik Agrell, and Polina Bayvel. Information rates of next-generation long-haul optical fiber systems using coded modulation. *Journal of Lightwave Technology*, 35(1):113–123, 2017.

[12] R. Mears, L. Reekie, I. Jauncey, and D. Payne. Low-noise erbium-doped fibre amplifier operating at 1.54 μm. *Electronics Letters*, 23(19):1026–1028, 1987.

[13] C. Fludger, T. Duthel, D. Van den Borne, C. Schulien, E. Schmidt, T. Wuth, J. Geyer, E. De Man, G. Khoe, and H. de Waardt. Coherent equalization and POLMUX-RZ-DQPSK for robust 100-GE transmission. *Journal of lightwave technology*, 26(1): 64–72, 2008.

[14] E. Agrell, M. Karlsson, A. Chraplyvy, D. Richardson, P. Krummrich, P. Winzer, K. Roberts, J. Fischer, S. Savory, B. Eggleton, et al. Roadmap of optical communications. *Journal of Optics*, 18(6):063002, 2016.

[15] P. Winzer. High-spectral-efficiency optical modulation formats. *Journal of Lightwave Technology*, 30(24):3824–3835, 2012.

[16] P. Winzer, D. Neilson, and A. Chraplyvy. Fiber-optic transmission and networking: the previous 20 and the next 20 years. *Optics express*, 26(18):24190–24239, 2018.

[17] HS Chung, SK Shin, DW Lee, DW Kim, and Yun Chur Chung. 640Gbit/s (32$\times$ 20Gbit/s) WDM transmission with 0.4 (bit/s)/Hz spectral efficiency using short-period dispersion-managed fibre. *Electronics Letters*, 37(10):1, 2001.

[18] Peter J Winzer, David T Neilson, and Andrew R Chraplyvy. Fiber-optic transmission and networking: the previous 20 and the next 20 years. *Optics express*, 26(18): 24190–24239, 2018.

[19] S. Hranilovic and F. Kschischang. Optical intensity-modulated direct detection channels: signal space and lattice codes. *IEEE Transactions on Information Theory*, 49(6): 1385–1399, 2003.

[20] Han Sun, Mehdi Torbatian, Mehdi Karimi, Robert Maher, Sandy Thomson, Mohsen Tehrani, Yuliang Gao, Ales Kumpera, George Soliman, Aditya Kakkar, et al. 800G DSP ASIC design using probabilistic shaping and digital sub-carrier multiplexing. *Journal of lightwave technology*, 38(17):4744–4756, 2020.

[21] P. Mitra and J. Stark. Nonlinear limits to the information capacity of optical fibre communications. *Nature*, 411(6841):1027, 2001.

[22] R. Essiambre, G. Kramer, P. Winzer, G. Foschini, and B. Goebel. Capacity limits of optical fiber networks. *Journal of Lightwave Technology*, 28(4):662–701, 2010.

[23] Ronen Dar and Peter J Winzer. Nonlinear interference mitigation: Methods and potential gain. *Journal of Lightwave Technology*, 35(4):903–930, 2017.

[24] René-Jean Essiambre, Gerhard Kramer, Peter J Winzer, Gerard J Foschini, and Bernhard Goebel. Capacity limits of optical fiber networks. *Journal of Lightwave Technology*, 28(4):662–701, 2010.

[25] Govind P Agrawal. *Fiber-Optic Communication Systems*. Wiley, 5th edition, 2021. ISBN 978-1-119-73736-0. doi: https://doi.org/10.1016/B978-0-12-397023-7.00002-4. URL https://www.wiley.com/en-us/Fiber+Optic+Communication+Systems%2C+5th+Edition-p-9781119737360#description-section.

[26] Abdelkerim Amari, Octavia A Dobre, Ramachandran Venkatesan, OS Sunish Kumar, Philippe Ciblat, and Yves Jaouën. A survey on fiber nonlinearity compensation for 400 Gb/s and beyond optical communication systems. *IEEE Communications Surveys & Tutorials*, 19(4):3097–3113, 2017.

[27] Ezra Ip and Joseph M. Kahn. Compensation of dispersion and nonlinear impairments using digital backpropagation. *Journal of Lightwave Technology*, 26(20):3416–3425, 2008. ISSN 07338724. doi: 10.1109/JLT.2008.927791.

[28] Ori Golani, Meir Feder, and Mark Shtaif. Kalman-MLSE equalization for NLIN mitigation. *Journal of Lightwave Technology*, 36(12):2541–2550, 2018.

[29] Fernando P. Guiomar, Sofia B. Amado, Nelson J. Muga, Jacklyn D. Reis, António L. Teixeira, and Armando N. Pinto. Simplified volterra series nonlinear equalizer by intra-channel cross-phase modulation oriented pruning. In *39th European Conference and Exhibition on Optical Communication (ECOC 2013)*, pages 1–3, 2013. doi: 10.1049/cp.2013.1450.

[30] M. Al-Khateeb, M. McCarthy, C. Sánchez, and A. Ellis. Nonlinearity compensation using optical phase conjugation deployed in discretely amplified transmission systems. *Optics express*, 26(18):23945–23959, 2018.

[31] Kavita Burse, Ram Narayan Yadav, and SC Shrivastava. Channel equalization using neural networks: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(3):352–357, 2010.

[32] Jagdish C Patra, Pramod K Meher, and Goutam Chakraborty. Nonlinear channel equalization for wireless communication systems using Legendre neural networks. *Signal Processing*, 89(11):2251–2262, 2009.

[33] John C. Cartledge, Fernando P. Guiomar, Frank R. Kschischang, Gabriele Liga, and Metodi P. Yankov. Digital signal processing for fiber nonlinearities [Invited]. *Optics Express*, 25(3), 2017. ISSN 1094-4087. doi: 10.1364/oe.25.001916.

[34] Chenjia Li, Fan Zhang, Yixiao Zhu, Mingxuan Jiang, Zhangyuan Chen, and Chuanchuan Yang. Fiber Nonlinearity Mitigation in Single Carrier 400 G and 800 G Nyquist-WDM Systems. *Journal of Lightwave Technology*, 36(17):3707–3715, 2018. doi: 10.1109/JLT.2018.2844467.

[35] Fangzheng Zhang, Jian Wu Yan Li, Kun Xu, and Jintong Lin. Intra-channel fiber nonlinearity mitigation based on DBP in single channel and WDM coherent optical transmission systems using different pulse shapes. *Frequenz*, 2011.

[36] Robert Maher, Lidia Galdino, Masaki Sato, Tianhua Xu, Kai Shi, Sean Kilmurray, Seb J. Savory, Benn C. Thomsen, Robert I. Killey, and Polina Bayvel. Linear and

nonlinear impairment mitigation in a Nyquist spaced DP-16QAM WDM transmission system with full-field DBP. In *2014 The European Conference on Optical Communication (ECOC)*, pages 1–3, 2014. doi: 10.1109/ECOC.2014.6963971.

[37] Ronen Dar and Peter J Winzer. On the limits of digital back-propagation in fully loaded WDM systems. *IEEE Photonics Technology Letters*, 28(11):1253–1256, 2016.

[38] Tom Sherborne, Benjamin Banks, Daniel Semrau, Robert I Killey, Polina Bayvel, and Domaniç Lavery. On the impact of fixed point hardware for optical fiber nonlinearity compensation algorithms. *Journal of Lightwave Technology*, 36(20):5016–5022, 2018.

[39] Kumar V. Peddanarappagari and Maïté Brandt-Pearce. Volterra series transfer function of single-mode fibers. *Journal of Lightwave Technology*, 15(12), 1997. ISSN 07338724. doi: 10.1109/50.643545.

[40] Maximilian Schaedler, Christian Bluemm, Maxim Kuschnerov, Fabio Pittala, Stefano Calabro, and Stephan Pachnicke. Deep neural network equalization for optical short reach communication. *Applied Sciences*, 9(21):4675, 2019.

[41] Zhaoyi Pan, Benoît Châtelain, Mathieu Chagnon, and David V. Plant. Volterra filtering for nonlinearity impairment mitigation in DP-16QAM and DP-QPSK fiber optic communication systems. *Optics InfoBase Conference Papers*, 1(2):4–6, 2011. ISSN 21622701. doi: 10.1364/nfoec.2011.jtha040.

[42] Antonio Mecozzi and René Jean Essiambre. Nonlinear shannon limit in pseudolinear coherent systems. *Journal of Lightwave Technology*, 30(12), 2012. ISSN 07338724. doi: 10.1109/JLT.2012.2190582.

[43] Zhenning Tao, Liang Dou, Weizhen Yan, Lei Li, Takeshi Hoshida, and Jens C. Rasmussen. Multiplier-free intrachannel nonlinearity compensating algorithm operating at symbol rate. *Journal of Lightwave Technology*, 29(17), 2011. ISSN 07338724. doi: 10.1109/JLT.2011.2160933.

[44] Tomofumi Oyama, Hisao Nakashima, Shoichiro Oda, Tomohiro Yamauchi, Zhenning Tao, Takeshi Hoshida, and Jens C. Rasmussen. Robust and efficient receiver-side compensation method for intra-channel nonlinear effects. In *Optics InfoBase Conference Papers*, 2014. doi: 10.1364/ofc.2014.tu3a.3.

[45] Qunbi Zhuge, Michael Reimer, Andrzej Borowiec, Maurice O'Sullivan, and David V. Plant. Aggressive quantization on perturbation coefficients for nonlinear pre-distortion. In *Optical Fiber Communication Conference, OFC 2014*, 2014. doi: 10.1364/ofc.2014.th4d.7.

[46] Ying Gao, John C. Cartledge, Abdullah S. Karar, Scott S.-H. Yam, Maurice O'Sullivan, Charles Laperle, Andrzej Borowiec, and Kim Roberts. Reducing the complexity of perturbation based nonlinearity pre-compensation using symmetric EDC and pulse shaping. *Optics Express*, 22(2), 2014. ISSN 1094-4087. doi: 10.1364/oe.22.001209.

[47] Machile Learning handbook by Yandex School of Data Analysis (in Russian). https://ml-handbook.ru/, 2022. Accessed: September 28, 2022.

[48] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[50] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

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

[52] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[53] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[54] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[56] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[57] A gentle introduction to torch.autograd. https://pytorch.org/docs/stable/autograd.html, 2022. Accessed: Jule 28, 2022.

[58] Oleg Sidelnikov, Alexey Redyuk, and Stylianos Sygletos. Equalization performance and complexity analysis of dynamic deep neural networks in long haul transmission systems. *Optics Express*, 26(25):32765–32776, 2018.

[59] Charu C Aggarwal. *Neural networks and deep learning*. Springer, 2018.

[60] Dan Claudiu Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[61] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[62] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354 – 377, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2017.10.013.

[63] Yann LeCun and Yoshua Bengio. *Convolutional Networks for Images, Speech, and Time Series*, page 255–258. MIT Press, Cambridge, MA, USA, 1998. ISBN 0262511029.

[64] Vinod Bajaj, Fred Buchali, Mathieu Chagnon, Sander Wahls, and Vahid Aref. Single-channel 1.61 Tb/s Optical Coherent Transmission Enabled by Neural Network-Based Digital Pre-Distortion. In *2020 European Conference on Optical Communications (ECOC)*, pages 1–4. IEEE, 2020.

[65] Abdalraouf Hassan and Ausif Mahmood. Convolutional recurrent deep learning model for sentence classification. *Ieee Access*, 6:13949–13957, 2018.

[66] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[67] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[68] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

[69] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[70] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471, 2000.

[71] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. LSTM fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[72] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016.

[73] Takayuki Osogami, Hiroshi Kajino, and Taro Sekiyama. Bidirectional learning for time-series models with hidden units. In *International Conference on Machine Learning*, pages 2711–2720, 2017.

[74] Xiaoxiao Dai, Xiang Li, Ming Luo, Quan You, and Shaohua Yu. LSTM networks enabled nonlinear equalization in 50-Gb/s PAM-4 transmission links. *Applied Optics*, 58(22):6079–6084, 2019.

[75] Xingyu Lu, Chao Lu, Weixiang Yu, Liang Qiao, Shangyu Liang, Alan Pak Tao Lau, and Nan Chi. Memory-controlled deep LSTM neural network post-equalizer used in high-speed PAM VLC system. *Optics Express*, 27(5):7822–7833, 2019.

[76] Maximilian Schaedler, Fabio Pittala, Georg Böcherer, Christian Bluemm, Maxim Kuschnerov, and Stephan Pachnicke. Recurrent Neural Network Soft-Demapping for Nonlinear ISI in 800Gbit/s DWDM Coherent Optical Transmissions. In *46th European Conference on Optical Communication (ECOC 2020)*, 2020.

[77] Pietro Verzelli, Cesare Alippi, and Lorenzo Livi. Echo state networks with self-normalizing activations on the hyper-sphere. *Scientific reports*, 9(1):1–14, 2019.

[78] Chenxi Sun, Moxian Song, Shenda Hong, and Hongyan Li. A review of designs and applications of echo state networks. *arXiv preprint arXiv:2012.02974*, 2020.

[79] Stenio M Ranzini, Roman Dischler, Francesco Da Ros, Henning Buelow, and Darko Zibar. Experimental investigation of optoelectronic receiver with reservoir computing in short reach optical fiber communications. *Journal of Lightwave Technology*, 2021.

[80] Mariia Sorokina. Dispersion-managed fiber echo state network analogue with high (including THz) bandwidth. *Journal of Lightwave Technology*, 38(12):3209–3213, 2020.

[81] Hai-Peng Ren, Hui-Ping Yin, Chao Bai, and Jun-Liang Yao. Performance improvement of chaotic baseband wireless communication using echo state network. *IEEE Transactions on Communications*, 68(10):6525–6536, 2020.

[82] Qiuyi Wu, Ernest Fokoue, and Dhireesha Kudithipudi. On the statistical challenges of echo state networks and some potential remedies. *arXiv preprint arXiv:1802.07369*, 2018.

[83] Eslam El-Fiky, Alireza Samani, David Patel, Maxime Jacques, Mohamed Sowailem, and David V Plant. 400 Gb/s O-band silicon photonic transmitter for intra-datacenter optical interconnects. *Optics express*, 27(7):10258–10268, 2019.

[84] Xueyuan Luo, Chenglin Bai, Xinyu Chi, Hengying Xu, Yaxuan Fan, Lishan Yang, Peng Qin, Zhiguo Wang, and Xiuhua Lv. Nonlinear impairment compensation using transfer learning-assisted convolutional bidirectional long short-term memory neural network for coherent optical communication systems. *Photonics*, 9(12):919, 2022. ISSN 2304-6732. doi: 10.3390/photonics9120919.

[85] Astrid Barreiro, Gabriele Liga, and Alex Alvarado. Data-driven enhancement of the time-domain first-order regular perturbation model. *arXiv preprint arXiv:2210.05340*, 2022.

[86] Marina M Melek and David Yevick. Nonlinearity mitigation with a perturbation based neural network receiver. *Optical and Quantum Electronics*, 52(10):1–10, 2020.

[87] Marina M Melek and David Yevick. Fiber nonlinearity mitigation with a perturbation based siamese neural network receiver. *Optical Fiber Technology*, 66:102641, 2021.

[88] Chao Li, Yongjun Wang, Jingjing Wang, Haipeng Yao, Xinyu Liu, Ran Gao, Leijing Yang, Hui Xu, Qi Zhang, Pengjie Ma, et al. Convolutional Neural Network-Aided DP-64 QAM Coherent Optical Communication Systems. *Journal of Lightwave Technology*, 40(9):2880–2889, 2022.

[89] Alexey Redyuk, Evgeny Averyanov, Oleg Sidelnikov, Mikhail Fedoruk, and Sergei Turitsyn. Compensation of nonlinear impairments using inverse perturbation theory with reduced complexity. *Journal of Lightwave Technology*, 38(6):1250–1257, 2020.

[90] Hubert Dzieciol, Toshiaki Koike-Akino, Ye Wang, and Kieran Parsons. Inverse regular perturbation with ML-assisted phasor correction for fiber nonlinearity compensation. *Optics Letters*, 47(14):3471–3474, 2022.

[91] Christian Häger and Henry D Pfister. Nonlinear interference mitigation via deep neural networks. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3. IEEE, 2018.

[92] Christian Häger and Henry D Pfister. Physics-based deep learning for fiber-optic communication systems. *IEEE Journal on Selected Areas in Communications*, 39(1): 280–294, 2020.

[93] Bertold Ian Bitachon, Amirhossein Ghazisaeidi, Marco Eppenberger, Benedikt Baeuerle, Masafumi Ayata, and Juerg Leuthold. Deep learning based digital back-propagation demonstrating SNR gain at low complexity in a 1200 km transmission link. *Optics Express*, 28(20):29318–29334, Sep 2020. doi: 10.1364/OE.401667. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-28-20-29318.

[94] Oleg Sidelnikov, Alexey Redyuk, Stylianos Sygletos, Mikhail Fedoruk, and Sergei Turitsyn. Advanced convolutional neural networks for nonlinearity mitigation in long-haul WDM transmission systems. *Journal of Lightwave Technology*, 39(8): 2397–2406, 2021.

[95] Ezra Ip and Joseph M Kahn. Compensation of dispersion and nonlinear impairments using digital backpropagation. *Journal of Lightwave Technology*, 26(20):3416–3425, 2008.

[96] Simone Gaiarin, Auro Michele Perego, Edson Porto da Silva, Francesco Da Ros, and Darko Zibar. Dual-polarization nonlinear Fourier transform-based optical communication system. *Optica*, 5(3):263–270, 2018.

[97] Govind P Agrawal. Nonlinear science at the dawn of the 21st century. *Lecture Notes in Physics*, 542:195–211, 2000.

[98] IS Terekhov, AV Reznichenko, and SK Turitsyn. Calculation of mutual information for nonlinear communication channel at large signal-to-noise ratio. *Physical Review E*, 94(4):042203, 2016.

[99] AV Reznichenko and IS Terekhov. Investigation of nonlinear communication channel with small dispersion via stochastic correlator approach. In *Journal of Physics: Conference Series*, volume 1206, page 012013. IOP Publishing, 2019.

[100] A V Reznichenko and I S Terekhov. Path integral approach to nondispersive optical fiber communication channel. *Entropy*, 22(6):607, May 2020. ISSN 1099-4300. doi: 10.3390/e22060607. URL http://dx.doi.org/10.3390/e22060607.

[101] Vinícius Oliari, Erik Agrell, and Alex Alvarado. Regular perturbation on the group-velocity dispersion parameter for nonlinear fibre-optical communications. *Nature communications*, 11(1):1–11, 2020.

[102] Moriya Nakamura, Yuta Fukumoto, Shotaro Owaki, Takahide Sakamoto, and Naokatsu Yamamoto. Experimental demonstration of SPM compensation using a complex-valued neural network for 40-Gbit/s optical 16QAM signals. *IEICE Communications Express*, 8(8):281–286, 2019.

[103] Taehwan Kim and Tülay Adali. Fully complex multi-layer perceptron network for nonlinear signal processing. *Journal of VLSI signal processing systems for signal, image and video technology*, 32(1-2):29–43, 2002.

[104] Hyunghun Cho, Yongjin Kim, Eunjung Lee, Daeyoung Choi, Yongjae Lee, and Wonjong Rhee. Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks. *IEEE Access*, 8:52588–52608, 2020.

[105] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[106] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

[107] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[108] Tobias A Eriksson, Henning Bülow, and Andreas Leven. Applying neural networks in optical communication systems: possible pitfalls. *IEEE Photonics Technology Letters*, 29(23):2091–2094, 2017.

[109] Pedro J. Freire and Vladislav Neskornuik. Complex-Valued Neural Network Design for Mitigation of Signal Distortions in Optical Links in Highly Nonlinear Regimes - CODE, October 2020. URL https://doi.org/10.5281/zenodo.4065404andhttps://github.com/pedrofreiree/Complex-NN-based-on-NLSE.

[110] Ginni Khanna, Talha Rahman, Erik De Man, Emilio Riccardi, Annachiara Pagano, Anna Chiado Piat, Bernhard Spinnler, Stefano Calabro, Danish Rafique, Uwe Feiste, et al. Comparison of single carrier 200G 4QAM, 8QAM and 16QAM in a WDM field trial demonstration over 612 km SSMF. In *ECOC 2016; 42nd European Conference on Optical Communication*, pages 1–3. VDE, 2016.

[111] Ginni Khanna, Talha Rahman, Erik De Man, Emilio Riccardi, Annachiara Pagano, Anna Chiado Piat, Stefano Calabro, Bernhard Spinnler, Danish Rafique, Uwe Feiste, et al. Single-carrier 400G 64QAM and 128QAM DWDM field trial transmission over metro legacy links. *IEEE Photonics Technology Letters*, 29(2):189–192, 2016.

[112] Antonio Napoli, Pablo Wilke Berenguer, Talha Rahman, Ginni Khanna, Mahdi M. Mezghanni, Lennart Gardian, Emilio Riccardi, Anna Chiadò Piat, Stefano Calabrò, Stefanos Dris, André Richter, Johannes Karl Fischer, Bernd Sommerkorn-Krombholz, and Bernhard Spinnler. Digital pre-compensation techniques enabling high-capacity bandwidth variable transponders. *Optics Communications*, 409:52 – 65, 2018. ISSN 0030-4018. doi: https://doi.org/10.1016/j.optcom.2017.09.062. URL http://www.sciencedirect.com/science/article/pii/S0030401817308465. Advances in modulation and DSP for optical transmission systems.

[113] Maxim Kuschnerov, M Chouayakh, K Piyawanno, B Spinnler, E De Man, P Kainzmaier, Mohammad S Alfiad, A Napoli, and Berthold Lankl. Data-aided versus blind single-carrier coherent receivers. *IEEE Photonics Journal*, 2(3):387–403, 2010.

[114] Andong Li, Minmin Yuan, Chengshi Zheng, and Xiaodong Li. Speech enhancement using progressive learning-based convolutional recurrent neural network. *Applied Acoustics*, 166:107347, 2020.

[115] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

[116] Pedro J. Freire, Vladislav Neskornuik, Antonio Napoli, Bernhard Spinnler, Nelson Costa, Ginni Khanna, Emilio Riccardi, Jaroslaw E. Prilepsky, and Sergei K. Turitsyn. Complex-valued neural network design for mitigation of signal distortions in optical links. *Journal of Lightwave Technology*, 39(6):1696–1705, 2021. doi: 10.1109/JLT.2020.3042414.

[117] Eric Jacobsen and Peter Kootsookos. Fast, accurate frequency estimators [DSP Tips & Tricks]. *IEEE Signal Processing Magazine*, 24(3):123–125, 2007.

[118] Bernhard Spinnler. Equalizer design and complexity for digital coherent receivers. *IEEE Journal of Selected Topics in Quantum Electronics*, 16(5):1180–1192, 2010.

[119] Shahnam Mirzaei, Anup Hosangadi, and Ryan Kastner. FPGA implementation of high speed FIR filters using add and shift method. In *2006 International Conference on Computer Design*, pages 308–313. IEEE, 2006.

[120] SHAHRAM Jahani. Zot-mk: a new algorithm for big integer multiplication. *MSc MSc, Department of Computer Science, Universiti Sains Malaysia, Penang*, 2009.

[121] Chaim Baskin, et al. Uniq: Uniform noise injection for non-uniform quantization of neural networks. *ACM Transactions on Computer Systems (TOCS)*, 37(1–4):1–15, 2021.

[122] Benjamin Hawks, Javier Duarte, Nicholas J Fraser, Alessandro Pappalardo, Nhan Tran, and Yaman Umuroglu. Ps and qs: Quantization-aware pruning for efficient low latency neural network inference. *arXiv preprint arXiv:2102.11289*, 2021.

[123] Junru Wu, Yue Wang, Zhenyu Wu, Zhangyang Wang, Ashok Veeraraghavan, and Yingyan Lin. Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In *International Conference on Machine Learning*, pages 5363–5372. PMLR, 2018.

[124] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*, 2019.

[125] Toshiaki Koike-Akino, Ye Wang, Keisuke Kojima, Kieran Parsons, and Tsuyoshi Yoshida. Zero-Multiplier Sparse DNN Equalization for Fiber-Optic QAM Systems with Probabilistic Amplitude Shaping. In *2021 European Conference on Optical Communications (ECOC)*, pages 1–4. IEEE, 2021.

[126] Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2359–2368, 2021.

[127] Haoran You, Xiaohan Chen, Yongan Zhang, Chaojian Li, Sicheng Li, Zihao Liu, Zhangyang Wang, and Yingyan Lin. Shiftaddnet: A hardware-inspired deep network. *arXiv preprint arXiv:2010.12785*, 2020.

[128] Paolo Gentili, Francesco Piazza, and Aurelio Uncini. Efficient genetic algorithm design for power-of-two FIR filters. In *1995 International conference on acoustics, speech, and signal processing*, volume 2, pages 1268–1271. IEEE, 1995.

[129] Joseph B Evans. Efficient FIR filter architectures suitable for FPGA implementation. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41(7):490–493, 1994.

[130] Wei Rong Lee, Volker Rehbock, Kok Lay Teo, and Lou Caccetta. Frequency-response masking based FIR filter design with power-of-two coefficients and suboptimum PWR. *Journal of Circuits, Systems, and Computers*, 12(05):591–599, 2003.

[131] Pran Kurup and Taher Abbasi. *Logic synthesis using Synopsys®*. Springer Science & Business Media, 2012.

[132] Huan Li and Wenhua Ye. Efficient implementation of FPGA based on Vivado High Level Synthesis. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pages 2810–2813, 2016. doi: 10.1109/CompComm.2016.7925210.

[133] Xilinx Staff. Gate Count Capacity Metrics for FPGAs. *Xilinx Corp., San Jose, CA, Application Note XAPP*, 59, 2020.

[134] Xilinx Inc. UltraScale Architecture DSP Slice, 2021. URL https://docs.xilinx.com/v/u/en-US/ug579-ultrascale-dsp.

[135] Philipp Gysel, Mohammad Motamedi, and Soheil Ghiasi. Hardware-oriented approximation of convolutional neural networks. *arXiv preprint arXiv:1604.03168*, 2016.

[136] Nhan Tran, Benjamin Hawks, Javier M Duarte, Nicholas J Fraser, Alessandro Pappalardo, and Yaman Umuroglu. Ps and qs: Quantization-aware pruning for efficient low latency neural network inference. *Frontiers in Artificial Intelligence*, 4:94, 2021.

[137] S. V. Padmajarani and M. Muralidhar. FPGA IMPLEMENTATION OF MULTIPLIER USING SHIFT AND ADD TECHNIQUE. *International Journal of Advances in Electronics and Computer Science-IJAECS*, 2(9):1–5, 2015.

[138] Vassil S Dimitrov, Kimmo U Jarvinen, and Jithra Adikari. Area-efficient multipliers based on multiple-radix representations. *IEEE Transactions on Computers*, 60(2): 189–201, 2010.

[139] Richard I Hartley. Subexpression sharing in filters using canonic signed digit multipliers. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(10):677–688, 1996.

[140] Vassil Dimitrov, Laurent Imbert, and Andrew Zakaluzny. Multiplication by a constant is sublinear. In *18th IEEE Symposium on Computer Arithmetic (ARITH'07)*, pages 261–268. IEEE, 2007.

[141] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 1D convolutional neural networks and applications: A survey. *Mechanical systems and signal processing*, 151:107398, 2021.

[142] Rahul Dey and Fathi M Salem. Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[143] Herbert Jaeger, Mantas Lukoševičius, Dan Popovici, and Udo Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[144] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *arXiv preprint arXiv:2104.10201*, 2021.

[145] Zhiqiang Que, Yongxin Zhu, Hongxiang Fan, Jiuxi Meng, Xinyu Niu, and Wayne Luk. Mapping Large LSTMs to FPGAs with Weight Reuse. *Journal of Signal Processing Systems*, 92(9):965–979, 2020.

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

[147] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[148] Xin Long, Zongcheng Ben, and Yan Liu. A survey of related research on compression and acceleration of deep neural networks. *Journal of Physics: Conference Series*, 1213:052003, jun 2019. doi: 10.1088/1742-6596/1213/5/052003. URL https://doi.org/10.1088/1742-6596/1213/5/052003.

[149] S. S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi. Brain-inspired wireless communications: Where reservoir computing meets mimo-ofdm. *IEEE Transactions on Neural Networks and Learning Systems*, 29(10):4694–4708, 2018.

[150] Stavros Deligiannidis, Charis Mesaritakis, and Adonis Bogris. Performance and complexity analysis of bi-directional recurrent neural network models versus volterra nonlinear equalizers in digital coherent systems. *Journal of Lightwave Technology*, 39 (18):5791–5798, 2021.

[151] Pedro J. Freire, Yevhenii Osadchuk, Bernhard Spinnler, Antonio Napoli, Wolfgang Schairer, Nelson Costa, Jaroslaw E. Prilepsky, and Sergei K. Turitsyn. Performance versus complexity study of neural network equalizers in coherent optical systems. *Journal of Lightwave Technology*, 39(19):6085–6096, 2021. doi: 10.1109/JLT.2021.3096286.

[152] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[153] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[154] M Augasta and Thangairulappan Kathirvalavakumar. Pruning algorithms of neural networks—a comparative study. *Open Computer Science*, 3(3):105–115, 2013.

[155] Sunil Vadera and Salem Ameen. Methods for pruning deep neural networks. *arXiv preprint arXiv:2011.00241*, 2020.

[156] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[157] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[158] Frederick Tung and Greg Mori. Deep neural network compression by in-parallel pruning-quantization. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):568–579, 2018.

[159] Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint arXiv:2003.02389*, 2020.

[160] Lei Deng, Guoqi Li, Song Han, Luping Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4):485–532, 2020. doi: 10.1109/JPROC.2020.2976475.

[161] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[162] Chun-Yen Chuang, Wei-Fan Chang, Chia-Chien Wei, Ching-Ju Ho, Cheng-Yu Huang, Jin-Wei Shi, Lindor Henrickson, Young-Kai Chen, and Jyehong Chen. Sparse Volterra nonlinear equalizer by employing pruning algorithm for high-speed PAM-4 850-nm VCSEL optical interconnect. In *Optical Fiber Communication Conference*, pages M1F–2. Optical Society of America, 2019.

[163] Wan-Jou Huang, Wei-Fan Chang, Chia-Chien Wei, Jun-Jie Liu, Yi-Ching Chen, Kai-Lun Chi, Chih-Lin Wang, Jin-Wei Shi, and Jyehong Chen. 93% complexity reduction of Volterra nonlinear equalizer by L1-regularization for 112-Gbps PAM-4 850-nm VCSEL optical interconnect. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3. IEEE, 2018.

[164] OS Sunish Kumar, Lutz Lampe, Shenghang Luo, Mrinmoy Jana, Jeebak Mitra, and Chuandong Li. Deep neural network assisted second-order perturbation-based nonlinearity compensation. In *Signal Processing in Photonic Communications*, pages SpF2E–2. Optical Society of America, 2021.

[165] Mingyuan Li, Wenjia Zhang, Qiao Chen, and Zuyuan He. High-throughput hardware deployment of pruned neural network based nonlinear equalization for 100-Gbps short-reach optical interconnect. *Optics Letters*, 46(19):4980–4983, 2021.

[166] Zhiquan Wan, Jianqiang Li, Liang Shu, Ming Luo, Xiang Li, Songnian Fu, and Kun Xu. Nonlinear equalization based on pruned artificial neural networks for 112-Gb/s SSB-PAM4 transmission over 80-km SSMF. *Optics express*, 26(8):10631–10642, 2018.

[167] Wenjia Zhang, Ling Ge, Yanci Zhang, Chenyu Liang, and Zuyuan He. Compressed nonlinear equalizers for 112-gbps optical interconnects: Efficiency and stability. *Sensors*, 20(17):4680, 2020.

[168] Lei Wang, Xiangye Zeng, Jingyi Wang, Dongqi Gao, and Mengshuai Bai. Low-complexity nonlinear equalizer based on artificial neural network for 112 Gbit/s PAM-4 transmission using DML. *Optical Fiber Technology*, 67:102724, 2021.

[169] Ling Ge, Wenjia Zhang, Chenyu Liang, and Zuyuan He. Compressed Neural Network Equalization Based on Iterative Pruning Algorithm for 112-Gbps VCSEL-Enabled Optical Interconnects. *Journal of Lightwave Technology*, 38(6):1323–1329, 2020.

[170] Ahmed Galib Reza and June-Koo Kevin Rhee. Nonlinear equalizer based on neural networks for PAM-4 signal transmission using DML. *IEEE Photonics Technology Letters*, 30(15):1416–1419, 2018.

[171] Hilde JP Weerts, Andreas C Mueller, and Joaquin Vanschoren. Importance of tuning hyperparameters of machine learning algorithms. *arXiv preprint arXiv:2007.07588*, 2020.

[172] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: AutoML for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[173] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[174] Li-Na Wang, Wenxue Liu, Xiang Liu, Guoqiang Zhong, Partha Pratim Roy, Junyu Dong, and Kaizhu Huang. Compressing deep networks by neuron agglomerative clustering. *Sensors*, 20(21):6033, 2020.

[175] Sanghyun Son, Seungjun Nah, and Kyoung Mu Lee. Clustering convolutional kernels to compress deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 216–232, 2018.

[176] Jung Hyun Lee, Jihun Yun, Sung Ju Hwang, and Eunho Yang. Cluster-promoting quantization with bit-drop for minimizing network quantization loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5370–5379, 2021.

[177] Github implementation of the weights clustering algorithm in tensorflow. https://github.com/tensorflow/model-optimization/blob/v0.7.2/tensorflow_model_optimization/python/core/clustering/keras/clustering_algorithm.py#L24-L194, 2022. Accessed: 2022-05-30.

[178] Minsik Cho, Keivan A Vahid, Saurabh Adya, and Mohammad Rastegari. Dkm: Differentiable k-means clustering layer for neural network compression. *arXiv preprint arXiv:2108.12659*, 2021.

[179] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[180] Olivia Weng. Neural network quantization for efficient inference: A survey. *arXiv preprint arXiv:2112.06126*, 2021.

[181] Haoli Bai, Lu Hou, Lifeng Shang, Xin Jiang, Irwin King, and Michael R Lyu. Towards efficient post-training quantization of pre-trained language models. *arXiv preprint arXiv:2109.15082*, 2021.

[182] Raziel Alvarez, Rohit Prabhavalkar, and Anton Bakhtin. On the efficient representation and execution of deep acoustic models. *arXiv preprint arXiv:1607.04683*, 2016.

[183] Javier Duarte, Song Han, Philip Harris, Sergo Jindariani, Edward Kreinar, Benjamin Kreis, Jennifer Ngadiuba, Maurizio Pierini, Ryan Rivera, Nhan Tran, et al. Fast inference of deep neural networks in FPGAs for particle physics. *Journal of Instrumentation*, 13(07):P07027, 2018.

[184] Claudionor N Coelho, Aki Kuusela, Shan Li, Hao Zhuang, Jennifer Ngadiuba, Thea Klaeboe Aarrestad, Vladimir Loncar, Maurizio Pierini, Adrian Alan Pol, and Sioni Summers. Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors. *Nature Machine Intelligence*, pages 1–12, 2021.

[185] Rishabh Goyal, Joaquin Vanschoren, Victor Van Acht, and Stephan Nijssen. Fixed-point quantization of convolutional neural networks for quantized inference on embedded platforms. *arXiv preprint arXiv:2102.02147*, 2021.

[186] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless CNNs with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[187] Shashank Prasanna. Deep learning deployment with NVIDIA TensorRT. *NVIDIA Deep Learning Institute, New York*, 2019.

[188] Xilinx. dnndk user guide. https://www.xilinx.com/support/documentation/sw_manuals/ai_inference/v1_6/ug1327-dnndk-user-guide.pdf.47, 2022. Accessed: 2022-05-30.

[189] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

[190] Prateeth Nayak, David Zhang, and Sek Chai. Bit efficient quantization for deep neural networks. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 52–56. IEEE, 2019.

[191] Hai Victor Habi, Reuven Peretz, Elad Cohen, Lior Dikstein, Oranit Dror, Idit Diamant, Roy H Jennings, and Arnon Netzer. Hptq: Hardware-friendly post training quantization. *arXiv preprint arXiv:2109.09113*, 2021.

[192] Xinyu Zhang, Ian Colbert, Ken Kreutz-Delgado, and Srinjoy Das. Training deep neural networks with joint quantization and pruning of weights and activations. *arXiv preprint arXiv:2110.08271*, 2021.

[193] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.

[194] Noriaki Kaneda, Ziyi Zhu, Chun-Yen Chuang, Amitkumar Mahadevan, Bob Farah, Keren Bergman, Doutje Van Veen, and Vincent Houtsma. FPGA implementation of deep neural network based equalizers for high-speed PON. In *Optical Fiber Communication Conference*, pages T4D–2. Optical Society of America, 2020.

[195] Xiaoan Huang, Dongxu Zhang, Xiaofeng Hu, Chenhui Ye, and Kaibin Zhang. Low-complexity recurrent neural network based equalizer with embedded parallelization for 100-gbit/s/λ pon. *Journal of Lightwave Technology*, 40(5):1353–1359, 2022.

[196] Pinjing He, Feilong Wu, Meng Yang, Aiying Yang, Peng Guo, Yaojun Qiao, and Xiangjun Xin. A Fiber Nonlinearity Compensation Scheme With Complex-Valued Dimension-Reduced Neural Network. *IEEE Photonics Journal*, 13(6):1–7, 2021.

[197] Diego Argüello Ron, Pedro J Freire, Jaroslaw E Prilepsky, Morteza Kamalian-Kopae, Antonio Napoli, and Sergei K Turitsyn. Experimental implementation of a neural network optical channel equalizer in restricted hardware using pruning and quantization. *Scientific Reports*, 12(1):1–14, 2022.

[198] Fayçal Ait Aoudia and Jakob Hoydis. Towards hardware implementation of neural network-based communication algorithms. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2019.

[199] Weihong Xu, Xiaosi Tan, Yuxin Lin, Xiaohu You, Chuan Zhang, and Yair Be'ery. On the efficient design of neural networks in communication systems. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 522–526. IEEE, 2019.

[200] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30, 1998.

[201] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[202] Pedro J. Freire, Sasipim Srivallapanondh, Antonio Napoli, Jaroslaw E. Prilepsky, and Sergei K. Turitsyn. Computational complexity evaluation of neural network applications in signal processing. *arXiv preprint arXiv:2206.12191*, 2022.

[203] Suhap Sahin, Yasar Becerikli, and Suleyman Yazici. Neural network implementation in hardware using FPGAs. In *International conference on neural information processing*, pages 1105–1112. Springer, 2006.

[204] Andrei Dinu, Marcian N. Cirstea, and Silvia E. Cirstea. Direct neural-network hardware-implementation algorithm. *IEEE Transactions on Industrial Electronics*, 57 (5):1845–1848, 2010. doi: 10.1109/TIE.2009.2033097.

[205] Welson Sun, Michael J Wirthlin, and Stephen Neuendorffer. FPGA pipeline synthesis design exploration using module selection and resource sharing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):254–265, 2007.

[206] Pedro J. Freire, Daniel Abode, Jaroslaw E. Prilepsky, Nelson Costa, Bernhard Spinnler, Antonio Napoli, and Sergei K. Turitsyn. Transfer learning for neural networks-based equalizers in coherent optical systems. *Journal of Lightwave Technology*, 39(21): 6733–6745, 2021. doi: 10.1109/JLT.2021.3108006.

[207] Junho Cho and Peter J Winzer. Probabilistic constellation shaping for optical fiber communications. *Journal of Lightwave Technology*, 37(6):1590–1607, 2019.

[208] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training quantized nets: A deeper understanding. *Advances in Neural Information Processing Systems*, 30, 2017.

[209] Pedro J. Freire, Daniel Abode, Jaroslaw E. Prilepsky, Nelson Costa, Bernhard Spinnler, Antonio Napoli, and Sergei K. Turitsyn. Transfer learning for neural networks-based equalizers in coherent optical systems. *Journal of Lightwave Technology*, 39(21): 6733–6745, 2021. doi: 10.1109/JLT.2021.3108006.

[210] Julián Caba, Fernando Rincón, Jesús Barba, A José, Julio Dondo, and Juan C López. Towards test-driven development for FPGA-based modules across abstraction levels. *IEEE Access*, 9:31581–31594, 2021.

[211] Xilinx ug1399 vitis hls, 2022. URL https://docs.xilinx.com/r/en-US/ug1399-vitis-hls.

[212] Strategy implemnetation vivado, 2022. URL https://docs.xilinx.com/r/en-US/ug904-vivado-implementation/Strategy.

[213] Alberto Lopez-Gasso. What are the best vivado synthesis and implementation strategies???, Apr 2021. URL https://miscircuitos.com/vivado-synthesis-and-implementation-strategies.

[214] Michael T Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G Dietterich. To transfer or not to transfer. In *NIPS 2005 workshop on transfer learning*, volume 898, pages 1–4, 2005.

[215] Francesco Musumeci, Virajit G Venkata, Yusuke Hirota, Yoshinari Awaji, Sugang Xu, Masaki Shiraiwa, Biswanath Mukherjee, and Massimo Tornatore. Transfer learning across different lightpaths for failure-cause identification in optical networks. In *2020 European Conference on Optical Communications (ECOC)*, pages 1–4. IEEE, 2020.

[216] Yijun Cheng, Wenkai Zhang, Songnian Fu, Ming Tang, and Deming Liu. Transfer learning simplified multi-task deep neural network for PDM-64QAM optical performance monitoring. *Optics express*, 28(5):7607–7617, 2020.

[217] Weiyang Mo, Yue-Kai Huang, Shaoliang Zhang, Ezra Ip, Daniel C Kilper, Yoshiaki Aono, and Tsutomu Tajima. ANN-based transfer learning for QoT prediction in real-time mixed line-rate systems. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3. IEEE, 2018.

[218] Le Xia, Jing Zhang, Shaohua Hu, Mingyue Zhu, Yingxiong Song, and Kun Qiu. Transfer learning assisted deep neural network for OSNR estimation. *Optics express*, 27(14):19398–19406, 2019.

[219] Zhengguang Gao, Shuangyi Yan, Jiawei Zhang, Marcus Mascarenhas, Reza Nejabati, Yuefeng Ji, and Dimitra Simeonidou. ANN-based multi-channel QoT-prediction over a 563.4-km field-trial testbed. *Journal of Lightwave Technology*, 38(9):2646–2655, 2020.

[220] Qiuyan Yao, Hui Yang, Ao Yu, and Jie Zhang. Transductive transfer learning-based spectrum optimization for resource reservation in seven-core elastic optical networks. *Journal of Lightwave Technology*, 37(16):4164–4172, 2019.

[221] Jing Zhang, Le Xia, Mingyue Zhu, Shaohua Hu, Bo Xu, and Kun Qiu. Fast remodeling for nonlinear distortion mitigation based on transfer learning. *Optics letters*, 44(17): 4243–4246, 2019.

[222] Zhaopeng Xu, Chuanbowen Sun, Tonghui Ji, Jonathan H Manton, and William Shieh. Feedforward and recurrent neural network-based transfer learning for nonlinear equalization in short-reach optical links. *Journal of Lightwave Technology*, 39(2): 475–480, 2020.

[223] Wanting Zhang, Taowei Jin, Tao Xu, Jing Zhang, and Kun Qiu. Nonlinear Mitigation with TL-NN-NLC in Coherent Optical Fiber Communications. In *Asia Communications and Photonics Conference*, pages M4A–321. Optical Society of America, 2020.

[224] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[225] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021. doi: 10.1109/JPROC.2020.3004555.

[226] Feng-Lei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 2021.

[227] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.

[228] Quanshi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *arXiv preprint arXiv:1802.00614*, 2018.

[229] Farhan Qamar, Muhammad Khawar Islam, Romana Shahzadi, Syed Zafar Ali, and Mudassar Ali. 128-QAM dual-polarization chaotic long-haul system performance evaluation. *Journal of Optical Communications*, 2020.

[230] AS Sarwar, F Qamar, and MS Ahmad. PERFORMANCE ANALYSIS OF 128-QAM DUAL POLARIZATION SYSTEM FOR LONG HAUL OPTICAL COMMUNICATION. *Pakistan Journal of Science*, 70(4):324, 2018.

[231] Govind P Agrawal. *Nonlinear Fiber Optics*. Academic Press, Boston, fifth edition, 2013. doi: https://doi.org/10.1016/B978-0-12-397023-7.00002-4. URL http://www.sciencedirect.com/science/article/pii/B9780123970237000024.

[232] Matheus Sena, M Sezer Erkilinc, Thomas Dippon, Behnam Shariati, Robert Emmerich, Johannes Karl Fischer, and Ronald Freund. Bayesian optimization for nonlinear system identification and pre-distortion in cognitive transmitters. *Journal of Lightwave Technology*, 39(15):5008–5020, 2021.

[233] L. Galdino, D. Lavery, Z. Liu, K. Balakier, E. Sillekens, D. Elson, G. Saavedra, R. I. Killey, and P. Bayvel. The trade-off between transceiver capacity and symbol rate. In *2018 Optical Fiber Communications Conference and Exposition (OFC)*, pages 1–3, 2018.

[234] Xinyu Liu, Yongjun Wang, Xishuo Wang, Hui Xu, Chao Li, and Xiangjun Xin. Bi-directional gated recurrent unit neural network based nonlinear equalizer for coherent optical communication system. *Optics Express*, 29(4):5923–5933, 2021.

[235] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[236] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[237] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: A survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[238] Alex Alvarado, Erik Agrell, Domaniç Lavery, Robert Maher, and Polina Bayvel. Replacing the Soft-Decision FEC Limit Paradigm in the Design of Optical Communication Systems. *Journal of Lightwave Technology*, 33(20):4338–4352, 2015. doi: 10.1109/JLT.2015.2450537.

[239] Michael D McKinley, Kate A Remley, Maciej Myslinski, J Stevenson Kenney, Dominique Schreurs, and Bart Nauwelaers. EVM calculation for broadband modulated signals. In *64th ARFTG Conf. Dig*, pages 45–52. Orlando, 2004.

[240] Rene Schmogrow, Bernd Nebendahl, Marcus Winter, Arne Josten, David Hillerkuss, Swen Koenig, Joachim Meyer, Michael Dreschmann, Michael Huebner, Christian Koos, et al. Error vector magnitude as a performance measure for advanced modulation formats. *IEEE Photonics Technology Letters*, 24(1):61–63, 2011.

[241] Wolfgang Freude, René Schmogrow, Bernd Nebendahl, Marcus Winter, Arne Josten, David Hillerkuss, Swen Koenig, Joachim Meyer, Michael Dreschmann, and Michael Huebner. Quality metrics for optical signals: Eye diagram, Q-factor, OSNR, EVM and BER. In *2012 14th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4. IEEE, 2012.

[242] Rishad Ahmed Shafik, Md Shahriar Rahman, and AHM Razibul Islam. On the extended relationships among EVM, BER and SNR as performance metrics. In *2006 International Conference on Electrical and Computer Engineering*, pages 408–411. IEEE, 2006.

[243] Tobias A Eriksson, Tobias Fehenberger, and Wilfried Idler. Characterization of nonlinear fiber interactions using multidimensional mutual information over time and polarization. *Journal of Lightwave Technology*, 35(6):1204–1210, 2017.

[244] Tobias A Eriksson, Tobias Fehenberger, Norbert Hanik, Peter A Andrekson, Magnus Karlsson, and Erik Agrell. Four-dimensional estimates of mutual information in coherent optical communication experiments. In *2015 European Conference on Optical Communication (ECOC)*, pages 1–3. IEEE, 2015.

[245] Tommaso Catuogno, Menelaos Ralli Camara, and Marco Secondini. Non-parametric estimation of mutual information with application to nonlinear optical fibers. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 736–740. IEEE, 2018.

[246] Peter J Rousseeuw and Katrien Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.

[247] Oliver Kramer. Scikit-learn. In *Machine learning for evolution strategies*, pages 45–53. Springer, 2016.

[248] Mutsam A Jarajreh, Elias Giacoumidis, Ivan Aldaya, Son Thai Le, Athanasios Tsokanos, Zabih Ghassemlooy, and Nick J Doran. Artificial neural network nonlinear equalizer for coherent optical ofdm. *IEEE Photonics Technology Letters*, 27(4): 387–390, 2014.

[249] Jing Zhang, Pingping Lei, Shaohua Hu, Mingyue Zhu, Zhenming Yu, Bo Xu, and Kun Qiu. Functional-link neural network for nonlinear equalizer in coherent optical fiber communications. *IEEE Access*, 7:149900–149907, 2019.

[250] Ling Liu, Meihua Bi, Shilin Xiao, Jiafei Fang, Tiancheng Huang, and Weisheng Hu. Ols-based rbf neural network for nonlinear and linear impairments compensation in the co-ofdm system. *IEEE Photonics Journal*, 10(2):1–8, 2018.

[251] Siming Liu, Peng-Chun Peng, Chin-Wei Hsu, Shuo Chen, Huiping Tian, and Gee-Kung Chang. An Effective Artificial Neural Network Equalizer with S-shape Activation Function for High-speed 16-QAM Transmissions using Low-cost Directly Modulated Laser. In *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 269–273. IEEE, 2018.

[252] Jerart L Julus, D Manimegalai, Asha C Beaula, Joshan J Athanesious, and Andrew A Roobert. Advanced nonlinear equalizer for filter bank multicarrier-based long reach-passive optical network system. *International Journal of Communication Systems*, 34 (14):e4921, 2021. doi: https://doi.org/10.1002/dac.4921. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4921.

[253] Oleksandr Kotlyar, Morteza Kamalian-Kopae, Maryna Pankratova, Anastasiia Vasylchenkova, Jaroslaw E. Prilepsky, and Sergei K. Turitsyn. Convolutional long short-term memory neural network equalizer for nonlinear Fourier transform-based optical transmission systems. *Opt. Express*, 29(7):11254–11267, Mar 2021. doi: 10.1364/OE.419314. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-29-7-11254.

[254] Hao Ming, Xinyu Chen, Xiansong Fang, Lei Zhang, Chenjia Li, and Fan Zhang. Ultralow complexity long short-term memory network for fiber nonlinearity mitigation in coherent optical communication systems. *Journal of Lightwave Technology*, pages 1–1, 2022. doi: 10.1109/JLT.2022.3141404.

[255] Irshaad Fatadin. Estimation of BER from Error Vector Magnitude for Optical Coherent Systems. *Photonics*, 3(2), 2016. ISSN 2304-6732. doi: 10.3390/photonics3020021. URL https://www.mdpi.com/2304-6732/3/2/21.

[256] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[257] Zheng Yang, Fan Gao, Songnian Fu, Ming Tang, and Deming Liu. Overfitting effect of artificial neural network based nonlinear equalizer: from mathematical origin to transmission evolution. *Science China Information Sciences*, 63:1–13, 2020.

[258] Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[259] Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

[260] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*, 2015.

[261] Chuan Wang and J.C. Principe. Training neural networks with additive noise in the desired signal. *IEEE Transactions on Neural Networks*, 10(6):1511–1517, 1999. doi: 10.1109/72.809097.

[262] Yves Grandvalet, Stéphane Canu, and Stéphane Boucheron. Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108, 1997. doi: 10.1162/neco.1997.9.5.1093.

[263] Shi Yin, Chao Liu, Zhiyong Zhang, Yiye Lin, Dong Wang, Javier Tejedor, Thomas Fang Zheng, and Yinguo Li. Noisy training for deep neural networks in speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015 (1):1–14, 2015.

[264] Salah Rifai, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*, 2011.

[265] Warick M Brown, Tamás D Gedeon, and David I Groves. Use of noise to augment training data: a neural network method of mineral–potential mapping in regions of limited known deposit examples. *Natural Resources Research*, 12(2):141–152, 2003.

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

[267] Timothy O'Shea and Jakob Hoydis. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017. doi: 10.1109/TCCN.2017.2758370.

[268] Sebastian Dörner, Sebastian Cammerer, Jakob Hoydis, and Stephan Ten Brink. Deep learning based communication over the air. *IEEE Journal of Selected Topics in Signal Processing*, 12(1):132–143, 2017.

[269] Maximilian Schädler, Georg Böcherer, and Stephan Pachnicke. Soft-demapping for short reach optical communication: A comparison of deep neural networks and volterra series. *Journal of Lightwave Technology*, 39(10):3095–3105, 2021. doi: 10.1109/JLT.2021.3056869.

[270] Vahid Aref and Mathieu Chagnon. End-to-end learning of joint geometric and probabilistic constellation shaping. In *2022 Optical Fiber Communications Conference and Exhibition (OFC)*, 2022.

[271] Pavel Golik, Patrick Doetsch, and Hermann Ney. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, volume 13, pages 1756–1760, 2013.

[272] Kamil Nar, Orhan Ocal, S Shankar Sastry, and Kannan Ramchandran. Cross-entropy loss leads to poor margins. *Rejected-ICLR 2019*, 2018.

[273] Anna Sergeevna Bosman, Andries Engelbrecht, and Mardé Helbig. Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing*, 400:113–136, 2020.

[274] Hussein Aly Kamel Rady. Shannon entropy and mean square errors for speeding the convergence of multilayer neural networks: A comparative approach. *Egyptian Informatics Journal*, 12(3):197–209, 2011.

[275] Le Nguyen Binh. *Noises in optical communications and photonic systems*. CRC Press, 2019.

[276] Parastoo Sadeghi, Pascal O Vontobel, and Ramtin Shams. Optimization of information rate upper and lower bounds for channels with memory. *IEEE Transactions on Information theory*, 55(2):663–688, 2009.

[277] Georg Böcherer, Patrick Schulte, and Fabian Steiner. Probabilistic shaping and forward error correction for fiber-optic communication systems. *Journal of Lightwave Technology*, 37(2):230–244, 2019.

[278] Neri Merhav, Gideon Kaplan, Amos Lapidoth, and S Shamai Shitz. On information rates for mismatched decoders. *IEEE Transactions on Information Theory*, 40(6): 1953–1967, 1994.

[279] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8792–8802, Red Hook, NY, USA, 2018. Curran Associates Inc.

[280] Georg Böcherer. Lecture notes on machine learning for communications. http://georg-boecherer.de/mlcomm, 2022. Accessed: 2022-09-30.

[281] Michael Pazzani, Christopher Merz, Patrick Murphy, Kamal Ali, Timothy Hume, and Clifford Brunk. Reducing misclassification costs. In *Machine Learning Proceedings 1994*, pages 217–225. Elsevier, 1994.

[282] Pedro Antonio Gutiérrez, Maria Perez-Ortiz, Javier Sanchez-Monedero, Francisco Fernandez-Navarro, and Cesar Hervas-Martinez. Ordinal regression methods: survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering*, 28 (1):127–146, 2015.

[283] Stéphane Lathuilière, Pablo Mesejo, Xavier Alameda-Pineda, and Radu Horaud. A comprehensive analysis of deep regression. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2065–2081, 2019.

[284] Shulun Wang, Feng Liu, and Bin Liu. Escaping the gradient vanishing: Periodic alternatives of softmax in attention mechanism. *IEEE Access*, 9:168749–168759, 2021. doi: 10.1109/ACCESS.2021.3138201.

[285] Zekun Niu, Hang Yang, Haochen Zhao, Chenhao Dai, Weisheng Hu, and Lilin Yi. End-to-end deep learning for long-haul fiber transmission using differentiable surrogate channel. *Journal of Lightwave Technology*, pages 1–1, 2022. doi: 10.1109/JLT.2022.3148270.

[286] Pedro J. Freire, Jaroslaw E. Prilepsky, Yevhenii Osadchuk, Sergei K. Turitsyn, and Vahid Aref. Deep neural network-aided soft-demapping in optical coherent systems: Regression versus classification. *arXiv preprint arXiv:2109.13843*, 2021.

[287] Yanghao Li, Cuiling Lan, Junliang Xing, Wenjun Zeng, Chunfeng Yuan, and Jiaying Liu. Online human action detection using joint classification-regression recurrent neural networks. In *European conference on computer vision*, pages 203–220. Springer, 2016.

[288] Lopamudra Mukherjee, Md Abdul Kader Sagar, Jonathan N Ouellette, Jyoti J Watters, and Kevin W Eliceiri. Joint regression-classification deep learning framework for analyzing fluorescence lifetime images using NADH and FAD. *Biomedical Optics Express*, 12(5):2703–2719, 2021.

[289] Mingxia Liu, Jun Zhang, Ehsan Adeli, and Dinggang Shen. Joint classification and regression via deep multi-task multi-channel learning for Alzheimer's disease diagnosis. *IEEE Transactions on Biomedical Engineering*, 66(5):1195–1206, 2018.

[290] Mingxia Liu, Jun Zhang, and et. al. Deep multi-task multi-channel learning for joint classification and regression of brain status. In *International conference on medical image computing and computer-assisted intervention*, pages 3–11. Springer, 2017.

[291] Ji-Yan Wu, Min Wu, Zhenghua Chen, Xiaoli Li, and Ruqiang Yan. A joint classification-regression method for multi-stage remaining useful life prediction. *Journal of Manufacturing Systems*, 58:109–119, 2021.

[292] Jing Chen, Long Cheng, Xi Yang, Jun Liang, Bing Quan, and Shoushan Li. Joint learning with both classification and regression models for age prediction. In *Journal of Physics: Conference Series*, volume 1168, page 032016. IOP Publishing, 2019.

[293] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[294] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[295] D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003. ISSN 08936080. doi: 10.1016/S0893-6080(03)00138-2.

[296] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent, 2017. URL https://arxiv.org/abs/1710.06451.

[297] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima, 2016. URL https://arxiv.org/abs/1609.04836.

[298] Linning Peng, Maryline Hélard, and Sylvain Haese. On bit-loading for discrete multitone transmission over short range POF systems. *Journal of lightwave technology*, 31 (24):4155–4165, 2013.

[299] Jong-Wan Kim and Chang-Hee Lee. Modulation format identification of square and non-square M-QAM signals based on amplitude variance and OSNR. *Optics Communications*, 474:126084, 2020.

[300] Laia Nadal, Josep Maria Fàbrega, Javier Vílchez, and Michela Svaluto Moreolo. Experimental analysis of 8-QAM constellations for adaptive optical OFDM systems. *IEEE Photonics Technology Letters*, 28(4):445–448, 2015.

[301] Jingjing Wang, Na Li, Wei Shi, Yangyang Ma, Xiulong Liang, and Xinli Dong. Capacity of 60 GHz wireless communications based on QAM. *Journal of Applied Mathematics*, 2014(SI03):1 – 5, 2014. doi: 10.1155/2014/815617. URL https://doi.org/10.1155/2014/815617.

[302] Jason Brownlee. *Better Deep Learning. Train Faster, Reduce Overfitting, and Make Better Predictions*, volume 1. Machine Learning Mastery, 2018.

[303] Sunghwan Ong, Cheolwoo You, Sooyong Choi, and Daesik Hong. A decision feedback recurrent neural equalizer as an infinite impulse response filter. *IEEE Transactions on Signal Processing*, 45(11):2851–2858, 1997. doi: 10.1109/78.650112.

[304] Janardan Misra and Indranil Saha. Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, 74(1-3):239–255, 2010.

[305] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[306] Jianming Ye, Shiliang Zhang, and Jingdong Wang. Hybrid network compression via meta-learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1423–1431, 2021.

[307] Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. Metaquant: Learning to quantize by learning to penetrate non-differentiable quantization. *Advances in Neural Information Processing Systems*, 32, 2019.

[308] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3296–3305, 2019.

[309] Burak Bartan and Mert Pilanci. Training quantized neural networks to global optimality via semidefinite programming. In *International Conference on Machine Learning*, pages 694–704. PMLR, 2021.

[310] Bohan Zhuang, Lingqiao Liu, Mingkui Tan, Chunhua Shen, and Ian Reid. Training quantized neural networks with a full-precision auxiliary module. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1488–1497, 2020.