Complexity Reduction for Deep Machine Learning-Based Optical Transmission Systems

Sasipim Srivallapanondh

Doctor of Philosophy

Aston Institute of Photonic Technologies
Aston University

March 2025

Aston University

Complexity Reduction for Deep Machine Learning-Based Optical Transmission Systems

Sasipim Srivallapanondh

Doctor of Philosophy

2025

Abstract

Over the past decade, there has been a massive increase in demand for bandwidth to serve bandwidth-hungry applications, for example, video calls, the Internet of Things (IoT), and 5G/6G. Many of these applications require not only high speed but also low latency. It is widely known that the majority of digital data is transmitted over optical fibers, resulting in a national and international infrastructure. However, fiber nonlinearity (e.g., the Kerr effect) imposes significant limitations on the optical launch power; as a result, it constrains the information rate in modern coherent transmission systems. To address these challenges, the development of innovative system designs is required, for instance, advanced modulation formats, wideband transmission, new fiber types and enhanced digital signal processing (DSP) techniques to mitigate fiber nonlinearity.

Mitigating fiber nonlinearity is essential to achieve higher transmission rates and improved signal quality, without the need for new infrastructure. Various techniques have been proposed, including traditional methods like digital backpropagation (DBP) and the Volterra series-based approach. However, the computational complexity is still the main challenge, encouraging the researchers to seek an alternative approach like machine learning (ML). ML, especially neural networks (NNs), has demonstrated its capability in a wide range of applications due to the universal approximation capability of NNs. NNs have been intensively studied for the optical channel post-equalization, because they can accurately approximate the inverse optical channel transfer function and reverse the nonlinear distortions. Despite their promising equalization performance, the limitations of the NN-based equalizers in real implementation still remain. The major challenges include the computational complexity, the parallelizability, and the generalizability.

This thesis investigates the integration of NN-based equalizers for nonlinear impairment mitigation in coherent optical long-haul communication systems. By leveraging NNs, this work aims to improve transmission quality while focusing on the three major aspects of the challenges in NN-based equalizers. This thesis contains some key contributions: i) the investigation of computational complexity reduction techniques, including weight clustering, and activation function approximation; ii) parallelization strategies using knowledge distillation to facilitate real-time inference; iii) the application of multi-task learning frameworks to improve model flexibility and adaptability in dynamic network conditions; and iv) the validation of these methods based on theoretical and experimental data. The comprehensive analysis of this thesis highlights the performance-complexity trade-offs, practical feasibility, and potential of NN-based equalizers. Finally, the results show that the NN-based equalizers can improve the quality of transmission, while keeping the complexity the same or lower than the traditional DSP algorithm, offering a promising approach for future optical networks.

Keywords: Coherent Optical Communications, Nonlinearity Equalization, Digital Signal Processing, Machine Learning, Computational Complexity

To my beloved family, whose unwavering support and encouragement have been my greatest strength.

Declaration

I hereby declare that, except where explicit reference is made to the work of others, the content of this thesis is entirely original. It has not been submitted, in whole or in part, for consideration for any other degree or qualification at this or any other university. This thesis represents my own work. Any material produced in collaboration with others will be explicitly acknowledged in the text and Acknowledgements.

Sasipim Srivallapanondh
March 2025

Acknowledgements

First and foremost, I would like to thank EU Horizon 2020 and the MENTOR project for their generous funding, which made this research possible. I am especially grateful to my supervisors, Professor Sergei K. Turitsyn and Dr. Yaroslav Prilepsky, for giving me the opportunity to carry out my research at AIPT and for their continuous guidance and support throughout my PhD. Their advice, encouragement, and insightful feedback have been invaluable from the very beginning to the very end of this journey.

I would also like to sincerely thank Dr. Antonio Napoli, my industrial supervisor from Infinera (now Nokia), for welcoming me into the team and fostering such a supportive work environment. His encouragement and guidance played a key role in shaping my PhD journey and ensuring the success of my secondment.

A very special thanks goes to Dr. Pedro Freire, whose contribution to my PhD has been truly invaluable. He generously shared his knowledge, provided training, offered hands-on experience, and gave me the motivation I needed along the way. Without him, this journey would have been far more difficult. Beyond being a great colleague, he has also been a wonderful friend, always supportive both inside and outside of work.

I am also grateful to Dr. Bernhard Spinnler and Dr. Nelson Costa from Infinera (now Nokia) for their helpful feedback, thoughtful discussions, and generosity in sharing their expertise. My sincere thanks as well to Dr. Giuseppe Parisi, Dr. Pablo Torres-Ferrera, Dr. Ulrich Gaubatz, and many other colleagues at Infinera for their continuous support and assistance with lab experiments. I truly appreciated their patience, guidance, and willingness to explain.

My heartfelt appreciation also goes to all my friends in Birmingham and colleagues at Aston University for the great times we shared in the UK: Nelson, Eliza, Dini, Diana, Mariia, Gelraldo, Egor, Olya, Stepan, Negar, Wing, Karina, Long, Minji, the bouldering gang (Dani, Marta, Carmen, Alberto, and Viktor), and the Eagle boxing gym gang, among many others. All of you made Birmingham feel like home to me. I am equally thankful to my friends in Munich and colleagues from Infinera: Fern, Mark, my amazing roomies (Aleja and Veena), the awesome Chiemsee gang, Mariano, Miquel, Marina,

Alberto, Esteban, Tiago, Mario, Mohammad, Carlos, Max, and many more, for the unforgettable memories and the best time. Moving to Germany just became so easy because of you all. To all my beloved friends from Thailand, thank you for always being there despite the distance. Every time I talk to you guys, it always feels like family, and I miss you all dearly. Your friendship and support made my PhD journey so much more enjoyable, and I will always treasure the moments we shared and the encouragement you gave me during challenging times.

Finally, I would like to express my deepest love and gratitude to my family—especially my dad, mom, and sister (Sasicha)—for their endless love, encouragement, and belief in me. They have stood by me through everything, and I could not have done this without them. They are, and will always be, my greatest strength. A heartfelt thank you also goes to Sami, for always cheering me up, supporting me through the hard times, and being a constant source of love and inspiration. I feel so lucky to have met someone as kind and caring as you. Lastly, my gratitude extends to Sami's family, for their warmth and for welcoming me so lovingly as part of their family.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

1D One-Dimensional

ADC Analog-to-Digital Converter

AI Artificial Intelligence

ASE Amplified Spontaneous Emission

BER Bit Error Rate

biLSTM Bidirectional Long-Short Term Memory

biRNN Bidirectional Recurrent Neural Network

CDC Chromatic Dispersion Compensation

CNN Convolutional Neural Network

CPU Central Processing Unit

DAC Digital-to-Analog Converter

DBP Digital Backpropagation

DP Dual-Polarization

DSCM Digital Subcarrier Multiplexing

DSP Digital Signal Processing

DWDM Dense Wavelength Division Multiplexing

EDFA Erbium-Doped Fiber Amplifier

FF Flip Flop

FIR Finite Impulse Response

FWM Four-Wave Mixing

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

IIR Infinite Impulse Response

IM/DD Intensity Modulation Direct Detection

IoT Internet of Things

IQ In-Phase and Quadrature

KD Knowledge Distillation

LEAF Large Effective Area Fiber

LO Local Oscillator

LUT Look-Up Table

MIMO Multiple Input Multiple Output

ML Machine Learning

MLP Multi-Layer Perceptron

MSE Mean Squared Error

MTL Multi-Task Learning

NLSE Nonlinear Schrödinger Equation

ANN Artifcial Neural Network

NN Neural Network

OFDM Orthogonal Frequency Division Multiplexing

PRBS Pseudo-Random Binary Sequence

PWL Piecewise Linear

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

ROADM Re-Configurable Add-Drop Multiplexers

RRC Root-Raised Cosine

SDM Space Division Multiplexing

SPM Self-Phase Modulation

SSFM Split-Step Fourier Method

SSMF Standard Single-Mode Fiber

STL Single-Task Learning

SVM Support Vector Machine

Tanh Hyperbolic Tangent

TL Transfer Learning

VSTF Voltera Series Transfer Function

WC Weight Cluster

WDM Wavelength Division Multiplexing

XPM Cross-Phase Modulation

Chapter 1

Introduction

1.1 Optical Communication Networks

In the past decade, demand for bandwidth has grown significantly, driven by bandwidth-intensive applications such as video calls, the Internet of Things (IoT), cloud computing and 5G/6G network [2, 3]. Many of these applications also require high speed and low latency to function efficiently. Internet traffic is expected to continuously increase in the coming year, especially with the increasing number of IoTs and the rise of artificial intelligence (AI) applications, including generative AI like ChatGPT that require big data processing and cloud computing.

The majority of digital data is transmitted over optical fibers, resulting in the backbone of national and international communication infrastructures [4, 5]. To satisfy the growing bandwidth demands, various advancements in optical communication technologies have been developed. Starting from the key milestone of the low-loss fiber, the fiber attenuation was reduced to 0.2 dB/km at 1.55 μm in 1979 [6], which significantly extended the reach of optical signals. Then, the development of the Erbium-Doped Fiber Amplifier (EDFA) in 1987 [7] came into play and enabled long-haul optical transmission. Around the same period, wavelength-division multiplexing (WDM) technology was also invented [8]. The WDM allows independent signals to be transmitted over one fiber at different wavelengths, resulting in an increased transmission capacity of optical communications. In the early days, optical communication relied only on the Intensity Modulation and Direct Detection (IM/DD) system. The IM/DD systems are simple and cost-effective, and are in use even today. Later on, in 2005, coherent optical communication gained more attention due to the demonstration of digital carrier-phase estimation in coherent receivers [9]. Coherent communication

utilizes full information on in-phase and quadrature (IQ) components (or amplitude and phase) of the complex amplitude of the optical electric field. This characteristic of coherent systems allows higher modulation formats to cope with the rising data rates.

In current long-haul coherent optical transmission systems, the optical fiber nonlinearity seriously causes a significant challenge by limiting the information rate. In addition, with the ever-increasing transmission bandwidth, the impact of nonlinearity becomes even more crucial [4]. Significant efforts are being directed toward developing new types of optical fibers, for example, hollow-core fibers [10] which provide reduced nonlinearity or the multi-core or multimode fibers that allow space division multiplexing (SDM) [11]. However, these new types of fibers necessitate a complete overhaul of the existing infrastructure. To exploit the current infrastructure, higher-order modulation formats are employed to boost the transmission rate. When the modulation format increases, the signal-to-noise ratio requirement is higher to maintain the same BER. This happens because higher modulation order adopts denser constellation points, making them more susceptible to noise and distortion. This results in a higher demand for optical launch power, and the increased power introduces the nonlinear impairments.

To achieve high data-rate transmission, various digital signal processing (DSP) techniques have been proposed to mitigate some types of signal impairments in coherent systems [12]. The signal impairments include linear effects such as chromatic dispersion and nonlinear effects like Kerr effects [13]. The signal impairments can be observed from the nonlinear Schrödinger equation (NLSE) [14]. The NLSE is a fundamental equation to describe the light propagation down the optical fiber. The NLSE can be directly derived from the Maxwell equations [15], which represent the foundations of electricity and magnetism [16]. The NLSE is formulated as:

$$\frac{\partial u(z,t)}{\partial z} = (\hat{D} + \hat{N})u(z,t), \tag{1.1}$$

where $u(z,t)$ denotes the electrical field which is a function of the propagation distance z and time t. $\hat{D}$ and $\hat{N}$, represent the linear and nonlinear parts of the NLSE, respectively, which are shown as:

$$\hat{D} = \underbrace{-\frac{\alpha}{2}}_{\text{loss}} \underbrace{-\frac{j\beta_2}{2}\frac{\partial^2}{\partial t^2}}_{\text{GVD}} + \underbrace{\frac{j\beta_3}{6}\frac{\partial^3}{\partial t^3}}_{\text{GVD slope}}, \tag{1.2}$$

$$\hat{N} = \underbrace{j\gamma|u(z,t)|^2}_{\text{Kerr effect}},$$

where α, $\beta_{2,3}$, and γ are the attenuation, the group velocity dispersion (GVD) parameter, and the nonlinear coefficient, respectively. As a result, the explicit form of the NLSE is as follows:

$$\frac{\partial u(z,t)}{\partial z}+\frac{\alpha}{2}u(z,t)+\frac{j\beta_2}{2}\frac{\partial^2 u(z,t)}{\partial t^2}-\frac{j\beta_3}{6}\frac{\partial^3 u(z,t)}{\partial t^3}=j\gamma|u(z,t)|^2u(z,t). \tag{1.3}$$

The Eq. (1.3) is used to model the single-polarization transmission, for example, the direct-detection systems deploying the intensity modulation [14]. However, in the case of coherent systems, the dual polarization (DP) signal is detected by the coherent transceiver deploying the advanced DSP. The spectral efficiency of the coherent system is doubled. DP is also considered in a vectorized form. The Eq. (1.3) is then extended to the Manakov equation which is given as:

$$\begin{aligned}\frac{\partial u_X(z,t)}{\partial z}=&\underbrace{-\frac{\alpha}{2}u_X(z,t)+\frac{j\beta_2}{2}\frac{\partial^2}{\partial t^2}u_X(z,t)-\frac{j\beta_3}{6}\frac{\partial^3}{\partial t^3}u_X(z,t)}_{\text{linear}}\\&\underbrace{-j\gamma\frac{8}{9}\left(|u_X(z,t)|^2+|u_Y(z,t)|^2\right)u_X(z,t),}_{\text{nonlinear}}\\\frac{\partial u_Y(z,t)}{\partial z}=&-\frac{\alpha}{2}u_Y(z,t)+\frac{j\beta_2}{2}\frac{\partial^2}{\partial t^2}u_Y(z,t)-\frac{j\beta_3}{6}\frac{\partial^3}{\partial t^3}u_Y(z,t)\\&-j\gamma\frac{8}{9}\left(|u_X(z,t)|^2+|u_Y(z,t)|^2\right)u_Y(z,t).\end{aligned} \tag{1.4}$$

where $u_X(z,t)$ and $u_Y(z,t)$ denote the two orthogonal polarization components of the electric field $u(z,t)$.

To mitigate fiber impairments like chromatic dispersion and nonlinear impairments, one needs a solution of the inverse Manakov equation with inverse optical link parameters. The solution of the propagation equation is analytically possible only for particular scenarios, for instance, zero-dispersion transmission. Thus, numerical solutions, such as the split-step Fourier method (SSFM) [14], are often necessary. The SSFM solves the NLSE by iteratively applying linear (dispersion) and nonlinear (Kerr effect) operators in small propagation steps, alternating between the time and frequency domains using the Fast Fourier Transform (FFT). The technique, like digital backpropagation (DBP) [17] comes into play to solve the inverse NLSE using the SSFM. This approach calculates the transmitted signal from the received signal. However, the main challenge of DBP is computational complexity.

In recent years, machine learning (ML), especially neural networks (NNs), has demonstrated its capability in a wide range of applications, for instance, pattern recognition, signal reconstruction, and time series analysis, etc [18, 19]. In addition, NNs have proven to solve various signal processing tasks, especially in the areas related to nonlinear signal processing, real-time signal processing and adaptive signal processing [20–22]. The NNs have also made their way to the field of optical communications and nonlinearity mitigation [23]. This thesis focuses on the application of NNs in the channel equalization problem in coherent optical transmission systems.

1.2 Motivation

Due to the universal approximation capability of NNs, NNs have been intensively studied for the optical channel post-equalization. The NNs can approximate the inverse optical channel transfer function with reasonable accuracy and revert the nonlinear distortions. However, despite their potential, significant challenges remain in implementing the NN-based equalizers in real hardware. The primary limitations are the computational complexity, inference latency, and flexibility [24, 25].

- Computational Complexity: The complexity of the model is not only reflected by the number of trainable parameters or the number of multiplications required by the model, but also by the model size, memory requirement, and the hardware resources needed for real-time inference. In the optical systems, resource constraints, and power efficiency are critical.

- Inference Latency: Ultra-low latency is one of the main requirements for real-time high-speed communication systems. In certain NN architectures, such as bi-directional long short-term memory (biLSTM) or recurrent NNs (RNN), sequential operations may cause delays during the inference phase. Their limited parallelizability prevents the NN-based equalizer from real implementation for high-speed systems. Parallelization is important as it allows better utilization of the hardware.

- Flexibility/Generalizability: The real-world optical networks are highly dynamic, in which the channel parameters fluctuate over time. Therefore, NN-based equalizers must be flexible enough to adapt to these changes without frequent retraining, due to the resource-constrained environment.

To allow the NN-based equalizers to be closer to the real implementations, the challenges need to be investigated and addressed. The proposed method should be able to outperform the traditional nonlinearity mitigation techniques, e.g. DBP, in terms of a trade-off between the computational complexity and the performance. In addition, the NN-based equalizers should also be studied in different types of transmission systems to validate their potential, for instance, in single-channel single-carrier systems, WDM systems and digital subcarrier multiplexing (DSCM) systems.

1.3 Contributions of the Thesis

This thesis focuses mainly on the possible solutions to alleviate the limitations of the NN-based equalizers and to move towards a real hardware implementation. The investigations are carried out in three different aspects: computational complexity, inference latency, and flexibility. The main contributions of the thesis are as follows:

- The low-complexity NN-based equalizer used in the DSCM systems via the weight clustering technique was proposed, showing the same level of performance with a significant reduction in computational complexity compared to the previous NN architecture based on perturbation theory [1].
- To reduce the complexity of the hardware implementation of NN-based equalizers, it is demonstrated that the biLSTM equalizer with approximated activation functions provides a performance close to that of the original model but significantly reduces the hardware requirements.
- To improve the latency and the non-parallelizability of biLSTM-based equalizers, *knowledge distillation* was proposed to recast the recurrent-based structure into a more parallelizable feedforward structure.
- Multi-task learning (MTL) is proposed to improve the flexibility of NN-based equalizers in coherent optical systems. A “single" NN-based equalizer improves the Q-factor, without retraining, even with variations in launch power, symbol rate, or transmission distance.
- To improve the flexibility of NN-based equalizers using MTL in coherent-detection WDM systems, a “single” NN-based equalizer can mitigate different levels of nonlinearity like cross-phase modulation (XPM) across diverse channel spacing,

outperforming conventional training and enhancing Q-factor without retraining the NN.

The following publications stem from this current research and are directly or indirectly connected to various chapters of this thesis:

Conference papers:

C1 **Srivallapanondh, S.**, Freire, P., Parisi, G., Devigili, M., Costa, N., Spinnler, B., Napoli A, Prilepsky, J.E. and Turitsyn, S.K., 2025, March. Weight-clustered neural networks for low-complexity nonlinear equalization in digital subcarrier multiplexing systems. In Optical Fiber Communication Conference. Optica Publishing Group.

C2 **Srivallapanondh, S.**, Freire, P., Napoli, A., Prilepsky, J. and Turitsyn, S., 2024, November. State-of-the-art neural network-based equalizers for coherent optical communication systems: architectures and complexity. In SBFoton IOPC 2024 (pp. 1-3). doi: 10.1109/SBFotonIOPC62248.2024.10813543.

C3 **Srivallapanondh, S.**, Freire, P.J., Alam, A., Costa, N., Spinnler, B., Napoli, A., Sedov, E., Turitsyn, S.K. and Prilepsky, J.E., 2023, October. Multi-task learning to enhance generalizability of neural network equalizers in coherent optical systems. In 49th European Conference on Optical Communications (ECOC 2023) (Vol. 2023, pp. 640-643). IET. doi: 10.1049/icp.2023.2276.

C4 **Srivallapanondh, S.**, Freire, P.J., Napoli, A., Turitsyn, S.K. and Prilepsky, J.E., 2023, May. Hardware realization of nonlinear activation functions for NN-based optical equalizers. In CLEO: Science and Innovations (pp. SF1F-4). Optica Publishing Group.

C5 **Srivallapanondh, S.**, Freire, P.J., Spinnler, B., Costa, N., Napoli, A., Turitsyn, S.K. and Prilepsky, J.E., 2023, March. Knowledge distillation applied to optical channel equalization: Solving the parallelization problem of recurrent connection. In Optical Fiber Communication Conference (pp. Th1F-7). Optica Publishing Group. doi: 10.1364/OFC.2023.Th1F.7.

C6 Freire, P.J., **Srivallapanondh, S.**, Spinnler, B., Napoli, A., Costa, N., Prilepsky, J.E. and Turitsyn, S.K., 2023, October. Low-complexity efficient neural network optical channel equalizers: Training, inference, and hardware synthesis. In 49th

European Conference on Optical Communications (ECOC 2023) (Vol. 2023, pp. 542-545). IET. doi: 10.1049/icp.2023.2240.

Journal papers:

J1 **Srivallapanondh, S.**, Freire, P., Parisi, G., Devigili, M., Costa, N., Spinnler, B., Napoli, A., Prilepsky, J. and Turitsyn, S., 2025. Low complexity neural network equalizer for nonlinearity mitigation in digital subcarrier multiplexing systems. Optics Express, 33(2), pp. 2558-2575, doi: 10.1364/OE.542061.

J2 **Srivallapanondh, S.**, Freire, P., Spinnler, B., Costa, N., Schairer, W., Napoli, A., Turitsyn, S.K. and Prilepsky, J.E., 2024. Experimental validation of XPM mitigation using a generalizable multi-task learning neural network. Optics Letters, 49(24), pp.6900-6903, doi: 10.1364/OL.535396.

J3 **Srivallapanondh, S.**, Freire, P.J., Spinnler, B., Costa, N., Napoli, A., Turitsyn, S.K. and Prilepsky, J.E., 2024. Parallelization of recurrent neural network-based equalizer for coherent optical systems via knowledge distillation. Journal of Lightwave Technology, 42(7), pp.2275-2284, doi: 10.1109/JLT.2023.3337604.

J4 Freire, P., **Srivallapanondh, S.**, Spinnler, B., Napoli, A., Costa, N., Prilepsky, J.E. and Turitsyn, S.K., 2024. Computational complexity optimization of neural network-based equalizers in digital signal processing: a comprehensive approach. Journal of Lightwave Technology, 42(12), pp. 4177-4201, doi: 10.1109/JLT.2024.3386886.

J5 Devigili, M., Sequeira, D., Torres-Ferrera, P., **Srivallapanondh, S.**, Costa, N., Ruiz, M., Castro, C., Napoli, A., Pedro, J. and Velasco, L., 2024. Twining digital subcarrier multiplexed optical signals with OCATA for lightpath provisioning. Journal of Lightwave Technology, 43(6), pp. 2599-2609, doi: 10.1109/JLT.2024.3498342.

J6 Freire, P.J., **Srivallapanondh, S.**, Anderson, M., Spinnler, B., Bex, T., Eriksson, T.A., Napoli, A., Schairer, W., Costa, N., Blott, M., Turitsyn, S.K. and Prilepsky, J.E., 2023. Implementing neural network-based equalizers in a coherent optical transmission system using field-programmable gate arrays. Journal of Lightwave Technology, 41(12), pp.3797-3815, doi: 10.1109/JLT.2023.3272011.

1.4 Thesis Outline

This thesis includes the technical introduction of the subject, the research findings, and the conclusion. The thesis is outlined as follows:

- Chapter 2 presents an introduction to the channel equalization in the optical communication systems. The traditional nonlinearity mitigation techniques for optical channel equalization and their challenges are discussed. After that, the fundamentals of the NN-based equalizers are presented. This chapter also provides an overview of the different traditional nonlinearity mitigation techniques, as well as NN topologies and their opportunities in the equalization tasks. The chapter also mentions the three main challenges of NN-based equalizers covered in this thesis: computational complexity, parallelizability, and generalizability. Finally, the complexity reduction methods for NNs and the complexity metrics for training and inference of NN are discussed.

- Chapter 3 examines the first challenge of the NN-based equalizer, which is the computational complexity. It investigates two different complexity reduction techniques: weight-clustering and the approximation of the nonlinear activation functions. The weight-clustering method was applied to reduce the number of real multiplications per equalized symbol in the NN-based equalizers used in DSCM. For the nonlinear activation function approximation, the study focuses on a trade-off between the performance and the hardware resources required for the nonlinear activation functions with and without approximation techniques; this study was examined with the data in a single-channel single-carrier system.

- Chapter 4 addresses the second aspect of the challenges of the NN-based equalizers covered in this thesis. It discusses the parallelizability problem of the recurrent-based NN equalizers. For the first time, a knowledge distillation framework was applied to allow the feed-forward NN to learn better and provide comparable performance in channel equalization as the recurrent NN-based equalizers. The feed-forward structure enables faster processing time than the recurrent one. This approach was studied using both simulated and experimental data.

- Chapter 5 investigates the last aspect of the challenges of NN-based equalizer covered in this thesis, which is generalizability. The MTL approach helps improve the flexibility of the NN equalizers. MTL allows the NN to still perform well when

the transmission setup changes, without retraining the model. First, the MTL was evaluated in the single-channel transmission where the launch power, number of spans, and the symbol rate were dynamic. After that, the MTL was assessed in the WDM systems, where the channel spacings can be dynamic.

- Chapter 6 concludes the thesis with some discussions and the future research direction.

The main parts of this thesis are based on my original research published in conferences C1 through C5 and journals J1 through J3, where I served as the primary author and made the predominant contributions, including developing the code and generating the results. Although some results and text are derived from journals J4 and J6—where I was a secondary author—I was the primary author for the portions included in this thesis. Finally, collaborative brainstorming with my co-authors was pivotal in shaping the ideas and revising the papers.

Chapter 2

NN-Based Equalizers in Optical Communications

2.1 Introduction

Research into nonlinearity mitigation techniques in optical communication systems remains an active area of investigation. Different techniques have been proposed and investigated. This chapter introduces various classical algorithms: DBP, the Volterra Series-based method, phase conjugation, and the perturbation theory-based method. After that, the NN-based equalizers are discussed along with their architecture based on data-driven and model-driven NNs, and previously proposed methods based on different transmission schemes. Then, it presents the introduction of computational complexity reduction techniques in the training, inference, and hardware synthesis phases. Finally, the metrics to measure the computational complexity in the training and inference phases are reviewed.

2.1.1 Traditional Equalizers for Nonlinearity Mitigation

Various techniques have been researched to alleviate the nonlinear effects of optical fibers. The DSP algorithms are used to compensate for the fiber impairments in coherent detection in optical communication. These DSP algorithms are deployed either on the transmitter or the receiver side, or a combination of both [26]. The digital nonlinearity compensation (NLC) techniques are presented as a key approach which is cost-effective to increase the data rate in the next-generation WDM optical

transmission systems [26]. The most common NLC techniques are presented here. These techniques aim to solve the Manakov equation.

Digital Backpropagation

DBP is a powerful signal processing technique designed to compensate for both chromatic dispersion and Kerr-induced nonlinear impairments. The idea of DBP is to digitally model a fictitious fiber with inverse parameters (opposite sign) when compared to the real fiber deployed in the forward propagation [26, 17], based on the SSFM. The SSFM is an effective numerical technique used to solve NLSE [14]. The SSFM divides the optical link into small segments. The signal propagation in those steps is modeled as a concatenation of linear and nonlinear operations which are treated separately. The DBP implemented on the receiver side can be seen in Fig. 2.1, where N_{span} is the number of steps. The conversion between time and frequency domains is undertaken by the FFT and inverse FFT (IFFT) [26, 13]. The linear compensation section is implemented in the frequency domain. With SSFM, we can derive the output of the linear compensation as stated in [26] as:

$$U^{\mathrm{CD}}_{X/Y}(z,\omega) = FFT(u_{X/Y}(z,t))e^{-jh\left(\frac{\alpha}{2}+\frac{\beta_2}{2}\omega^2\right)}, \tag{2.1}$$

where h is the length of each step, ω is the frequency variable, and z is the current transmission distance. The exponential term represents the inverse of the signal phase change due to the dispersion.

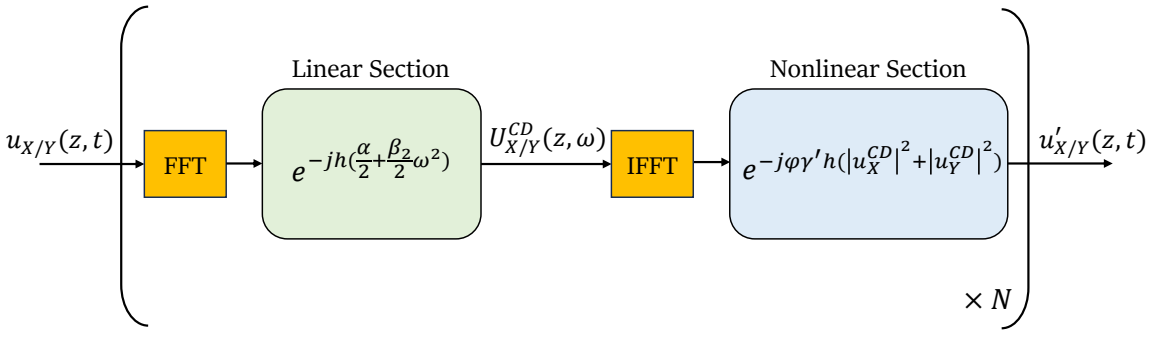


Fig. 2.1 Principle of DBP implementation, FFT: fast Fourier transform and IFFT: inverse FFT.

After that, the NLC is performed in the time domain to address the Kerr effects. The output of the DBP is formulated by [26] as:

$$u'_{X/Y}(z,t) = IFFT(U^{\mathrm{CD}}_{X/Y}(z,\omega))e^{-j\varphi\gamma' h\left(|u^{\mathrm{CD}}_X|^2+|u^{\mathrm{CD}}_Y|^2\right)}, \tag{2.2}$$

where $0 < \varphi < 1$ is a real-valued optimization parameter. The exponential term represents the phase change due to the Kerr effect. Apart from the phase shift caused by the self-modulation of polarization X/Y, the signal in polarization X induces a nonlinear phase variation in polarization Y, and vice versa.

The approximation of the solution for this SSFM can be improved by increasing the number of steps per span, resulting in better nonlinearity compensation. Nevertheless, a higher number of steps per span makes the computation expensive and infeasible in the implementation. Various research papers have made an effort to reduce the complexity of the algorithm, for example, reduced complexity DBP based on the joint usage of Wiener-Hammerstein model and a halved back-propagation [27], weighted DBP [28], correlated DBP [29] and the time-domain DBP with deep-learned chromatic dispersion filters [30]. The time-domain DBP is shown to be implementable in the Application-Specific Integrated Circuit (ASIC) [30].

Volterra Series-Based Nonlinear Equalizer

The Volterra series-based nonlinear equalizer (VNLE) uses the Volterra series transfer function (VSTF) to model the fiber nonlinear effects [31, 26]. The Volterra series is suitable for modeling the memory effects and is a powerful tool for solving the Manakov equation (Eq. (1.4)). After modeling the optical channel using VSTF, the inverse VSTF (IVSTF) kernels are derived as a function of the VSTF as in Ref. [32]. The IVSTF kernels characterize the nonlinear equalizer to compensate for the nonlinear effects and the dispersion of a transmitted signal. The VNLE also aims to construct the inverse of the channel like DBP. For each polarization, the compensation operation is divided into linear (like chromatic dispersion) and nonlinear parts. However, the NLC part of VNLE can perform the compensation operation in parallel, which can provide hardware benefits. The compensated output signal is a combination of the output of linear and each nonlinear stage; see Fig. 2.2. The output of VNLE can be formulated as a function

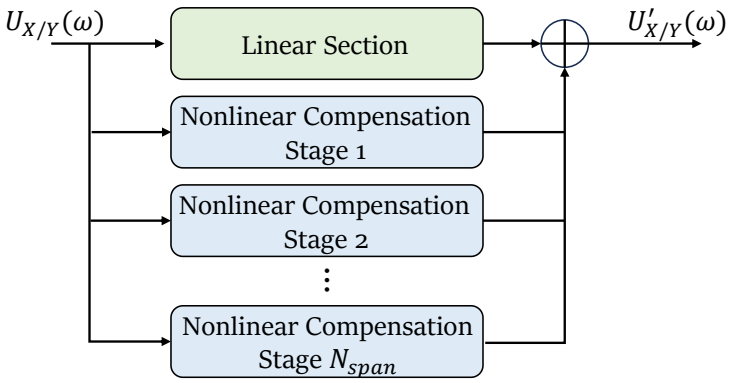


Fig. 2.2 Principle of VNLE implementation.

of the received signal as follows:

$$\begin{aligned} U'_{X/Y}(\omega) = k_1(\omega)U_{X/Y}(\omega) + \iint k_3(\omega_1,\omega_2,\omega-\omega_1+\omega_2) \\ \times [U_X(\omega_1)U_X^*(\omega_2) + U_Y(\omega_1)U_Y^*(\omega_2)] \\ \times U_{X/Y}(\omega-\omega_1+\omega_2)d\omega_1 d\omega_2, \end{aligned} \tag{2.3}$$

where k_1 and k_3 are the first- and third-order IVSTF kernels, ω is the physical optical frequency, ω_1 and ω_2 dummy parameters that influence the interactions between light waves at different frequencies, and the superscript * is the complex conjugation. The equations of k_1 and k_3 can be written as:

$$k_1(\omega) = e^{j\omega^2\beta_2 N_{span}L/2} \tag{2.4}$$

$$k_3(\omega_1,\omega_2,\omega-\omega_1+\omega_2) = \frac{jck_1(\omega)}{4\pi^2}\sum_{k=1}^{N} e^{jk\beta_2\Delta\omega L} \tag{2.5}$$

where L denotes the span length, $\Delta\omega = (\omega_1-\omega)(\omega_1-\omega_2)$ corresponds to the spacing between the discrete frequencies in the sampling spectrum and $c = \gamma' L_{eff}$, where L_{eff} is the effective length.

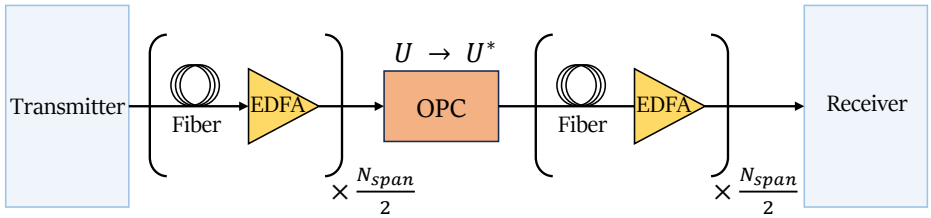
There are different forms of the VNLE attempting to improve the performance and reduce the complexity; for instance, weighted Volterra series nonlinear equalizer [33]. Most of the VNLEs are based on truncating the series to the third order. However, some papers propose the fifth-order equalizer [34, 35], offering performance improvement in the single-channel systems but the complexity is also increased compared to the

third order. It is worth noting that the nonlinear interference caused by the adjacent subcarriers in the super-channel transmission decreases the VNLE performance [26]. Also, the complexity of the Volterra series-based method increases with the number of spans [13].

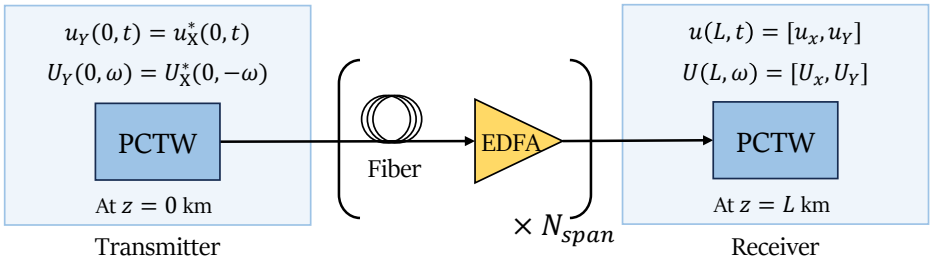
Phase Conjugation-Based NLC

Phase conjugation (PC) exploits the symmetry of nonlinear distortions to address the nonlinear phase shift induced by the Kerr effects. This PC approach has been proposed for both in the optical domain as optical phase conjugation (OPC) [36] and in the digital domain as digital phase conjugated twin waves (PCTW) [37].

OPC inverts the spectrum of the data signal in the optical domain midway through the transmission link. The fundamental concept of OPC is that the nonlinear phase shift accumulated in the first half of the fiber can be effectively canceled by the second half when the conjugate wave is transmitted. The implementation of OPC is presented in Fig. 2.3a However, this approach significantly limits the flexibility of the dynamic optical network and is difficult to implement. The key challenge of OPC is the requirement for a symmetric fiber link and precise positioning.



(a) OPC implementation.



(b) PCTW implementation.

Fig. 2.3 Implementations of phase conjugate-based NLC in (a) optical domain and (b) digital domain.

In the dual polarization system, PCTW is a DSP technique that was proposed to mitigate the first-order nonlinear distortions [37]. PCTW is implemented at the receiver side, as seen in Fig. 2.3b. With this approach, the original signal is transmitted on the X polarization and its conjugate is sent on the Y polarization:

$$u_Y(0,t) = u_X^*(0,t), \quad U_Y(0,\omega) = U_X^*(0,-\omega). \tag{2.6}$$

The nonlinear distortions faced by PCTWs are anticorrelated. The nonlinear distortions are:

$$\delta u_Y(L,t) = -[\delta u_X(L,t)]^*, \quad \delta U_Y(L,\omega) = -[\delta U_X(L,-\omega)]^*. \tag{2.7}$$

The received signal is approximated as:

$$u_{X/Y}(L,t) = u_{X/Y}(0,t) + \delta u_{X/Y}(L,t), \quad U_{X/Y}(L,\omega) = U_{X/Y}(0,\omega) + \delta U_{X/Y}(L,\omega). \tag{2.8}$$

The anticorrelation results in the first-order cancellation of nonlinear phase shift, by the superposition of the two signals at the receiver side. The original signal field ($u(0,t)$ or $U_X(0,\omega)$) can be restored by:

$$u_X^{'}(L,t) \approx u(0,t) = \frac{u_X(L,t) + u_Y^*(L,t)}{2}, \tag{2.9}$$

$$U_X^{'}(L,\omega) \approx U_X(0,\omega) = \frac{U_X(L,\omega) + U_Y^*(L,-\omega)}{2}. \tag{2.10}$$

The main advantage of this PC approach is its low complexity implementation. On the contrary, when adopting PCTW, the loss of half spectral efficiency is the main limitation because the conjugate signal requires transmission on the Y polarization. To address this, advanced coding schemes such as polarization coding [38] and subcarrier coding [39] have been proposed to improve efficiency.

Perturbation-Based NLC

The perturbation-based approach allows an approximate numerical solution to the Manakov equation, or Eq. (1.4). The core concept of this technique is to treat the field in the fiber as the combination of the linear propagation caused by dispersion and attenuation, and perturbed terms due to nonlinear distortions [26]. With the first-order perturbation, the received field $u_{X/Y}(z,t)$ is the sum of the solution to the

linear propagation $u_{0,X/Y}(z,t)$ and the first-order perturbation $u_{1,X/Y}(z,t)$. It can be written as: $u_{X/Y}(z,t) = u_{0,X/Y}(z,t) + u_{1,X/Y}(z,t)$.

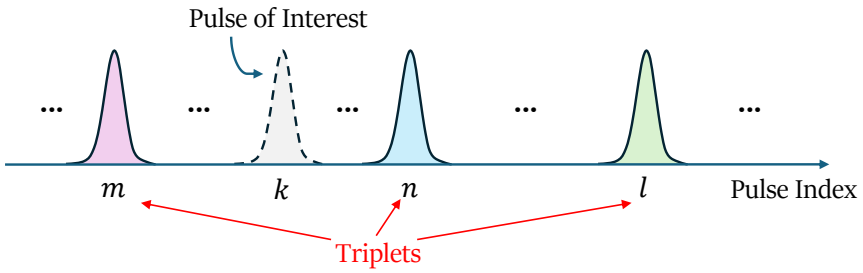


Fig. 2.4 Triplet pulses in perturbation-based NLC

Two simplifying assumptions hold in the first-order perturbation-based NLC, including: the full electrical chromatic dispersion compensation (CDC) at the receiver and the Gaussian shape assumption for input pulses [13]. Based on first-order perturbation theory, at the time indices T_m, T_l and T_n, three input Gaussian pulses interact nonlinearly and generate a ghost pulse. Fig. 2.4 provides a visual representation of the triplet pulses to generate the first-order field. For simplification, without loss of generality, the nonlinear distortion field is evaluated at index $k = 0$ (i.e. when $l = m + n$) by considering the symbol rate operation. The first-order distortion field is as follows:

$$u_{1,X/Y}(L,t) = j\frac{8}{9}\gamma P_0^{3/2} \sum_m \sum_n \left[a_{m,X/Y} a^*_{m+n,X/Y} a_{n,X/Y} + a_{m,Y/X} a^*_{m+n,Y/X} a_{n,X/Y} \right] \mathbf{C}_{m,n}, \tag{2.11}$$

where $C_{m,n}$ are the nonlinear perturbation coefficients, $*$ denotes the complex conjugate operation, P_0 is peak power of the pulse at the launch point and $a_{m,X/Y}$ represents the symbol complex amplitudes at time m for X and Y polarization. In a typical dispersion uncompensated system, the pulse spreading due to chromatic dispersion is much higher than the symbol duration, i.e., $\beta_2 z \gg \tau^2$ [40]. $C_{m,n}$ can be formulated as [41]:

$$\mathbf{C}_{m,n} = \begin{cases} \frac{\tau^2}{\sqrt{3}|\beta_2|} \int_0^L dz \frac{1}{\sqrt{\tau^4/(3\beta_2^2) + z^2}}, & m = n = 0 \\ \frac{\tau^2}{\sqrt{3}|\beta_2|} \frac{1}{2} E_1\left(\frac{(n-m)^2 \tau^2 \tau^2}{3|\beta_2|^2 L^2} \right), & m \text{ or } n = 0 \\ \frac{\tau^2}{\sqrt{3}|\beta_2|} E_1\left(-j\frac{mn\tau^2}{\beta_2 L} \right), & m \neq n \neq 0, \end{cases} \tag{2.12}$$

where $E_1(x) = \int_x^\infty \frac{e^{-t}}{t} dt$. These perturbation coefficients are computed in advance and stored in a look-up table. To compensate for nonlinearities, the first-order distortion

field $u_{1,x/y}$ was calculated and is subtracted from the symbol for interest $a_{0,X/Y}$ to generate $a'_{0,X/Y}$. This can be written as: $a'_{0,X/Y} = a_{0,X/Y} - u_{1,x/y}$.

The perturbed term can be calculated and utilized for the nonlinearity mitigation at either the transmitter or the receiver side. The perturbation-based NLC can also be implemented with one sample per symbol to reduce the speed requirement of the digital-to-analog converter (DAC), and the analog-to-digital converter (ADC) [40]. Tao et al. [40] demonstrated that the perturbation-based NLC can be implemented without any multipliers when adopting low spectral efficiency modulation formats, like QPSK. However, it did not apply to the higher-order modulation. Different works have attempted to reduce the number of perturbation terms, for example, quantization on perturbation coefficients [42].

Nonlinear Fourier Transform

An alternative approach to nonlinearity mitigation is the nonlinear Fourier transform (NFT). NFT is a framework that exploits the integrability of the NLSE (Eq. (1.1)), which describes pulse propagation in single-mode fibers [43–45].

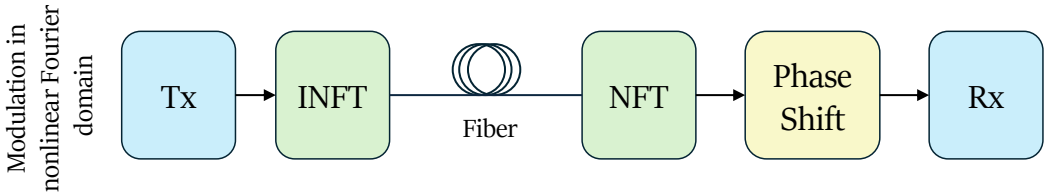


Fig. 2.5 Basic design of NFT-based transmission systems in the NFT domain.

Fig. 2.5 [44] shows the basic design of the NFT-based transmission systems, where the transmitted information is encoded by inverse NFT (INFT) directly onto the nonlinear Fourier (NF) signal spectrum (modulation in the nonlinear Fourier domain – NFD). In this design, one can modulate discrete and continuous NF spectrum parts either separately or simultaneously.

The forward NF decomposition can be performed by the solutions of the Zakharov–Shabat spectral problem (ZSSP) [46, 47]. For anomalous dispersion ($\beta_2 < 0$), the NLSE equation is integrable and can be mapped to the ZSSP. The ZSSP corresponds to the scattering problem for two auxiliary functions $\phi_{1,2}(t)$, where the transferred

temporal profile $u(0,t) \equiv u(t)$ enters as an effective potential [45]:

$$\frac{d}{dt}\begin{pmatrix}\phi_1(t,\xi)\\ \phi_2(t,\xi)\end{pmatrix} = \begin{pmatrix}-i\xi & u(t)\\ -u^*(t) & i\xi\end{pmatrix}\begin{pmatrix}\phi_1(t,\xi)\\ \phi_2(t,\xi)\end{pmatrix}, \tag{2.13}$$

where $\xi \in \mathbb{C}$ is the nonlinear spectral parameter.

The nonlinear spectrum (NS) fully characterizes the original time-domain waveform $u(t)$ and is obtained from the scattering coefficients $a(\xi)$ and $b(\xi)$. It consists of two distinct components:

1. **Continuous spectrum** — defined by the reflection coefficient

$$r(\xi) = \frac{b(\xi)}{a(\xi)}, \quad \xi \in \mathbb{R}. \tag{2.14}$$

2. **Discrete spectrum** — a set of solitonic eigenvalues $\xi_d \in \mathbb{C}^+$ satisfying $a(\xi_d) = 0$, together with their associated residues:

$$r(\xi_d) = \frac{b(\xi_d)}{a'(\xi_d)}. \tag{2.15}$$

For finite-duration pulses, these definitions are always valid in optical communication applications. Low-energy signals contain only the continuous spectrum, while the discrete part appears only when the signal energy exceeds a threshold [48]:

$$\int_{-\infty}^{\infty} |q(t)|^2\, dt > C. \tag{2.16}$$

In the case of normal chromatic dispersion ($\beta_2 > 0$), the discrete spectrum is absent altogether.

The significance of the NFT in fiber-optic communications lies in the simple, decoupled evolution of the nonlinear spectrum in the ideal, noise-free (without ASE noise or higher-order perturbations) NLSE model. For the normalized NLSE, the propagation over a distance L is described by [45]:

$$\mathrm{NS}(L) = \begin{cases} b(L,\xi) = b(0,\xi)\, e^{2i\xi^2 L}, \\ a(L,\xi) = a(0,\xi), \\ a'(L,\xi) = a'(0,\xi), \\ \xi_d(L) = \xi_d(0), \end{cases} \qquad \forall \xi \in \mathbb{R} \cup \Xi_d. \tag{2.17}$$

This invariance under propagation is what makes the nonlinear spectrum an attractive domain for information encoding and transmission.

In NFT-based transmission, data symbols are modulated directly onto the nonlinear spectrum—often onto the continuous part [49, 48]. The INFT generates the corresponding time-domain waveform for transmission, while at the receiver, the NFT recovers the transmitted nonlinear spectrum, which is largely free from deterministic nonlinear distortions. This principle underlies nonlinear inverse synthesis (NIS) [50] and nonlinear frequency-division multiplexing (NFDM) [44, 45].

Although NFT-based systems promise nonlinearity-immune transmission, practical implementation faces challenges. The challenges include the computational cost of accurate forward/inverse transforms and sensitivity to ASE noise in the nonlinear spectral domain. Ongoing research is addressing these issues through fast NFT algorithms [51], discrete-spectrum modulation strategies [49], and hybrid DSP–NFT architectures [44, 45].

2.1.2 Machine Learning-Based Equalizers for Nonlinearity Mitigation

Most traditional methods for nonlinearity compensation are still challenging to implement due to their high complexity, despite their effectiveness. This limitation has driven the research to seek alternative strategies. These new approaches should achieve comparable performance while significantly reducing computational complexity. ML techniques have been increasingly used in the past decade to design optical equalizers, as they have the ability to learn from the data and adapt to changing channel conditions. Different ML techniques are proposed to be used as optical equalizers, such as support vector machine (SVM), `K-means++` and NNs. The M-ary SVM in [52] was proposed for use in the 16-QAM coherent optical systems to mitigate the nonlinear phase noise (NLPN) which is one of the major distortion factors. The paper has shown that their approach, based on numerical simulations for 112-Gb/s single channel 16-QAM systems, can enhance the system performance independently of the specific characteristics of the fiber link.

The sparse K-means++ equalizer in [53] was proposed to mitigate optical fiber nonlinearity effects in a 16-QAM self-coherent real-time system at 40 Gb/s using an FPGA. The authors reported a 3 dB Q-factor improvement with respect to linear equalization only after transmission along 50 km of optical fiber using a launch power

close to the optimal value of 14 dBm and the tested scenario was a single-span short-reach system. The aforementioned techniques adapt their decision boundaries to the residual nonlinear distortion of the received signal instead of performing hard decisions, providing a significant performance gain. The improvement of the performance is noticeable when they are deployed in the memoryless systems or coping with NLPN [54].

Among the ML-based approaches, NN-based equalizers have garnered the most attention for their ability to model and mitigate complex nonlinear impairments, offering significant improvements over traditional nonlinearity compensation methods [55, 56]. For this reason, this thesis focuses on NN-based equalizers and the reduction of their complexity.

2.2 NN-Based Equalizers for Nonlinearity Mitigation

One significant advantage of NNs is their capacity to handle the vast datasets generated by optical transmission, enabling effective model training. With their universal approximation capability, NNs have been extensively explored for optical post-equalization, offering promising results compared to traditional techniques like DBP, particularly in terms of reduced complexity [16]. A straightforward application of NN-based equalizers is to be used as post-equalizers, where the NN is applied at the receiver side to reverse the channel distortions and recover the transmitted signal with high accuracy. This approach, which positions the NN structure only after the fiber channel at the receiver, provides an intuitive yet powerful method for addressing signal impairments in optical fiber systems. Fig. 2.6 illustrates the position of the post-equalizer in the optical communication systems. Another approach is end-to-end learning [57], which uses an autoencoder to optimize the entire communication system, including the transmitter, channel, and receiver. This end-to-end learning method, in [58] based on a parallelizable perturbative channel model, jointly optimized constellation shaping and nonlinear pre-emphasis, demonstrating mutual information gain. In this thesis, we focus on the post-equalization. By leveraging architectures such as convolutional NN (CNN) and RNN, and particularly a biLSTM, NN-based equalizers have demonstrated robust performance in mitigating nonlinear impairments [59, 16]. In this section, we first discuss some fundamental architectures of the NNs, both the data-driven and model-driven approaches for nonlinearity mitigation. Then the considerations when designing the NN-based equalizers are presented.

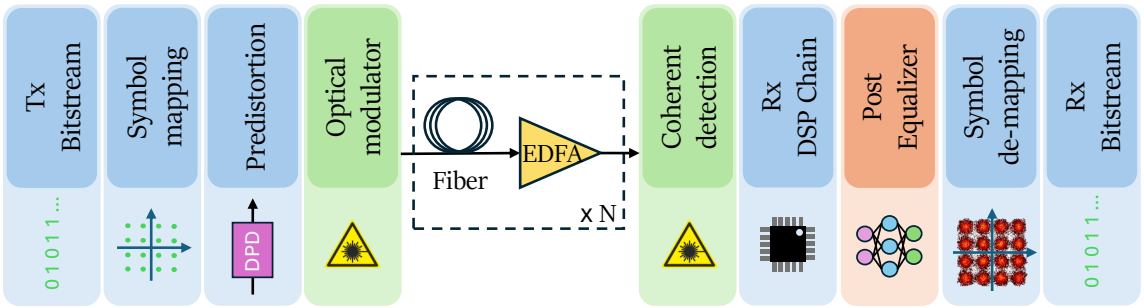


Fig. 2.6 Diagram depicting the position of NN as post equalizers in optical communications systems.

2.2.1 Architectures for Data-Driven NN-Based Equalizers

Multi-Layer Perceptron

Multi-layer perceptron or MLP is the simplest and feed-forward NN-based equalizer. The MLP consists of some layers of a dense layer, which is formulated as:

$$y = \phi(Wx + b), \tag{2.18}$$

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:

$$Wx + b = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n_i} \\ w_{21} & w_{22} & \dots & w_{2n_i} \\ \vdots & \vdots & \dots & \vdots \\ w_{n_n1} & w_{n_n2} & \dots & w_{n_nn_i} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_i} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n_n} \end{bmatrix}, \tag{2.19}$$

where n_i is the number of features in the input vector and n_n represents the number of neurons in the layer.

The MLP-based architectures have been well studied in the optical transmission systems [54, 60, 61]. In [54], the channel response in long-haul transmission systems is approximated by using the MLP networks. The paper demonstrates that their MLP-based equalizer provides the equivalent mitigation performance to the traditional DBP nonlinearity compensation of 2 STpS and 2 samples per symbol, but with a significantly lower computational cost. As the MLP model is unsuitable for learning sequential data, the extra step of data processing is required. In the paper, the delay blocks were used at the input layer of the MLP to consider the channel memory effect, meaning

that the preceding symbols are also considered when equalizing each symbol. The NN topology consists of two 16-neuron hidden layers and their neurons have hyperbolic tangent sigmoid transfer functions. The two output-layer neurons have linear transfer functions, providing real and imaginary parts of the equalized symbol. Note that this MLP equalizer is used after a linear equalization stage, enabling ideal CDC.

Three-layer MLP was demonstrated in [62] that when the complexity was restricted to a lower level, the MLP showed the best performance compared to other more complex models in terms of Q-factor. However, this type of NN used in the optical communication systems with pseudorandom bit sequences or with limited memory depths has the risk of overestimating the performance gain by predicting the short pattern instead of compensating the studied channel/phenomena [63]. MLPs aim to repeatedly discover the correlation among each pair of the data samples in each layer using fully-connected layers. As a result, they are prone to over-fitting due to a large number of trainable parameters and cause a large number of floating-point operations (FLOPs) [64].

Convolutional Neural Network

CNNs are a feed-forward NN that has the capability to extract patterns from data. This capability can be useful in learning and compensating nonlinearity, especially the stack of convolutional layers, which acts as a multi-channel nonlinear learned local pattern detector. It can overcome inter-symbol interference (ISI) and device nonlinearity [65]. In CNN, we apply the convolutions 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. In this thesis, we focus on 1D-CNNs, which apply to processing sequential data [18].

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:

$$y_i^f = \phi\left(\sum_{n=1}^{n_i}\sum_{j=1}^{n_k} x_{i+j-1,n}^{in} \cdot k_{j,n}^f + b^f\right), \tag{2.20}$$

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:

$$\text{OutputSize} = \left[\frac{n_s + 2\,padding - dilation(n_k - 1) - 1}{stride} + 1 \right], \tag{2.21}$$

where n_s is the input time sequence size.

The 1D-CNN has been proposed to be used with the MLP [66, 65, 67], using the CNN part to capture short-temporal dependencies among neighboring symbols and the MLP part for capturing long-term dependencies [64]. However, the MLP part is still inefficient due to a large number of FLOPs [64]. For the CNN without the MLP to capture long-term dependencies, it requires large sequential kernels, causing high computational complexity [64]. The 1D-CNN for the equalization of nonlinear impairments [66, 65, 67] were investigated in the short-reach transmission. Later on, the CNN for nonlinear mitigation in coherent systems has been studied to be used with other types of NNs, for example, 1D-CNN together with biLSTM network [62].

Vanila Recurrent Neural Network

RNNs are useful in learning sequential data due to their ability to handle memory, which is different from the feed-forward NNs. This ability of RNN can be quite beneficial for analyzing time series data. In RNN, the output of the current stage h_t takes into account the current stage input x_t and the output of the previous stage h_{t-1}, in which the equation of the RNN for a given time step t is as follows:

$$h_t = \phi(W x_t + U h_{t-1} + b), \tag{2.22}$$

where ϕ is 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, n_h is the number of hidden units, $W \in \mathbb{R}^{n_h \times n_i}$ and $U \in \mathbb{R}^{n_h \times n_h}$ represent the trainable weight matrices, and b is the bias vector.

Despite RNN's efficient memory handling, they still struggle to capture long-term dependencies due to the vanishing gradient problem [68]. The RNN-based equalizers have been researched in the nonlinearity mitigation subject, especially the bidirectional RNN (bi-RNN) [69, 64, 70].

Long Short-Term Memory Neural Networks

LSTM networks are a specialized form of RNNs. LSTM was designed to overcome short-term memory issues of RNNs due to the vanishing gradient problem. The LSTM network has the ability to learn long-term dependencies between time steps (t) [71, 72]. An LSTM cell consists of three types of gates: an input gate (i_t), a forget gate (f_t), and an output gate (o_t). There is also a cell state vector (C_t) proposed as a long-term memory to aggregate relevant information throughout the time steps. The forward pass of the LSTM cell given a time step t can be formulated as follows:

$$\begin{aligned} i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\ f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\ C_t &= f_t \odot C_{t-1} + i_t \odot \phi(W_c x_t + U_c h_{t-1} + b_c), \\ h_t &= o_t \odot \phi(C_t), \end{aligned} \tag{2.23}$$

where W_i, U_i, W_f, U_f, W_o, U_o, W_c and U_c are weight matrices associated with each gate (input gate, forget gate, cell state vector, and output gate, respectively), b_i, b_f, b_o and b_c are biases for each gate, ϕ is usually the "tanh" activation function, 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.

Because of the ability to learn long-term dependencies and keep track of information over large sequences, the biLSTM-based equalizers have gained research attention [73, 62]. In [73], the LSTM architecture was used for the first time to mitigate the fiber nonlinearity impairments in digital coherent systems for single-channel and multi-channel 16-QAM modulation format. The LSTM demonstrated superior performance over the DBP, especially in the multi-channel transmission scenario, while being able to retain the complexity to be less than that of the DBP in long distances (>1000 km). However, it is worth noting that the complexity of the LSTM grows as the number of hidden units and the channel memory increase. Lastly, the key benefit of biLSTM is to handle the ISI between the preceding and successive symbols induced by CD.

2.2.2 Physics-Inspired/Model-Driven NN-Based Equalizers

Unlike purely data-driven methods, which rely solely on large datasets for training, model-driven approaches integrate physical models, like the NLSE equation, into the network architecture. With the integration of knowledge of a channel model, the model has better interpretability.

Learned Digital Backpropagation

Learned DBP refers to the structure of a deep NN that is built upon the SSFM [74]. The analogy shows the similarity between the SSFM and the deep NN where each layer alternates between linear operations and nonlinearities. In learned DBP, the parameters in the deep NN are trainable, making it more flexible than the conventional DBP. Ref. [74] demonstrated the significant complexity reduction of the learned DBP compared to the conventional DBP, showing that the complexity can be reduced through pruning while still maintaining high performance. Ref. [75] further enhanced learned DBP for the WDM systems by addressing the nonlinearities from self-phase modulation (SPM) and XPM using an improved SSFM.

While effective, the learned DBP still depends on the sequential operations of the linear and nonlinear steps. Therefore, these sequential dependencies can make the learned DBP less parallelizable and result in the processing latency as hardware parallelism limitation [76].

Perturbation Theory-Based Models

Perturbation theory-based NN takes the intra-channel XPM (IXPM) and intra-channel four-wave mixing (IFWM) symbol triplets as input [77, 78], based on Eq. (2.11). This approach allows the perturbation parameters to be trainable. Different approaches have been proposed: to estimate perturbation coefficients [78] or directly predict nonlinear distortions from received symbols [77, 79]. Ref. [1] has proposed NN architecture based on perturbation analysis of fiber nonlinearity for DSCM systems. The approach based on perturbation theory offers an enhancement in the equalization tasks, however, the computational complexities still need to be addressed.

Learned Volterra Models

Learned Volterra algorithm builds upon the VSTF [76] and leverages machine learning to optimize nonlinear filters for mitigating nonlinearities. Volterra series-based models generally offer more parallelizable capabilities compared to SSFM-based methods like DBP. Ref. [76] proposed to use the learned inverse VSTF (L-IVSTF) models in a multiple-input multiple-output (MIMO) configuration for nonlinear equalization in WDM systems. This approach incorporates trainable finite impulse response (FIR) filters at the input and output of each nonlinear step and uses the hybrid structure between time and frequency domains. Despite the improved equalization performance of this learned Volterra algorithm, the computational complexity is still the main limitation.

It is worth noting that both the data- and model-driven approaches have their own advantages and disadvantages. The data-driven approach can approximate highly complex and unknown channel models without relying on explicit mathematical formulations and the models can adapt to various channel conditions without prior knowledge of the system's parameters. However, this results in higher training costs and resources, and the model is not interpretable. The model-driven NN improves the interpretability of the model and reduces the dependence on the large training data. On the other hand, the quality of the model-driven model is directly tied to how accurately the physical model represents the real-world system and also may not perform well when the channel contains unknown impairments. This thesis focuses on the data-driven approach.

2.2.3 Designing NN-Based Equalizers

Despite the NN's potential, the computational complexity of NN-based equalizers remains a challenge for real-world implementation, necessitating further research into optimization techniques.

Regarding the NN architecture, various models, including feed-forward NN (FNN), RNNs, and Transformers, have demonstrated their effectiveness in NLC. The FNN has the risk of overestimating the performance gain by predicting the short pattern instead of compensating the studied channel/phenomena [63]. RNNs have shown better equalization performance compared to FNNs when addressing nonlinear impairments [25]. Among these, RNNs, particularly in their bi-directional and gated configurations e.g. LSTM, have shown great promise for addressing nonlinear optical fiber effects in bandwidth-limited optical coherent systems [59]. Recurrent-based models are capable

of modeling and compensating for the distortions of nonlinear channels with memory as they account for the output of the previous time step.

Freire et al. [80] compared classification and regression modeling to determine the best approach for NN-based equalizers. One challenge with classification is that the datasets often contain few errors when calculating loss, making it difficult to train the model effectively and causing issues like exploding or vanishing gradients. In contrast, regression with mutual information early stopping allows every data point to contribute, providing continuous feedback and improving model performance. These factors are crucial when designing NN-based post-equalization techniques for transmission systems.

Another aspect to consider is if the NN would deploy the real or complex values. With complex values, it is more challenging with the training, activation functions, and more complex arithmetic. However, it has been shown that the optical NLC can benefit from complex-valued NN [56].

Lastly, the output structure of an NN equalizer earlier focused on single-symbol recovery, where the NN predicted the real and imaginary parts of each symbol. However, multi-symbol equalization is currently used in recent studies [81, 55]. This approach reduces overall complexity by cutting down the number of sliding windows in equalization, as one sliding window can compensate for many more symbols at a time. This method improves learning efficiency and system performance. Multi-symbol output not only reduces the computational complexity per symbol but also reveals a superior performance compared to its single-symbol counterpart [82, 81].

2.3 Previously Proposed NN-Based Equalizers in Different Transmission Schemes

2.3.1 Single-Channel Transmission

Many studies have focused on NN-based equalizers in single-channel transmission because they simplify the problem, allowing the isolation of key impairments like chromatic dispersion and fiber nonlinearities without the added complexity of inter-channel interference found in multi-channel systems. Additionally, single-channel scenarios provide a clear benchmark to prove the concept of NN-based equalization before extending it to more complex systems like WDM.

Liu et al. [83] leveraged the Attention mechanism in the bidirectional RNN (biRNN)-based equalizer. The Attention mechanism is a technique used in NNs, especially in

sequence processing tasks, that allows the model to focus on the most relevant parts of the input when making a prediction. This paper used this Attention mechanism to explore the contribution of each input symbol in the input sequence as well as their hidden representations for predicting the received symbols. The low-complexity partial-biRNN with the gated recurrent unit was proposed, showing a 56.16% reduction in real multiplications per symbol (RMpS) in comparison to the full biLSTM model.

Huang et al. [84] introduced a perturbation theory-aided complex-valued fully connected NN (P-CFNN) model for NLC, enhanced by a complex principal component analysis technique. The proposed P-CFNN model demonstrated superior performance, while significantly reducing computational complexity. Specifically, it delivers a 40% reduction in time complexity and a 70% reduction in space complexity compared to real-valued NN with equivalent complexity. In this work, space complexity refers to how much memory an algorithm needs to run, based on the size of the input data. It is typically measured by the total number of parameters in the model.

Xiang et al. [85] proposed a low-complexity nonlinear equalizer based on a conditional generative adversarial network (c-GAN) for coherent data-center interconnections. The proposed c-GAN equalizer presented superior performance compared to traditional approaches like Volterra filter equalizers and other models such as MLP and LSTM. The c-GAN reduced the complexity by up to 98.8% compared to LSTM, while significantly improving performance. The reduction in complexity resulted from lightweight c-GAN architecture that uses only the generator during inference and optimizes input representation via a sliding window and normalization.

Freire et al. [25] demonstrated biLSTM, CNN equalizer, and CDC block implementation on the FPGA. The authors assessed the complexity reduction of the NN due to the implementation using fixed-point arithmetic and nonlinear activation function approximations.

Gautam et al. [86] investigated a Transformer-based nonlinear equalizer, showing superior performance compared to DBP, fully connected NNs (FCNN), and biLSTM. This Transformer-based model leverages its self-attention mechanism and an optimized encoder-only architecture with a linear feature extractor. Even though the Transformer required significantly lower complexity in RMpS than DBP, it is more computationally expensive than the FCNN and biLSTM.

Jiang et al. [87] developed a wide and deep-based equalizer. The authors explored both wide and deep CNN-based and wide and deep bidirectional gated recurrent unit (biGRU)-based nonlinear equalizers. It was reported that a wide network can capture

the power feature factor of a single symbol better, whereas a deep network handles the feature sequences that contain nonlinear interference relationships between symbols. The results showed that this proposed approach improved the optical performance at a cost of less than 0.1% increase in complexity compared to the normal architecture.

2.3.2 Wavelength-Division Multiplexing

To increase the capacity, it is necessary to adopt ultra-high bandwidth channels in WDM, which are characterized by a dense arrangement. This approach allows for the transmission of more data simultaneously by utilizing a greater number of closely spaced wavelengths within the available spectrum. The difficulties of the nonlinearity compensation in the WDM environment are the necessity to have more taps and more memory in the equalizers to take into account the inter-channel nonlinearities. Many papers have investigated the NN-based equalizer in WDM systems, however, more investigations on the complexity and the requirement of the neighboring channel's information for the NLC are necessary.

Sidelnikov et al. [88] presented a deep CNN architecture for long-haul WDM systems. The proposed model simulated the effects of DBP with optimized convolutional and activation layers, effectively suppressing a significant portion of the XPM-induced signal distortions while maintaining low computational complexity. The results demonstrate that the proposed model outperforms traditional linear and DBP methods in both single and multi-channel compensation scenarios.

Freire et al [89] introduced the CNN+biLSTM equalizer that outperformed DBP with 3 STpS. This proposed equalizer only utilized the information of the channel under test without additional information about neighboring channels (only information from the signals leaked from the adjacent channels) as input. It was also reported that the NN topology based on biLSTM layers was able to partially recover the XPM since the higher gain in Q-factor was obtained compared to the single channel scenario.

Deligiannidis et al. [59] proposed a multichannel equalization approach using biRNN for coherent WDM optical systems, through the simulated and experimental data. The proposed method effectively mitigates nonlinear impairments such as XPM, offering significant improvements in optical performance while drastically reducing computational complexity compared to the full-field DBP.

2.3.3 Digital Subcarrier Multiplexing

DSCM has recently gained attention as an effective solution to address the rapid growth of internet traffic. In DSCM, subcarriers are generated and managed in the digital domain using DSP. By optimizing symbol rates, DSCM offers greater resilience against nonlinear distortion compared to traditional single-carrier systems [90]. This happens because splitting a high-baud-rate signal into multiple low-baud-rate subcarriers improves nonlinearity tolerance. This improved tolerance is attributed to the theory of four-wave mixing efficiency or the walk-off between subcarriers due to chromatic dispersion [90]. This makes DSCM a promising approach for the evolution of optical networks, enabling the transition from point-to-point to point-to-multipoint architectures. Its flexibility leads to significant reductions in both capital and operational expenditures in optical networks, making it an attractive choice for future deployments.

Saif et al. [91] presented deep learning-assisted NLC, based on the combination of the subcarrier multiplexing (SCM)-DBP and learned DBP. The SCM-LDBP method extends SCM-DBP by incorporating MLP to perform the compensation. Instead of frequency-domain filters used in SCM-DBP for CDC and cross-subcarrier nonlinearity (CSN) mitigation, SCM-LDBP employs time-domain filters implemented as MLP layers. The results showed around 38% complexity reduction compared to the SCM-DBP [92].

Bakhshali et al. [1] proposed different NN architectures, including the black-box approach and the model based on perturbation analysis to compensate for optical channel nonlinearities in DSCM. The authors showed the performance of NN with different complexity limits and demonstrated that the model based on perturbation theory outperformed other models given the same complexity budget.

There are still a limited number of papers exploring the NN-based equalizer in DSCM. However, some literature [93–95] investigated different NN architectures in orthogonal frequency division multiplexing (OFDM). OFDM divides the available bandwidth into multiple orthogonal subcarriers[1], allowing data to be transmitted in parallel streams. This approach also enhances spectral efficiency and provides robustness against channel impairments like chromatic dispersion and polarization mode dispersion.

[1] In the DSCM, the subcarriers are not orthogonal.

2.4 Complexity Reduction Techniques of NN-Based Equalizers

As computational complexity is still the main challenge of the NN-based equalizers, different complexity reduction techniques have been explored. These complexity reduction techniques are categorized into three phases of implementation: training, inference, and hardware synthesis phase [55]. Fig. 2.7 shows the overview of the most common approaches that can be used to reduce the complexity of the NN-based equalizers in each phase. This thesis later shows the use case of some of the techniques mentioned here and explains them in detail in the following chapters.

2.4.1 Complexity Reduction Techniques in Training Phase

Minimizing complexity in the training phase is crucial for efficiency, scalability, and real-world deployment. As in the real-world situation, the training resources might be limited, especially in resource-constrained optical systems. Various techniques can also enhance the performance of the model and efficiency in the training, including transfer learning and methods that improve generalization. Examples of these methods are data augmentation, domain randomization, and semi-supervised learning. These strategies

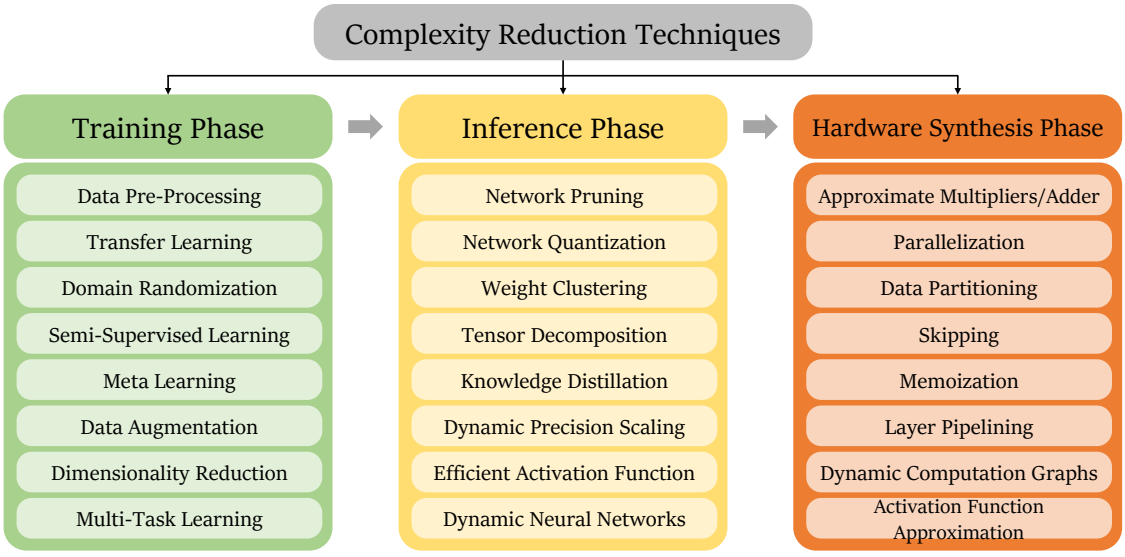


Fig. 2.7 Complexity reduction techniques for NN-based equalizers in training, inference, and hardware synthesis phases.

help lower dependency on large training datasets. Improving generalization decreases the need for complex models, resulting in faster and more efficient training.

Data pre-processing is an essential step to prepare the input for the NN training. High-quality input data can be beneficial in enhancing the model's efficiency and accelerating convergence. This is because the quality of input data contributes to the stability of the training, improves generalizability, and leads to successful data interpretation by the model [96]. Data pre-processing includes data normalization and feature engineering. Data normalization ensures features are on a consistent scale. Feature engineering involves selecting and transforming input features to highlight relevant information while eliminating noise and irrelevant data. In the context of this thesis, feature engineering specifically evaluates the relevance of given features to improve the Q-factor or Bit Error Rate (BER) performance. For example, while the real and imaginary parts of received symbols are intuitive features, it is crucial to assess whether transmission parameters (e.g., number of spans, transmission power) provide a meaningful contribution to the learning process or merely increase model complexity without proportional benefit.

Transfer learning (TL) applies the knowledge acquired in the source tasks to the related target tasks. The training time and resources required can be reduced significantly with TL. This approach is especially practical when training or fine-tuning the models on resource-constrained hardware. TL allows the NN to derive knowledge from a more sophisticated pre-trained model and fine-tune it for a more specific task. TL was studied in equalization tasks in both direct detection [97] and coherent [98] optical systems. Ref. [98] exhibited the potential of TL to reduce the training time and the size of the training dataset while maintaining the equalizer's performance. The work carried out in this thesis also leverages TL to reduce the computational complexity in the training.

Domain randomization is used for data generation to improve the flexibility and the robustness of NN models when applied in new environments [99]. Domain randomization generates training data from a random distribution with given desired properties and stores it in a library accessible by the NN. By using synthetic data, this method reduces the dependence on real-world data, thus improving the training efficiency [100]. This technique is especially valuable when dealing with complex and dynamic environments, as the model becomes more adaptable to variations encountered during deployment.

Semi-supervised learning combines labeled and unlabeled data in training, allowing the model to learn from both. Semi-supervised learning enables NN to leverage the

available labeled data more effectively by incorporating information from the unlabeled samples. This method enhances the model's performance without requiring additional labeled data, which makes the model more flexible to transmission changes. This method resembles decision-directed adaptive equalization [101] for channel equalization.

Meta learning involves training models that can efficiently adapt to new tasks with only a small amount of training data based on experiences gained from a variety of learning tasks [102]. This approach explicitly trains the model parameters to enable efficient generalization on new tasks with a small number of gradient steps and minimal training data, making it simple to fine-tune.

Data augmentation allows datasets to be more diverse and representative by artificially generating additional data points from existing data [103]. Data augmentation used in optical NN-based equalizers is a technique to improve equalization performance and decrease the training complexity of supervised learning in nonlinearity mitigation. In supervised learning tasks, normally a large training dataset is required. The model will also need to be re-trained when the channel conditions change. However, big data collection can be challenging. The efficient use of a limited dataset is more desirable for practical implementation. Ref. [104] showed that data augmentation reduces the size of the dataset up to 6 times while maintaining the optical performance. This technique enables a less overfitting model, fewer model parameter requirements, and a faster convergence of the training.

Dimensionality reduction techniques address the challenges associated with high-dimensional input spaces. These techniques aim to capture and retain the most informative aspects of the data while reducing the number of input features significantly [105]. Principal Component Analysis (PCA) [106], for instance, transforms the original features into a lower-dimensional space defined by principal components (a new set of uncorrelated variables), retaining the maximum variance.

Multi-task learning (MTL) is a framework in which a single model is trained to perform multiple but related tasks simultaneously. In contrast, traditional single-task learning trains multiple separate models for each task independently. Multi-task learning leverages shared representations across tasks and the model's parameters are optimized jointly across all tasks. This training approach can lead to better generalization of the model and reduce the need to deploy several models for different tasks, and does not require re-training when performing related tasks. This technique not only reduces the number of models that need to be trained but also reduces the complexity in the inference phase. However, this technique can exhibit a trade-off between overall

performance and specific task performance. This approach has been shown to be efficient in the NN-based equalizers in both IM/DD [107] and coherent systems [108]. MTL is explained further with the case study in Chapter 5.

2.4.2 Complexity Reduction Techniques in Inference Phase

Next, during inference, the NN must accurately equalize the input signal using the minimum computational resources while meeting the required performance metrics. As in the real world, the environments and the resources are more constrained. This result can be achieved by using different techniques, for example, network pruning, sparse representation, knowledge distillation (KD), and tensor decomposition.

Network pruning or sparsification is the method to produce sparse NNs, aiming to reduce the computational complexity of the NN. This technique removes parameters, neurons, or even layers or parts of the NN that do not significantly impact its performance [109]. Pruning is known to be robust to various settings, able to achieve good performance, and able to support both trained from scratch and pre-trained models. 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 [16]. The information on each type of pruning is detailed in [110, 111]. The static, iterative, unstructured, global magnitude-based pruning is one of the simplest pruning approaches. The lowest magnitude weights are globally pruned throughout the NN. The weights are removed offline from the NN after training but before inference [16]. This iterative approach enables the NN to remove more weights while preserving accuracy.

Network quantization is an approach to decrease the bitwidth of the numbers in arithmetic operations. When it is deployed in signal processing, the complexity reduction of the processing is significant. For instance, the floating-point numbers are quantized to be in integer forms. This approach enables the NN to be represented using less memory and allows high-performance vectorized operations on various hardware platforms [16, 112]. This quantization technique has shown promising results in different NNs during both the training and inference process [112, 111]. This technique is even more effective when being deployed in the inference phase since the computing resources are saved without remarkable accuracy loss [16]. Particularly, the NNs can benefit from quantization, as the NNs are exceptionally robust to aggressive quantization due to a large number of parameters involved in the NN (over-parameterized models) [16].

Weight clustering also known as weight-sharing, is a compression technique to reduce the complexity of the NN. Weight clustering reduces the number of effective weights used by the NN, considering that several connections may share the same value of weights. Then it fine-tunes those shared weights [16]. Various papers have addressed the complexity issues in the feed-forward NNs by this approach [113–115]. This approach was explained in detail and demonstrated with the use case in Chapter 3.

Tensor decomposition decomposes high-dimensional data into a lower-dimensional space [116]. In other words, a multidimensional tensor is broken down into a combination of simpler tensors. By decomposing tensors, especially weight tensors in NN, into smaller and more manageable components, tensor decomposition reduces the number of parameters and computations needed in the inference phase. In [117], the authors showed that the sparse decomposition of the tensor in convolutional filters can successfully reduce model complexity and memory usage during inference.

Knowledge distillation (KD) is applied to transfer knowledge from a larger model (teacher) to a more compact one (student) using teacher predictions to assist student learning. KD can reduce the size of the model [118]. The distilled model retains the essential information from the teacher model, making it suitable for deployment in resource-constrained environments during the inference process. KD is discussed and examined with the real optical data in Chapter 4, to allow parallelization.

Dynamic precision scaling (DPS) dynamically adjusts the precision of the numerical values of the weights and activations during computation, based on the specific requirements of each computation. With this approach, the NN can utilize lower precision when the accuracy demands allow, as DPS optimizes the utilization of available resources. This approach provides an effective reduction of complexity during inference. Ref. [119] showed that DPS could be used in both the forward pass (inference) and backward pass for training.

Efficient activation function selection can play an important role in complexity reduction. The expensive activation functions, e.g., hyperbolic tangent or sigmoid, can be replaced by the approximated alternatives or with the look-up table to reduce the computation [25]. Simpler functions such as Rectified Linear Unit (ReLU) are also commonly chosen because of their simplicity in calculation and speed.

Dynamic neural networks [120] can adapt their structures or parameters and dynamically allocate computational resources based on different inputs. Instead of executing a fixed number of operations for all inputs, dynamic NN skips unnecessary computations and leads to reducing overall complexity. For instance, early exiting

processes the simpler samples with fewer layers or a more shallow network, without executing deeper layers. Unlike pruning and quantization that permanently reduce the size or precision of model parameters, dynamic NN offers a more dynamic and flexible solution.

2.4.3 Complexity Reduction Techniques in Hardware Synthesis Phase

Complexity reduction techniques of NN in hardware synthesis play a crucial role in optimizing the NN implementation on dedicated hardware. In hardware synthesis, the NN is mapped onto the hardware architecture, and the hardware design is optimized to achieve the desired performance while minimizing resource utilization. There are some techniques to reduce the complexity of the hardware, such as multiplier/adder approximations, parallelization, memoization, and skipping. The choice of suitable techniques depends on the NN architecture, the characteristics of the target hardware, and the desired trade-off between computational efficiency and model accuracy.

Approximate Multipliers/adders are to reduce the hardware resource requirements by approximating multiplier and adder, which are key components of the hardware implementation for NN computation [121]. The approximation replaces full-precision multipliers and adders with less resource-intensive multipliers and adder implementations, such as approximate adders or low-precision. Binary or ternary multipliers are examples of low-precision alternatives. By using lower-precision multipliers and adder implementations, the overall hardware complexity is reduced. This can lead to more efficient use of hardware resources without significantly sacrificing model accuracy.

Parallelization involves dividing the NN into multiple sub-networks to be processed simultaneously, aiming for faster and more efficient execution in terms of latency and throughput [122]. This methodology capitalizes on parallel hardware architectures, such as GPUs, yielding higher computational efficiency. As detailed in Ref. [123], parallelization can take various forms including: **Data Parallelism**, where multiple NN instances operate simultaneously on distinct data batches; **Model Parallelism**, involving the division of a single NN across multiple processors or GPUs, with different components processed on separate devices[2]; and **Pipeline Parallelism (inter-layer**

[2]In addition to the aforementioned parallelization techniques, another subcategory worth mentioning is intra-layer parallelism, often referred to as Tensor Parallelism. This method entails parallelizing computations within a single layer of the NN. Specifically, it involves partitioning large tensors, such as

parallelism), which segments the NN computation into stages, each executed by a distinct processing unit such that the data flow resembles a sequential assembly line.

Data partitioning is an approach to divide the input data into subsets to be processed in parallel by different hardware components independently [124]. After that, the result of each partition is combined. Ref. [125] demonstrated that with their data partitioning approach, only small memory storage is required, instead of duplicating the whole data set size over all the processing units.

Skipping can be used to decrease the executed workload and reduce computational costs. This method selectively skips certain computations based on their relevance to the final output or the predefined conditions. Skipping approximations can be performed by a simple calculation to evaluate if a more complex computation can be eliminated [126].

Memoization stores and reuses intermediate results of expensive computations in memory to avoid recalculation when the same input reappears [126, 127]. This can be especially beneficial in RNN or other architectures with repetitive computations, leading to improved hardware efficiency. Even the simple implementation of memoization in Ref. [127] could speed up different experiments with different workloads ranging from 7% up to 25%.

Layer pipelining splits the processing of different layers in NN into sequential stages that overlap in time to allow parallelization of the computation. This method [128] allows scalable model parallelism with high hardware utilization and training stability. Pipelining algorithm library, GPipe, from [128] is a library to train a giant NN, with efficiency (speeds up the process), flexibility (supports any deep network), and reliability (guarantees consistent training).

Dynamic computation graph allows the structure of the NN to change dynamically at runtime [129]. While the static graph fixes the structure of the NN (like the sequence of operations and connections) before the beginning of training, the dynamic graph constructs the structure of the NN on the fly during execution. This approach allows flexibility as it adapts the structure efficiently depending on varying input types and conditions.

Activation function approximation [130] simplifies the computation of nonlinear functions, such as sigmoid, and tanh, which are commonly used in the LSTM layer. The approximation can improve efficiency in hardware implementations. Standard

weight matrices, of a layer across multiple devices, facilitating parallel computations on these segmented chunks.

activation functions often involve exponentials, divisions, or other complex mathematical operations. These functions are computationally expensive and not ideal for resource-constrained hardware like FPGAs, ASICs, or other embedded systems. The common approximation methods are piecewise linear functions, look-up tables (LUT) and Taylor series expansions. This can reduce power consumption, decrease latency, and lower memory requirements. This method is explained in depth and used in a study case in Chapter 3.

2.5 Complexity Metrics: Training and Inference

After implementing the mentioned complexity reduction strategies, it is essential to evaluate their effectiveness. This section gives a comprehensive understanding of the model's complexity during the training and real-time inference phases on the target hardware platform. Fig. 2.8 and Fig. 2.9 show the metrics used to measure the complexity of training and inference phases, respectively.

2.5.1 Complexity Metrics for Training Phase

For training, the computational complexity of the NN should be appropriately evaluated, as it allows for efficient resource allocation and is useful for comparing different models to assess the efficiency and effectiveness of various model architectures. Several metrics can be used to assess the complexity of NNs as in Fig. 2.8, which can be categorized

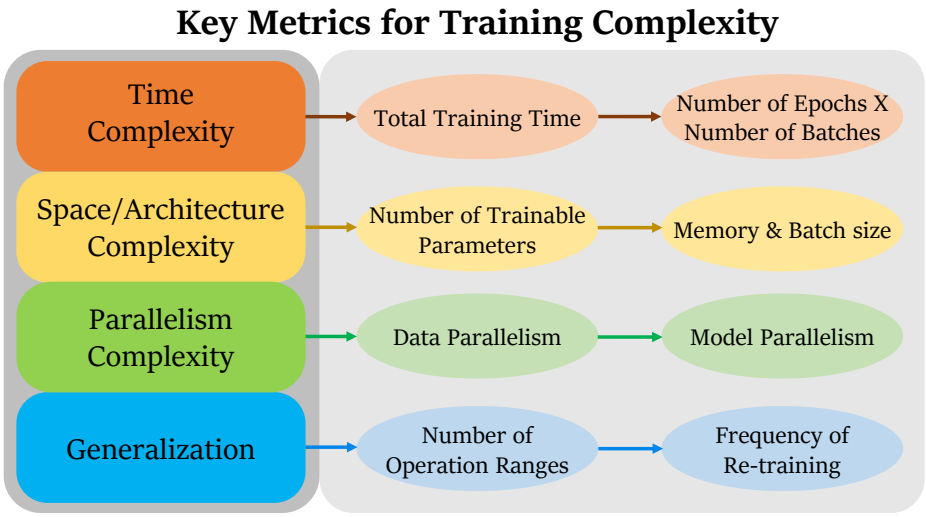


Fig. 2.8 Key metrics to evaluate complexity in the training phase for NN-equalizers.

into four key areas: time, space/architecture complexity, parallelism complexity, and generalization [55]. Adopting a multidimensional perspective is important because a single metric cannot provide a holistic understanding of the true complexity of training. Each dimension offers unique insights, guiding informed trade-offs between model performance, resource requirements, and generalization capabilities. Next, we will further detail each of these key areas of training complexity measurements.

Time Complexity: The traditional measures refer to the training time and the number of epochs required to achieve the desired performance. These metrics also take into consideration the learning rate and the optimizer used. Even though the training time metric and the number of epochs metrics show, to some degree, the training complexity, these metrics are a poor benchmark since the training time depends heavily on the hardware resources used and the size of the training dataset. In addition, two NNs with the same number of epochs to achieve the same performance can have very distinct training time, also depending on the batch size. To address these issues, an additional metric is proposed. The product of the number of epochs and the number of batches (NENB) [55], reflects the model's computational demand. More training epochs generally indicate a more complex and computationally demanding model. The number of batches refers to the number of subsets of data used during each epoch, which is affected proportionally by the dataset size and batch size. The number of epochs and the number of batches cannot be evaluated separately, as one model may require more epochs but fewer batches, while another model may require fewer epochs but more batches. Lastly, FLOPs (floating-point operations) can be used to measure the number of floating-point operations required to train the model. A higher number of FLOPs typically indicates higher time complexity.

Space/Architecture Complexity: The number of trainable parameters, while commonly used, may not fully capture the complexity due to different architectural designs. For example, two NNs with the same number of trainable parameters can have very distinct training complexity [131]. The model architecture indicates the complexity of the NN architecture itself, such as the depth, width (e.g., the number of layers and neurons) of the network, and specific architectural choices such as recurrent, or convolution. The next metric is the memory requirement for storing the weights, biases of the NN, and intermediate computations during training. The space complexity can be influenced by the batch size used during training. Larger batch sizes might require more memory, particularly on GPU devices.

Parallelism Complexity: This aspect consists of data parallelism and model parallelism. Data parallelism is the parallelization of training across multiple devices by splitting the dataset. Model parallelism is about distributing the model across different devices for processing. Parallelizability also refers to how scalable the training process is, with added computational resources and the efficiency of distributing the training process across multiple GPUs. For example, the MLP feed-forward NN is fully parallelizable and can result in faster training. To be more specific, the training time of RNN compared to the MLP with the same number of trainable parameters can be significantly longer, as the recurrent architecture of RNN is more complex than the feed-forward structure of the MLP.

Generalization: The flexibility/generalizability is assessed by estimating the number of operational ranges in which the NN equalizer operates with an acceptable gain. If the NN can only perform a specific task, it requires frequent re-training in the future, contributing to the overall complexity. Therefore, the NN that performs well in different but related tasks without re-training is preferable [108].

2.5.2 Complexity Metrics for Inference Phase

Evaluating computational complexity accurately is essential in the design of DSP devices. This will allow a better understanding of the implementation feasibility and bottlenecks within each device's structure. With this consideration, Fig. 2.9 summarizes the most commonly used measures for assessing computational complexity, from the software to the hardware level [55].

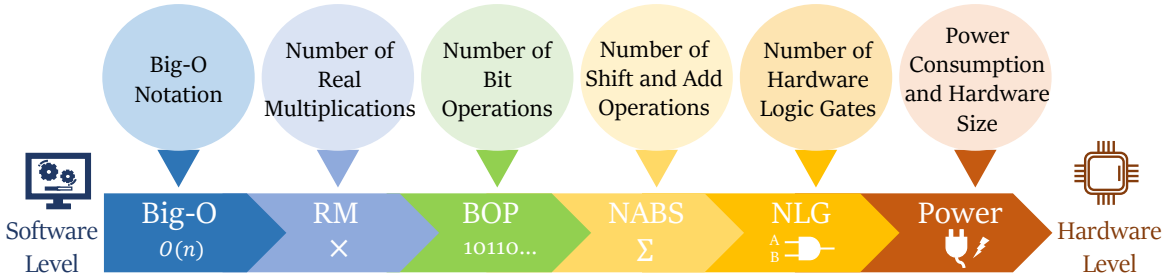


Fig. 2.9 Key metrics to evaluate computational complexity in the inference phase for NN-equalizers.

Big-O Complexity

Big-O notation is a key concept for evaluating an algorithm's computational complexity. It describes how runtime or resource usage scales with input size. Big-O (O) provides an upper bound on the worst-case scenario of the time required to operate. In addition, there are also Big-Theta (θ) and Big-Omega (Ω) notations that are the same family to compare the efficiency of different operations. Big-Theta (Θ) represents the exact bound, while Big-Omega (Ω) is the best case or the lower bound. For instance, when adding two integers with n digits, the computational complexity is $\Theta(n)$. This can be interpreted as the time required growing linearly with the number of digits. On the other hand, multiplication of two n-digit numbers results in $O(n^2)$ complexity. $O(n^2)$ indicates a quadratic growth in processing time, showing that multiplication is often more computationally expensive than addition.

Real Multiplications

One of the primary ways to evaluate the computational complexity of an algorithm is the number of real multiplications (RM) [132, 133], while ignoring additions. RM is a software-oriented estimation and is often defined per one processed element, such as per sample or symbol. RM only focuses on multiplications because, in both hardware and software implementations, the multiplier is generally the slowest element and consumes the largest chip area [134, 132]. In contrast, additions are generally inexpensive in terms of processing time and resource usage. For algorithms that use floating-point arithmetic with a fixed bit-width precision, RM offers a practical way to compare complexity. This metric is commonly used to benchmark the complexity in the DSP operations for optical channel equalization tasks [133]. In this thesis, we also adopt RM as the main complexity of the NN to benchmark against other methods.

Number of Bit-Operations

When transitioning to fixed-point arithmetic, it becomes necessary to consider the number of bit-operations (BOP) to evaluate computational complexity. BOP reflects the impact of varying the bitwidth precision on the complexity. This metric estimates the BOP required for fundamental arithmetic operations, such as addition and multiplication, based on the bitwidth of two operands. In essence, the BOP metric extends the concept of floating-point operations (FLOPs) to a more accurate measure of complexity in heterogeneously quantized NNs. FLOPs cannot efficiently evaluate

integer arithmetic operations [135, 136]. For BOP estimation, both multiplication and addition complexities must be assessed, since complexity is evaluated in terms of the most common operations in NNs: multiply-and-accumulate (MAC) operations [135, 136, 115]. BOP reflects how multiplier counts scale with operand bitwidth and adder counts with accumulator bitwidth. Since most DSP implementations rely on dedicated logic macros (e.g., DSP slices in FPGAs or MAC units in ASICs), BOP serves as an effective complexity metric by considering operand-specific bitwidths in MAC operations.

Number of Additions and Bit Shifts

With the development of advanced NN quantization techniques [137–140], fixed-point multiplications can be efficiently implemented using a few bit-shifters and adders [141–143]. BOP primarily counts bit-level operations from multiplications and additions but does not distinguish between standard multiplications and those optimized using bit-shifting techniques. Therefore, the NABS metric (the number of additions and bit shifts) is required to measure the complexity one step closer to the hardware level. NABS actually only counts the number of total equivalent additions to represent the multiplication operation, while neglecting the shift operations. This is because shift operations incur no additional hardware cost and execute in constant time with $O(1)$ complexity. Despite this, the term "number of additions and bit shift" is retained to emphasize that multiplications are now represented through shifts and adders.

Number of Logic Gates

Unlike NABS, which estimates computational complexity based on additions and bit shifts, the number of logic gates (NLG) is a hardware-level metric used to evaluate the actual implementation cost of a design on hardware platforms such as ASICs and FPGAs. In contrast to other complexity metrics, the NLG metric includes the cost of the activation functions, which are often implemented using LUTs rather than adders and multipliers, to reduce the complexity. Other relevant hardware-level metrics include flip-flops (FF), registers, general logic blocks, memory blocks, and specialized functional macros. Since NLG depends on the specific circuit design, there is no direct conversion from NABS to NLG. Tools such as Synopsys Synthesis [144] can estimate gate counts for ASICs. However, for the FPGA design, it is more challenging to correctly estimate the gate count from the report of FPGA tools [145]. For FPGA-based designs, NLG

can be estimated based on the number of logic gates per configurable logic block or logic cell [146].

Power Consumption and other aspects

Power consumption, memory footprint, and activation function complexity should be evaluated. Power consumption is critical as it can cause a bottleneck during the implementation. Memory usage depends on the input size of the time series and the number of parameters stored in the memory, considering the quantization scheme. Additionally, activation function complexity, including different function types and approximation methods, should be examined [25].

2.6 Conclusion

This chapter introduced NN-based equalizers for nonlinearity mitigation in optical communication systems. The chapter reviewed traditional equalization methods, including DBP, Volterra series-based approaches, phase conjugation, and perturbation-theory-based methods alongside machine learning-based techniques, emphasizing their advantages and challenges. The chapter's focus was to present different complexity reduction techniques and how to evaluate the complexity of the NNs. Complexity reduction techniques were introduced based on the different implementation phases: training, inference, and hardware synthesis. Likewise, the complexity metrics were divided into the training and inference phases. For the inference phase, the metrics can reflect the complexity from the software to hardware levels.

Chapter 3

Low-Complexity Techniques for NN-Based Equalizers

3.1 Introduction

Although NN-based optical channel post-equalization can offer lower computational complexity than traditional mitigation techniques like DBP [55], the computational complexity still hinders their practical implementation. Among the various NN architectures, RNNs, such as LSTM modules, have shown better equalization performance compared to feed-forward NNs when addressing nonlinear impairments [62, 73], since the LSTM layer is suitable for time-series processes. However, the computational complexity of the LSTM layer is still considerably high.

This chapter discusses the first aspect of the challenges: computational complexity. The reduction techniques used in the LSTM-based equalizers are investigated. Section 3.2, based on C1 [147] and J1 [148], considers the weight clustering method to reduce the computational complexity in the NN-based equalizer in a DSCM system, which is a more challenging transmission system compared to a single carrier transmission. Section 3.3, based on C4 [149] and J6 [25], examines the approximation techniques of the nonlinear activation functions in an LSTM-based equalizer, aiming to reduce the hardware resources required for the implementation.

3.2 Weight Clustering to Reduce Number of Real Multiplications

Several authors have investigated LSTM- or NN-based equalizers in single-carrier transmission systems [62, 73, 108], however, a limited number of works [1, 91] investigated the NN-based equalizer in the DSCM system. DSCM has recently emerged as an effective alternative to cope with the current rapid evolution of internet traffic, e.g. to limit effects such as equalization-enhanced phase noise (EEPN) [150]. It has also been proven to provide more robustness against nonlinearity distortion by optimizing symbol rates compared to single-carrier systems[90, 151]. DSCM and OFDM differ in how subcarriers are managed. While OFDM relies on overlapping subcarriers with carefully maintained orthogonality for efficient spectrum utilization, DSCM uses fully separated subcarriers with virtually zero frequency cross-talk, simplifying receiver design. Compared to OFDM, DSCM offers greater resilience to time-frequency synchronization challenges within DSP algorithms. OFDM operates at a much lower symbol rate, which complicates the detection of subcarriers, particularly when signals originate from multiple OFDM transmitters at varying distances [152]. Finally, DSCM provides significant flexibility, resulting in reduced capital and operational expenditures in optical networks [153, 154].

Bakhshali et al. [1] have already proposed NN architectures for optical channel nonlinearity compensation in DSCM systems. However, the complexity of their approach remains considerable. Therefore, further complexity reductions are paramount in the path towards real implementation and sustainable nonlinear equalizers.

This work investigates nonlinearity compensation in DSCM transmission systems by applying the NN-based post-equalizer. To allow the lower complexity, the model compression technique "weight clustering" is applied. Weight clustering reduces the number of effective weights used by the model, resulting in a significant decrease in computational complexity [16]. Note that the weight clustering method has already been investigated in the single-carrier transmission [16]. The contribution of this chapter can be summarized as follows:

- We demonstrate how the input structure and the NN architecture are adapted to simultaneously recover signals from all subcarriers in DSCM systems.
- We investigated the potential of weight clustering as a complexity reduction technique in an NN-based equalizer used in DSCM systems.

- We propose the low-complexity NN model that offers a reduction of up to 34% in computational complexity in terms of real multiplication per equalized symbols (RMpS), compared to the standard DBP 1 STpS (step per span) and up to 97.9% RMpS reduction compared to its uncompressed version.
- We propose NN models leading to optical performance similar to that reported in [1], but with a reduction of up to 91% complexity in RMpS.
- We compare the trade-off between the NN models' performance and complexity with different numbers of weight clusters (WC).

3.2.1 NN Architecture

The utilization of NN in DSCM is notably under-researched. Various architectures of NN have been proposed for nonlinear mitigation in single-channel single-carrier transmission systems, such as MLP, CNN, or RNNs. However, feedforward NNs like MLP and CNN have demonstrated inferior performance compared to RNN- or biLSTM-based architectures [62]. Due to the increased complexity in DSCM systems arising from the nonlinearities of adjacent subcarriers, the constraints of MLPs and CNNs are expected to be even more significant. Consequently, the NN architecture proposed here is a combination of the biLSTM and CNN model, as depicted in Fig. 3.1. The architecture based on an LSTM layer has shown superior performance in previous research works

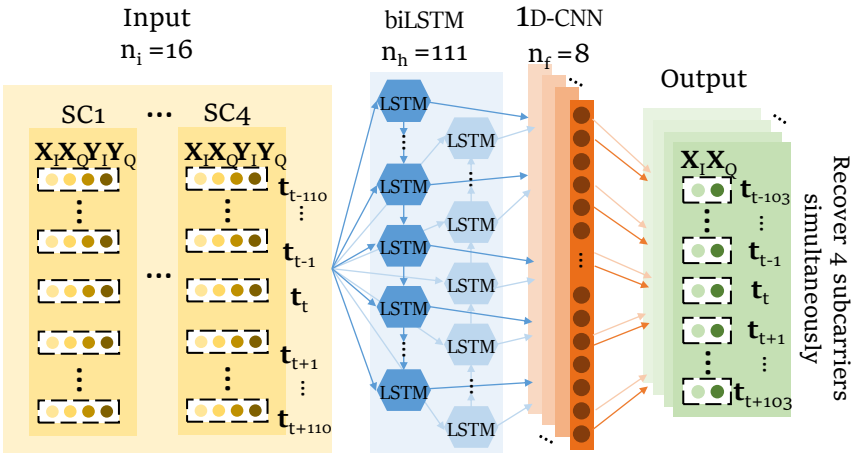


Fig. 3.1 The NN equalizer design employs biLSTM and 1D-CNN layers, utilizing the data from all four subcarriers as inputs to simultaneously recover symbols in the X polarization across all four subcarriers.

[1, 16, 73]. Unlike RNNs that face short-term memory limitations, LSTM networks are capable of learning long-term dependencies across time steps, which are essential to mitigate dispersion-induced memory effects and ISI caused by chromatic dispersion [73]. LSTM was specifically designed to overcome the gradient challenges associated with RNNs [71, 72]. The core properties of the LSTM-based layer are sequential processing and retaining past information through past hidden states. The biLSTM layer adopted here consists of two separate LSTM layers for forward and backward directions. The equations of the LSTM layer and its complexity can be found in Section 3.2.4. Note that all of the NN-based equalizers in this work operate at one sample per symbol.

In this work, the biLSTM layer has 111 hidden units (n_h), and the 1D-CNN layer adopts 8 filters (n_f) and a kernel size (n_k) of 15 with the linear activation function. This 1D-CNN layer has 8 filters to recover real and imaginary parts of X polarization[1] of all four subcarriers simultaneously. The input window (M) has a size of 221 input symbols. In order to equalize the signal in DSCM systems, the NN takes in 16 input features that consist of real and imaginary parts of X and Y polarization for four subcarriers. In this way, the NN can have sufficient knowledge to learn the pattern of nonlinear distortions like self-subcarrier nonlinearity (SSN) including those arising from the neighboring subcarriers, and mitigate the cross-subcarrier nonlinearity (CSN). The model performs a regression task to predict the real and imaginary parts of the recovered symbols or 207 symbols for each subcarrier per one inference step. To recover multiple symbols, it is necessary to account for the system memory length induced by chromatic dispersion in fiber communication systems. When the NN equalizer processes a window of M input symbols, only $M-x$ symbols can be reliably recovered, where x depends on the system's memory length. This is because the initial and final symbols in the input window lack sufficient information from their neighboring symbols (due to dispersion-induced memory effects), making them difficult to recover accurately. To address this, the dimensionality of the input window is reduced without losing crucial information by using a 1D convolutional layer. This layer processes the data with a kernel size n_k, zero padding, dilation, and stride set to 1. As a result, the size of the recovery window becomes $M-n_k+1$. The loss function used in this model is mean square error (MSE), and the optimizer is Adam with a learning rate of $5.81{\cdot}10^{-4}$. Adam is an optimization algorithm that extends Stochastic Gradient Descent (SGD) by using

[1]While it is possible to modify the NN output to recover both X and Y polarizations simultaneously, this was not tested in this study. For consistency, the CDC block and the NN equalizer were applied separately to each polarization.

adaptive learning rates and momentum to improve convergence speed and stability. The model with 214 hidden units of biLSTM is also included to enable better performance in Q-factor, which aligned with findings in [1] for comparison.

Bayesian Optimizer [62, 155] was utilized to optimize the NN hyperparameters. Unlike blind search methods like grid or random search, it leverages past evaluations to build a probabilistic model (often a Gaussian Process) of the objective function. This model helps it intelligently decide where to test next, balancing exploring new areas with exploiting promising ones. This leads to much faster and more efficient optimization. The optimized hyperparameters included the learning rate, batch size, number of hidden units in the biLSTM layer, and number of output window taps. A range of acceptable values for each hyperparameter optimized was defined to ensure that the model did not end up with very high complexity. For example, models with 214 hidden units were selected to achieve the highest Q factor, while models with 111 hidden units were chosen to demonstrate a reasonable trade-off between complexity and performance. Models with fewer hidden units were not included in this study, as the primary goal was to maintain a performance level comparable to existing benchmarks while optimizing computational efficiency.

3.2.2 Complexity Reduction of NN Using Weight Clustering

The weight clustering method illustrated in Fig. 3.2, which is also known as weight-sharing, is a model compression technique that reduces the computational complexity of NN by decreasing the number of distinct weight values used in the model. This method takes advantage of the observation that many connections in an NN can share the same weight value [113, 156] and, as a result, significantly reduce the number of unique multipliers needed during matrix multiplication. The shared weights can be defined by the centroids' initialization technique using K-means++, thus ensuring that multiple weights will converge to the nearest centroid. In addition, one can fine-tune those shared weights to improve accuracy[2]. The number of multiplications required in the NN is reduced by applying weight clustering, as multiple operations of the same values can be combined into a single multiplication. The number of distinct multipliers in matrix multiplication is reduced to at least the number of clusters per input element. For example, the weight matrix on the top left in Fig. 3.2 is the matrix W. To define

[2]This work carried out a two-step training method. The original model without weight clustering was first trained to achieve good performance and well-trained weights, after that the weights after clustering were fine-tuned to mitigate the approximation errors.

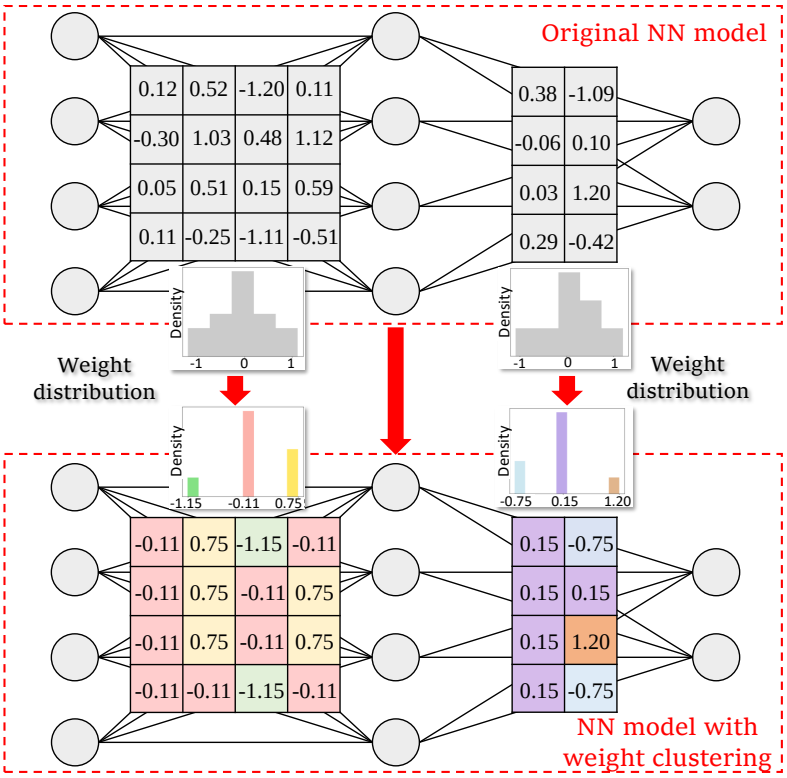


Fig. 3.2 Illustration of weight clustering framework where the trained weights of the original MLP model (top) are clustered into 3 weight clusters with the closest centroid.

the output vector O before clustering, the input vector I needs to be multiplied by the matrix W. In this example, multiplying W and I leads to 16 real multiplications (input size $\times$ hidden layer size $= 4 \times 4$), as follows:

$$O = W \times I = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}. \tag{3.1}$$

To cluster the weights into three clusters, as an example in Fig. 3.2 (bottom left), the centroids of 3 clusters are calculated as c_1, c_2, and c_3. This can be seen as:

$$O = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \begin{bmatrix} c_2 & c_3 & c_1 & c_2 \\ c_2 & c_3 & c_2 & c_3 \\ c_2 & c_3 & c_2 & c_3 \\ c_2 & c_2 & c_1 & c_2 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix}. \tag{3.2}$$

In this case, the multiplications can be rearranged by summing first the input elements that share the common weight clusters. The number of multiplications is reduced to 9 multiplications from the original 16 multiplications as follows:

$$O = \begin{bmatrix} o_1 \\ o_2 \\ o_3 \\ o_4 \end{bmatrix} = \begin{bmatrix} (i_1 + i_4)c_2 + i_2 c_3 + i_3 c_1 \\ (i_1 + i_3)c_2 + (i_2 + i_4)c_3 \\ (i_1 + i_3)c_2 + (i_2 + i_4)c_3 \\ (i_1 + i_2 + i_4)c_2 + i_3 c_1 \end{bmatrix}. \tag{3.3}$$

In the worst case, the number of multiplications would be 12 real multiplications (input size $\times$ the number of clusters = 4×3), as it carries out all possible unique multiplications and the rest are only additions. The benefits of weight clustering depend on the size of the input vectors, the weight matrices, and how the trained weight pattern spreads over the weight matrix.

After the weights are clustered, the fine-tuning process can be applied to mitigate the impact of clustering on the optical performance. This work follows the clustering algorithm [156] from the TensorFlow framework. During training, the gradients are computed with respect to these centroids, allowing the network to optimize while reducing the number of unique weight values. This is accomplished through a lookup table that maps weights to centroids during the forward pass and updates the centroids during backpropagation. This approach effectively reduces the number of distinct multipliers in matrix operations, resulting in significantly lower overall computational load. Additionally, weight clustering acts as a form of non-uniform quantization, where the NN's weight distribution is optimized to fit a limited set of values. This not only simplifies the hardware implementation and enables better memory efficiency, but also maintains the model's optical performance close to its original uncompressed state.

It is worth noting that the fine-tuning or training phase for weight-clustered models indeed involves additional computational cost due to an additional stage to fine-tune the clustered weights. However, this fine-tuning typically requires only about 10 to 20% of the original training epochs in this study. In this work, the main focus is on the complexity of inference, specifically in terms of real multiplications, as this metric directly reflects the cost of implementing the model in real-world deployments. It is common practice to train the model on a more powerful device and deploy it on a resource-constrained one. While training complexity can be assessed using metrics mentioned in Section 2.4.1, such as the number of trainable parameters, training epochs, or total training time [55], this analysis falls outside the scope of this study.

3.2.3 Data Generation, Training, and Evaluation

Data Generation Using DSCM Simulator

The numerical simulator created the dataset assuming a single-channel DSCM transmission with 4 subcarriers and 16-QAM DP modulation format. The total symbol rate is 32 GBd, resulting in 8 GBd per subcarrier. The transmission length is 40×80 km along standard single-mode fiber (SSMF) spans. The SSMF is modeled with an effective area of $80\mu m^2$, chromatic dispersion coefficient D = 17 ps/(nm·km), and attenuation parameter α = 0.2 dB/km. The block diagram of the system is reported in Fig. 3.3. Furthermore, this work has also considered transmission along TrueWave Classic (TWC) fiber with a total fiber length of 15×80 km and with the fiber parameters γ = 2.5 (W·km)$^{-1}$, D = 2.8 ps/(nm·km), and α = 0.23 dB/km.

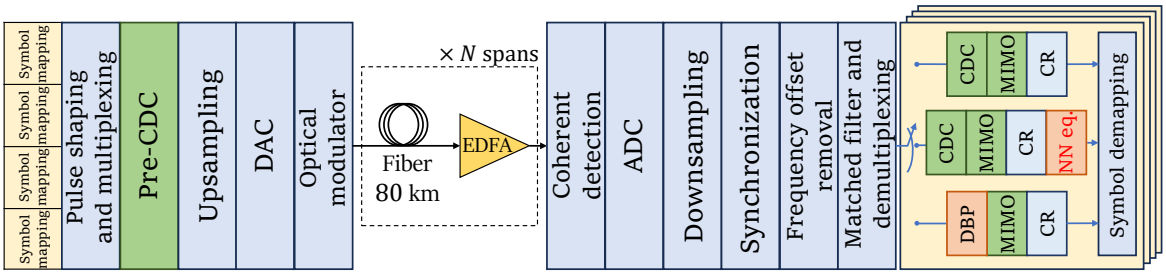


Fig. 3.3 Considered simulation setup.

The ideal electrical components[3] are assumed, namely, Mach-Zhender modulator, DAC, and ADC along with ideal TX/RX laser sources having zero linewidths and an ideal optical front-end at the receiver. Moreover, electronic noise sources are not considered for the simplicity of this analysis. At the transmitter side, the digital signals of each subcarrier are filtered by a digital root-raised cosine (RRC) filter with a 1/16 roll-off[4], shifted to different frequencies, and multiplexed in the frequency domain. Subsequently, the pre-CDC was performed, where 50% of the total dispersion is digitally pre-compensated. The full band signal is finally inverse Fourier transformed to the time domain and propagated. The propagation of the signal through the fiber was modeled

[3]The primary focus of this work is to demonstrate a proof-of-concept method for mitigating fiber-induced nonlinearities. In addition, in modern coherent transceivers, nonlinearities from devices such as the Mach-Zehnder modulator (MZM) are typically mitigated (i) using a digital pre-distortion module included within the DSP at the transceiver, or (ii) by operating the Mach-Zehnder in linear regime. Therefore, the electrical components are assumed to be ideal and their nonlinear effects are out of the scope of this work.

[4]Note that the channel parameters are set to match the parameters of the simulator in [1].

according to the well-known Manakov equation [157]. The simulations were carried out based on a full-band symmetric split-step method [14] with an adaptive step size [158]. At the end of each fiber span, optical fiber losses are perfectly compensated for by an EDFA with a 6 dB noise figure.

On the receiver side, all subcarriers are captured simultaneously in a single detection. The subcarrier multiplexed signal is digitized by an ideal ADC and re-sampled to twice the total symbol rate[5]. After the transform to the frequency domain, each subcarrier is successively shifted to the baseband and filtered out using a digital matched RRC filter. Finally, the low-symbol-rate signal in each subcarrier is down-sampled at 2 samples per symbol. Standard DSP algorithms [159] are used for the post-processing of the received signals from each subcarrier, independently. Digital CDC is considered at the front-end of the DSP unit for the post-compensation of 50% of the total link dispersion when only linear equalization is considered. NLC is not performed. For the case of the 16-QAM format, a training symbol-assisted decision-directed least mean square (DD-LMS) with 1024 training symbols and a 21-tap filter to implement the MIMO equalization [159] is used. A feedforward carrier recovery (CR) based on maximum likelihood estimation [160] is adopted for phase correction under correlated nonlinear phase noise. Within the CR algorithm, a joint CR method [161] has been chosen for cycle slip corrections under non-differential phase coding. Afterward, the BER analysis is performed based on the statistical Monte Carlo method. Finally, the received symbols are used as inputs for the NN. The NN operates at one sample per symbol.

Note that for SSMF 40×80 km, the simulator used in this study matched the optimum power and Q-factor of [1]. The performance in the nonlinear regime of the simulator in this work and the simulator in [1] is almost indistinguishable, while in the linear regime, the simulator in this study provided a slightly better Q-factor than the simulator in [1]. However, the main focus remains the nonlinear regime.

NN Training

The training datasets were generated with a random bitstream consisting of 2^{19} symbols. For each epoch, a subset of 2^{18} symbols was randomly selected from this dataset as input symbols to train the model. For testing and validation, the dataset contained 2^{17} unseen symbols. All models in this work were trained, validated, and tested with the same dataset size. The training was carried out for 1000 epochs and a mini-batch size

[5]Oversampling at three or more samples per symbol can have a slight improvement in the DBP algorithm (see [17]).

of 3824. The mini-batch input of the NN was structured in three dimensions [62]: (B, n_s, 16) where B is the mini-batch size and n_s is the number of time steps or the memory size depending on the number of neighbor symbols considered, N, as $n_s = 2N+1$. The last dimension, 16, corresponds to the number of features for each symbol across all four subcarriers. The models used four input features per subcarrier (4 subcarriers$\times$ 2 polarizations$\times$ 2 complex value components), derived from the in-phase and quadrature components of the complex signal (X_I, X_Q, Y_I, and Y_Q), where $X_I + jX_Q$ and $Y_I + jY_Q$ are the signals in the X and Y polarizations, respectively. The NN output is designed to recover the real and imaginary parts of multiple symbols across all four subcarriers in the X polarization simultaneously. The output batch shape is defined as (B, $n_s - n_k + 1$, 8), where $n_s - n_k + 1$ is the number of symbols recovered in the output and the third dimension is 8 as it refers to the real and imaginary parts of the X polarization of four subcarriers.

The weights of the trained models were saved at the epoch where the BER on the test dataset was at its minimum, a technique known as early stopping. Note that at the end of every epoch, the trained model is tested with the validation dataset to evaluate the performance in BER. Once the model was trained for a specific launch power, transfer learning [98] was employed to transfer the learned knowledge to different launch powers, thereby accelerating the training process for these new conditions.

Once the uncompressed trained models were trained, the original model with 111 hidden units was selected for compression. The model is compressed by the weight clustering framework, in this case from TensorFlow [156]. This study considers 25, 35, and 45 WCs for SSMF and 10, 20, and 30 WCs for TWC in order to evaluate the trade-off between performance and complexity. The specific values of weight clusters were determined through a grid search. The goal was to identify the number of weight clusters that would provide performance comparable to the state-of-the-art work presented in [1]. This ensures a fair comparison while demonstrating the effectiveness of the proposed method.

Benchmarking Models

To benchmark the performance of the proposed approach, the NN models in this work are compared against several established methods (Fig. 3.4): CDC, ideal DBP with 20 STpS, standard DBP with 1 STpS, advanced DBP (ADBP) - subcarrier multiplexing (SCM) [92] with 1 STpS, and the NN architecture from [1].

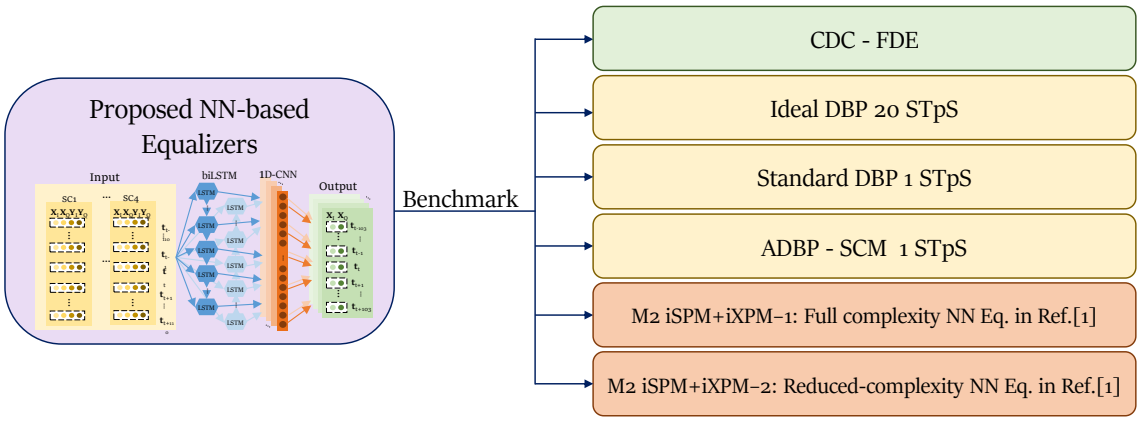


Fig. 3.4 Proposed NN models are compared against several benchmarking models.

The CDC method is the linear approach to compensate for chromatic dispersion by dynamically adjusting the signal with the inverse of the chromatic dispersion transfer function, based on the estimated chromatic dispersion after transmission. In this work, both pre- and post-CDC are deployed, each compensating for 50% of the accumulated chromatic dispersion[6].

The ideal DBP with 20 STpS is included to approximate the theoretical performance by considering the entire bandwidth and reversing the Manakov equation [162]. The complexity of this method will not be benchmarked, as the ideal DBP faces implementation challenges [163], with complexity beyond what is considered in this study.

The standard DBP [17] with 1 STpS and 2 Samples/symbol is performed after demultiplexing at the subcarrier level, which means that each subcarrier is processed independently. This method approximates the inverse of the Manakov equation for each subcarrier, accounting only for SSN, but ignoring CSN. This approach reduces the computational complexity compared to full-bandwidth DBP, therefore, it is also considered as a benchmark for complexity.

The ADBP-SCM, applied to DSCM systems [92], takes into account both SSN and CSN. The CSN compensation is carried out on the basis of the analytical model of time-varying XPM distortion proposed for WDM systems. This method adjusts the standard DBP to account for the nonlinear interactions between subcarriers, leading to improved compensation for nonlinear impairments. The ADBP-SCM considers both the

[6]The performance of 50 % pre-CDC together with 50% post-CDC is not significantly different than the 100% post compensation, however, we keep the former approach to be compatible with the simulation of [1]

CSN-induced cross-polarization modulation and phase noise. It achieves good overall performance in scenarios where the CSN significantly affects signal quality.

Lastly, the state-of-the-art approaches for NN-based equalizers applied to DSCM systems proposed in [1] are considered. In this case, the authors have demonstrated the performance of the Q-factor with different complexity constraints, when training was based on perturbation coefficients. The model with the best performance is called "M2 iSPM+iXPM". Both the Q-factor performance and the computational complexity of the proposed NN in this study are benchmarked against the M2 iSPM+iXPM models. In this work, "M2 iSPM+iXPM - 1" refers to their M2 iSPM+iXPM model with a full complexity limit of 5×10^5 RMpS, whereas "M2 iSPM+iXPM - 2" corresponds to the reduced complexity model with a complexity limit of 2.5×10^5 RMpS.

3.2.4 Complexity Analysis

This section outlines the complexity analysis for assessing the computational complexity of NN layers, both with and without weight clustering, in comparison to conventional techniques like CDC and DBP, including scenarios with and without XPM compensation. Complexity is quantified using RMpS.

Complexity of Original NNs

In this work, the NN model consists of biLSTM and 1D-CNN layers. The complexity equations for this NN model are reported in detail in [55]. Starting with the biLSTM layer, first, the standard LSTM layer is assessed. The formulas are presented in Section 2.2.1. The number of RM of an LSTM layer is:

$$\mathrm{RM}_{\mathrm{LSTM}} = n_s n_h (4n_i + 4n_h + 3), \tag{3.4}$$

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 one can see in Eq. (2.23). Therefore, $4n_h n_i$ and $4n_h^2$ are obtained, respectively. Moreover, the complexity equation also needs to include the element-wise product operated three times in Eq. (2.23), which costs $3n_h$. Finally, the process is repeated n_s times; therefore, n_s is multiplied by the total number.

For the 1D-CNN layer, the equation describing this layer is present in Eq. (2.20). Note that the output size can be computed by:

$$\text{OutputSize} = \left[\frac{n_s + 2\,padding - dilation(n_k - 1) - 1}{stride} + 1\right], \tag{3.5}$$

where n_s is the input sequence size. In this case, with padding equal to 0, dilation equal to 1, and stride equal to 1, the output size is equal to $n_s - n_k + 1$. To calculate the complexity of a 1D-CNN layer, the number of RM can be formulated as:

$$\text{RM}_{\text{CNN}} = n_f n_i n_k \cdot \text{OutputSize}, \tag{3.6}$$

where n_f is the number of filters, also known as the output dimension, n_i is the number of features in the input vector and n_k is the kernel size. Note that the input of the 1D-CNN layer is the output of the biLSTM layer. As in Eq. (3.6), there are $n_i n_k$ multiplications per sliding window, and the number of times that the sliding window process needs to be repeated is equal to the output size. The procedure is then repeated for all n_f filters [55].

As a result, the complexity of the biLSTM+1D-CNN layer in terms of the number of RMpS can be calculated by:

$$\text{RMpS}_{\text{NN}} = \frac{\text{RM}_{\text{LSTM}}^{\text{forward}} + \text{RM}_{\text{LSTM}}^{\text{backward}} + \text{RM}_{\text{CNN}}}{(n_s - n_k + 1) \cdot n_{\text{subcarriers}}}. \tag{3.7}$$

The total RM of the NN is divided by $(n_s - n_k + 1) \cdot n_{\text{subcarriers}}$, because the NN recovers $n_s - n_k + 1$ for all subcarriers simultaneously. To be specific, the last 1D-CNN layer has $2 \times n_{\text{subcarriers}}$ filters (number 2 represents real and imaginary parts of the symbol), and each filter has the output size of $n_s - n_k + 1$.

Complexity of Weight-Clustered NNs

In this section, the complexity of the NNs when the weight clustering technique [16] is evaluated, as explained in Section 3.2.2, is applied. In the biLSTM layer, the input weight matrix W is assumed to have c_i clusters, and the recurrent kernel weight matrix U is assumed to have c_h clusters. In the worst case, the number of unique real multiplications would be reduced to $n_i \times c_i$ multiplications involving matrix W and $n_h \times c_h$ multiplications involving matrix U. Therefore, the number of RM for a clustered

LSTM layer is:

$$\mathrm{RM}_{\text{clustered LSTM}} = n_s\left(n_i c_i + n_h c_h + 3n_h\right), \tag{3.8}$$

For the 1D-CNN layer, it is supposed that there are c_j clusters in each filter n_f. Normally, each filter has a size n_k and the input has n_i features, resulting in $n_i \times n_k$ multiplications per filter application. With weight clustering, the number of unique weights in each filter is reduced to c_j clusters. Therefore, it only requires c_j multiplications per filter application. Deriving from Eq. (3.6), the number of RM of a cluster 1D-CNN layer reads as:

$$\mathrm{RM}_{\text{clustered CNN}} = \left(n_o c_j\right) \cdot \text{OutputSize}. \tag{3.9}$$

Lastly, the total RMpS of the clustered model can be calculated in the same way as in Eq. (3.7), but with the clustered versions of these equations.

Complexity of CDC and DBP

According to Ref. [55], the computational complexity of CDC using the frequency domain equalizer (FDE) is as follows:

$$\mathrm{RM}_{\mathrm{CDC}} = 4 \cdot \left(\frac{N(\log_2 N + 1)q}{N - N_D + 1}\right). \tag{3.10}$$

Here, N represents the FFT size, q is the oversampling ratio, and $N_D = q\tau_D/T$ where τ_D/T is the dispersive channel impulse response, and T is the symbol interval. This complexity calculation includes two polarizations, which require four N-point FFTs and $2N$ complex multiplications. The factor of 4 accounts for the fact that one complex multiplication equals four real multiplications. The term $N - N_D + 1$ indicates the number of useful samples based on the overlap-save algorithm for blockwise FD filtering. Note that optimization of FFT size is crucial to minimize complexity.

For the standard DBP technique [17], DBP is performed at a subcarrier level independently. Thus, the same formula as for a single carrier channel is considered. The RM of the standard DBP is [55]:

$$\mathrm{RM}_{\mathrm{DBP}} = 4qN_{\mathrm{Sp}}N_{\mathrm{STpS}}\left(\frac{N(\log_2 N + 1)}{N - N_{Dq} + 1} + 1\right), \tag{3.11}$$

where N_{Sp} is the total number of spans, N_{STpS} is the number of propagation steps per span, and q is the oversampling factor. The complexity of each DBP step includes the

linear part, which matches the RM of the CDC, and a nonlinear part, represented by a single RM corresponding to the multiplication with a nonlinear term.

For the ADBP-SCM, the RM per symbol is derived from the formula of complex multiplication per sample in Ref. [92], as follows:

$$\mathrm{RM}_{\text{ADBP-SCM}} = 4q\Bigg[(M+1)\frac{K_{\mathrm{CD}}(\log_2 K_{\mathrm{CD}}+1)}{K_{\mathrm{CD}}-P_{\mathrm{CD}}} + M\left(6.5+\frac{1.5K_{\mathrm{NL}}(\log_2 K_{\mathrm{NL}}+N_{\mathrm{sc}}-1)}{K_{\mathrm{NL}}-P_{\mathrm{NL}}}\right)\Bigg], \tag{3.12}$$

where the multiplier 4 refers to 4 real multiplications per one complex multiplication, q is the oversampling ratio, M is the number of steps, K_{CD} is the subcarrier block size for the frequency domain CDC filter obtained by the overlap and add (OLA) method, P_{CD} is the overhead of CDC filter for each subcarrier; $P_{\mathrm{CD}} = (1+\rho)\pi\beta_2(L_{\mathrm{step}})/T_s^2$, K_{NL} denotes the OLA block size of the CSN low-pass filter, P_{NL} represents the CSN low-pass filter overhead; $P_{\mathrm{NL}} = NL_{\mathrm{span}} + \frac{2}{\alpha}\Delta\beta'_{\mathrm{max}}/T_s$. Here, $\Delta\beta'_{\mathrm{max}} = \pi(1+\rho)\beta_2(N_{\mathrm{sc}}-1)/T_s$ and T_s is the sampling rate for one subcarrier. In this analysis, K_{CD} is set to 1024 and K_{NL} is set to 128.

3.2.5 Results and Discussion

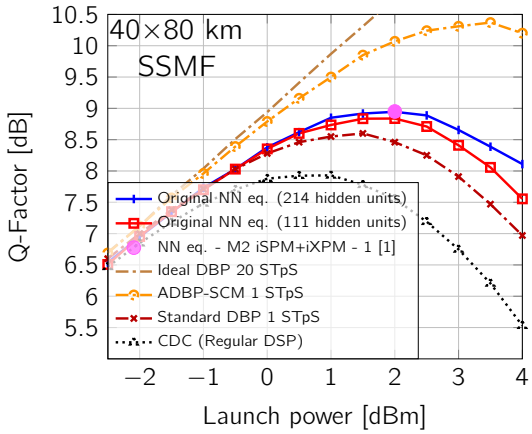
Equalization Performance

This thesis presents the optical performance in terms of the Q-factor which is calculated directly from the BER using:

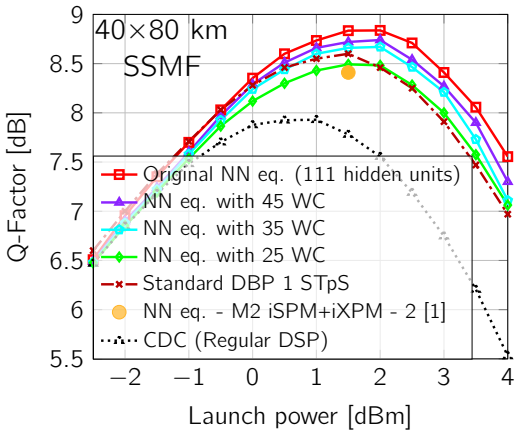
$$Q = 20\log_{10}\left[\sqrt{2}\mathrm{erfc}^{-1}(2\mathrm{BER})\right]. \tag{3.13}$$

First, the performance in Q-factor of the original NNs (without WC) is compared with analytical approaches, namely, CDC, ideal DBP 20 STpS, standard DBP [17] 1 STpS, and ADBP-SCM [92] 1 STpS, which are described in Section 3.2.3. Fig. 3.5a presents the Q-factor as a function of the optical launch power of the original NN models proposed: with 214 and 111 LSTM hidden units. The ideal DBP 20 STpS is used to provide a reference optical performance close to the ideal one. The results show that the original models with 214 and 111 hidden units provide a Q-factor improvement of 1 dB and 0.9 dB, respectively, when compared to employing CDC only. The optimal launch power also increases from 1 dBm to 2 dBm in this case. Both NN models also

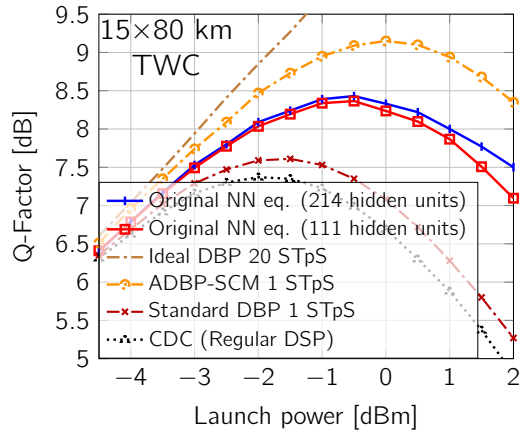
outperform the standard DBP with 1 STpS by 0.35 dB for the model with 214 hidden units and 0.24 dB for the model with 111 hidden units. As expected, the approach with higher complexity (higher number of hidden units) leads to the best optical performance. However, compared to the ADBP-SCM 1 STpS approach, the former still leads to a 1.4 dB higher Q-factor.



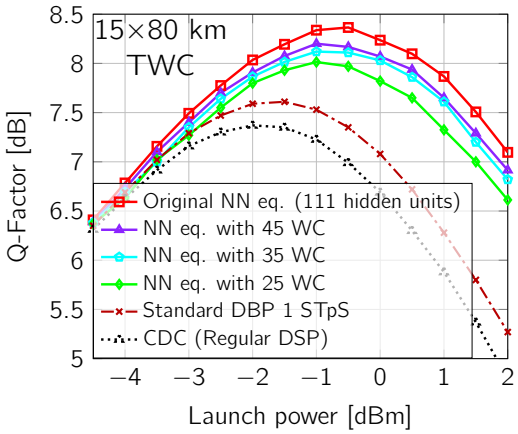
(a) 40 × 80 km SSMF - Original NNs (without complexity reduction) compared with NN model from Ref. [1] with full complexity (M2 iSPM+iXPM-1), different approaches of DBP and CDC.



(b) 40 × 80 km SSMF – Reduced-complexity NNs with different numbers of WC, compared to reduced-complexity NN model (M2 iSPM+iXPM-2) from [1], standard DBP and CDC.



(c) 15 × 80 km TWC – Original NNs (without complexity reduction) compared with different approaches of DBP and CDC.



(d) 15 × 80 km TWC – Reduced-complexity NNs with different numbers of WC, compared with standard DBP and CDC.

Fig. 3.5 Q-factor as a function of the launch power for the NN-based equalizers with the original complexity (left) and the reduced complexity (right), compared to the CDC and different approaches of DBP.

Fig. 3.5b shows the Q-factor as a function of the optical launch power of the original NN model proposed with 111 hidden units and compares it with the performance of its compressed versions (with weight clustering) with different numbers of WC: 25, 35, and 45 clusters. It can be observed that with a lower number of WC, the optical performance decreases. The NN with 45 WC experienced a 0.1 dB reduction in Q-factor compared to the original NN, while the NN with 35 WC showed a decrease of 0.17 dB. The NN with 35 WC demonstrated a performance close to the standard DBP 1 STpS. For the NN with 25 WC, the Q-factor was reduced to around 8.5 dB which is comparable to the M2 iSPM+iXPM - 2 [1] model and faced a drop of 0.35 dB from the original NN. Note that all compressed models still maintain the optimum power of 2 dBm.

Considering the 15×80 km TWC system, Fig. 3.5c shows the Q-factor versus launch power for the original NNs with 214 hidden LSTM units and 111 hidden units compared with ideal DBP 20 STpS, ADBP-SCM [92] 1 STpS, standard DBP with 1 STpS and CDC as baselines. The ideal DBP 20 STpS provides a reference optical performance close to the ideal one. The model with 214 hidden units has a slight Q-factor improvement compared to the model with 111 hidden units, however, it comes with the cost of more complexity in terms of hidden units. Similar to the SSMF case, the performance curves of the original NN with 111 hidden units and its compressed versions in Fig. 3.5d exhibit a similar trend to that of the SSMF system. However, the NN-based equalizers significantly outperformed the DBP 1StpS which only compensated for self-phase modulation. In this case, increasing the number of steps per span in the standard DBP did not further improve the Q-factor because TWC's high nonlinearity coefficient made inter-SC cross-phase modulation more significant [164]. It can be observed that even the compressed NNs can partially compensate for the CSN in the DSCM systems, showing the obvious improvement from the standard DBP. This figure clearly reflects the trade-off between CC and performance; the higher number of WCs, the better the Q-factor. In this case, the optimal Q-factor for the original NN, and the NNs with 45 WC, 35 WC and 25 WC were 8.36, 8.20, 8.12 and 8.01 dB, respectively.

Computational Complexity

This section highlights the computational complexity comparison of the proposed models and the benchmarking models.

First, to demonstrate how the weight distribution of the original NN models changed when the weight clustering was applied, Fig. 3.6a shows the weight distribution of the original uncompressed NN and Fig. 3.6b for the NN with 25 WC. One can observe that

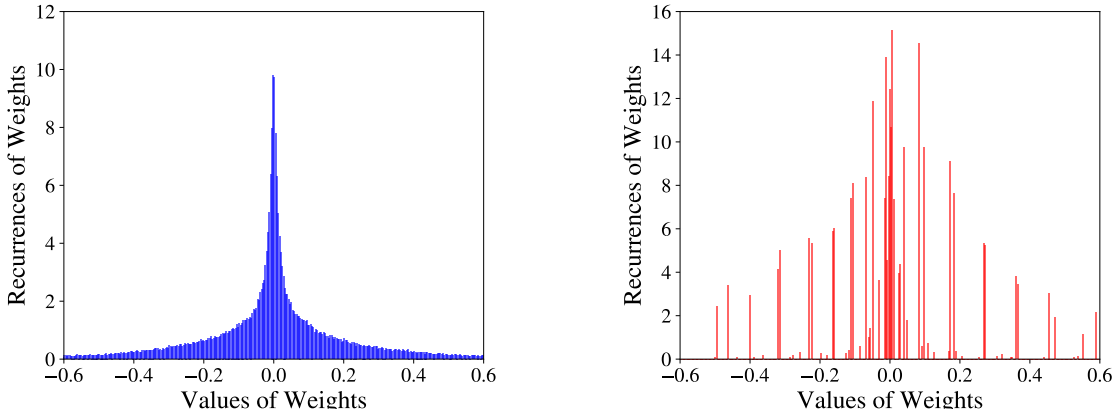
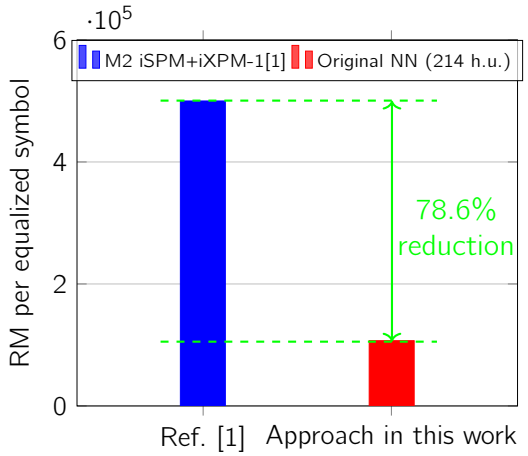


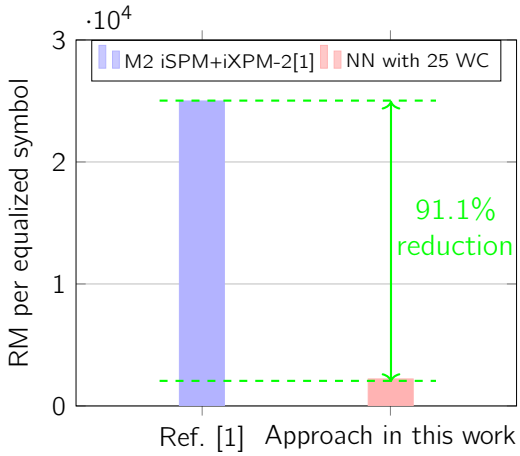
Fig. 3.6 Comparison of weight distributions of all concatenated layers of (a) the original model (without weight clustering) and (b) the model with 25 weight clusters.

the unique number of weights is drastically reduced. To be more specific, there are 1.39×10^5 unique weights in the original model but only 125 unique weights in the NN with 25 WC. In summary, weight clustering is a powerful technique to reduce the computational complexity of NN-based equalizers in optical communications. This approach enables the NN-based equalizers to be more suitable for real-time implementation while retaining their effectiveness in mitigating fiber nonlinearity.

For 40×80 km of SSMF system, compared to [1], with the same transmission scenarios, the simulator in this work provided the same optimal Q-factor and the comparable Q-factor in the nonlinear regime. Ref. [1] has shown their "M2 iSPM+iXPM" models with different complexity budgets. Among these models, their two most outstanding models are utilized as benchmark: the model with the highest Q-factor (M2 iSPM+iXPM-1) and the model with the lowest complexity budget (M2 iSPM+iXPM-2). Their model with the highest performance with full complexity provides up to around 8.95 dB Q-factor, at the cost of 5.0×10^5 RMpS, while the model with the lowest complexity of 2.5×10^4 RMpS provides a Q-factor of around 8.5 dB. Fig. 3.7a presents the complexity comparison of the original model with 214 hidden units and their M2 iSPM+iXPM-1 model, considering the same performance of 8.95 dB Q-factor. It can be seen that the original model with 214 hidden units requires around 1.07×10^5 RMpS, which is 78.6% fewer RMpS than their M2 iSPM+iXPM-2 model. Similarly, Fig. 3.7b shows the complexity of the proposed NN with 25 WC and their model with the lowest complexity budget (M2 iSPM+iXPM-2). It highlights that with a similar level



(a) The original NN with 214 hidden units in this work compared to the full-complexity NN model (M2 iSPM+iXPM-1) from Ref. [1], with the same Q-factor of 8.95 dB.



(b) Reduced-complexity NN with 25 WC, compared to reduced-complexity NN model (M2 iSPM+iXPM-2) from Ref. [1], with the same Q-factor of 8.5 dB.

Fig. 3.7 Complexity in RMpS comparison of the models in this study and the models M2 iSPM+iXPM in Ref. [1], for 40×80 km of SSMF.

of performance, the proposed NN with 25 WC enables a 91.1% reduction in complexity or around 2215 RMpS.

Fig. 3.8 illustrates the complexity reduction of the compressed model compared to the original model with 111 hidden units. The compressed model with 25 WC provided 97.9% reduction from the original model, at the cost of 0.45 dB Q-factor drop.

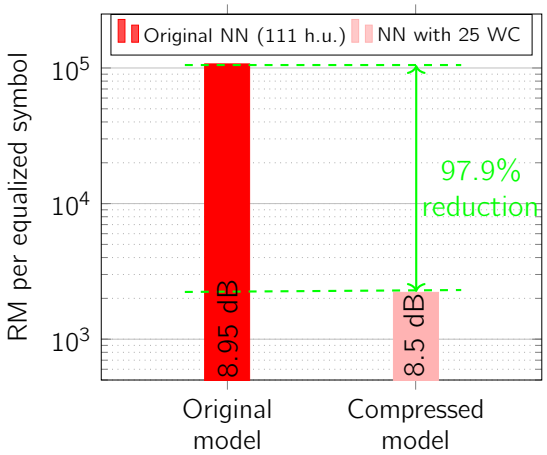
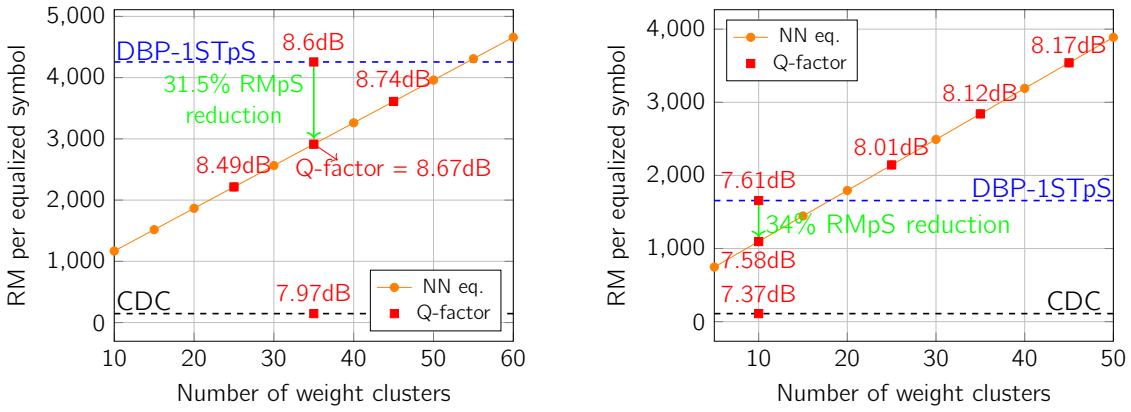


Fig. 3.8 Complexity in RMpS comparison of the original model with 111 hidden units and its compressed version with 25 WC for 40×80 km of SSMF.



(a) 40×80km of SSMF (b) 15×80km of TWC

Fig. 3.9 Comparison of complexity (RMpS) between biLSTM+1D-CNN and traditional channel equalizers (DBP 1STpS and CDC) as a function of the number of clusters utilized in weight clustered compression.

Next, it is crucial to evaluate the trade-off between performance and complexity in weight clustering for NN. In this regard, Fig. 3.9a illustrates the complexity in terms of RMpS as a function of the number of WC, using CDC (FDE) and standard DBP 1 STpS as benchmarks for 40×80 km of SSMF system. The NN with 35 WC achieved performance levels comparable to standard DBP 1 STpS, but with a 31.5% reduction in complexity (RMpS). However, when comparing the complexity of the NN equalizer to that of CDC, it becomes clear that significant improvements are necessary to achieve lower complexity levels, particularly in terms of multiplicative operations. Notably, while CDC typically requires around 150 multiplications per recovered symbol, the compressed NN still requires between 2000 and 3000 multipliers in its lower complexity configurations. For the standard DBP 1 STpS, more than 4200 RMpS are typically necessary.

Similarly, for 15×80 km of TWC system, Fig. 3.9b illustrates the trade-off between performance and complexity for the NNs with varying numbers of WC, using the standard DBP with 1 STpS and CDC as benchmarks. As expected, a higher number of WC results in a higher Q-factor. However, the improvement of the Q-factor is not linearly increased with the number of WC, indicating that beyond a certain threshold, additional WC contribute marginally to performance enhancement. To achieve comparable performance to the standard DBP at 1 STpS, the model required 10 WC, resulting in a 34% reduction in RMpS. Note that the standard DBP with more steps per span did not improve the performance due to its limitation in compensating for the CSN.

In this study, it can be observed that aggressive weight clustering (reducing the number of clusters to very low values) is less effective for DSCM systems than for simpler single-carrier single-channel systems. For instance, Ref. [16] demonstrated that as few as three weight clusters could achieve performance comparable to DBP in single-carrier systems. However, for the more complex DSCM system, at least 35 clusters were needed for SSMF transmission and 10 clusters for TWC transmission to achieve similar performance to standard DBP with 1 STpS. This is also because the NN in DSCM is designed to recover four subcarriers simultaneously, adding difficulties to the weight clustering process. These findings highlight the increased computational and modeling demands of DSCM systems, where the balance between performance and complexity becomes particularly critical. The ongoing challenge of optimizing NN equalizers to reduce computational complexity, especially in terms of multiplication operations, while maintaining high performance, needs to be further investigated.

To show the comprehensive result comparing to the previous literature, Table 3.1 summarizes different equalization approaches for the 40×80 km SSMF systems, including CDC, ideal DBP, standard DBP, ADBP-SCM, the models from previous literature [1], and the NN models in this work with different complexity limits. The table shows their Q-factor performance, optimum launch power, and computational complexity in terms of RMpS.

The current approach is to recover only one polarization at a time. Although technically feasible to modify the output of the model to simultaneously recover the X and Y polarizations, this approach was not explored in this study. For consistency,

Name	Method	Average Optimal Q-factor (dB)	Optimum Launch Power (dBm)	Complexity (RMpS)
CDC	Linear	7.90	1.0	146
Standard DBP 1 STpS	Nonlinear	8.60	1.5	4256
ADBP - SCM 1 STpS[92]	Nonlinear	**10.37**	**3.5**	1.07×10^5
NN Eq. - M2 iSPM+iXPM-1 [1]	Nonlinear	8.95	2.0	5.0×10^5
NN Eq. - M2 iSPM+iXPM-2 [1]	Nonlinear	8.41	1.5	2.5×10^4
Original NN Eq.(214 hidden units)	Nonlinear	8.95	2.0	1.07×10^5
Original NN Eq. (111 hidden units)	Nonlinear	8.84	2.0	3.1×10^4
NN Eq. with 45WC	Nonlinear	8.74	2.0	3611
NN Eq. with 35WC	Nonlinear	8.67	2.0	2913
Proposed NN Eq.with 25WC (Proposed)	Nonlinear	8.49	2.0	**2215**

Table 3.1 Summary of the different equalization approaches, with their Q-factor performance, optimum launch power and complexity in terms of RMpS for 40×80 km of SSMF.

the CDC block and the NN equalizer were applied independently to each polarization channel. It is worth noting that during the inference phase, the computational complexity in terms of RMpS would not increase if two separate models were deployed for the two polarizations. This is because the total number of recovered symbols would also double, maintaining the RMpS at the same level. Moreover, modifying a single model to recover both polarizations could potentially reduce RMpS, as a shared architecture could leverage additional efficiencies while increasing the number of symbols recovered per inference.

3.3 Approximation of Nonlinear Activation Functions

This section considers the approximation of the nonlinear activation functions in NN. In this thesis, the main architecture is based on biLSTM, because of its performance and aforementioned advantages in the time-series processing. In an LSTM cell, the sigmoid and tanh functions are deployed as activation functions[7], and they are computationally expensive. Both functions contain exponential functions, making it difficult to implement them on resource-constrained hardware and requiring a large chip area in the real implementation [165]. The implementation of NNs' nonlinear activation function is one of the crucial components in the hardware design of NN. In contrast to the hardware realization of the NN's weights and inputs, which can be readily proceeded from the float to fixed-point representation [55], the activation functions' realization in hardware is not straightforward. This is because activation functions directly influence the final

[7]Empirically, the simpler types of activation functions, such as ReLU or leaky ReLU, were tested in the NN-based equalizers in this study. However, the performance has dropped drastically.

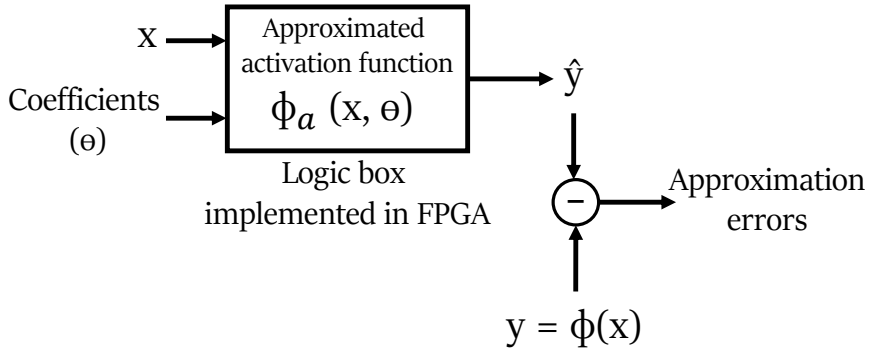


Fig. 3.10 Diagram of the input/output of approximated activation functions based on the logic box implemented in FPGA.

output of each layer. Moreover, while the overparameterization of NN weights can help mitigate quantization errors, this benefit does not directly translate to the realization of activation functions. Therefore, function approximation techniques are required in place of the exact functions to reduce the overall computational complexity and to realize them in the resource-constrained hardware like FPGA [166, 165, 167, 168]. This study focuses on approximating the sigmoid and tanh. Three different approximation methods are considered: Taylor series expansion, piecewise linear (PWL), and look-up table (LUT). The scenario, in which the training of the NN with approximated activation functions is undertaken to reduce the approximation errors, was also investigated. Finally, the amount of resources required to implement the activation functions within the FPGA, is also evaluated.

As shown in Fig. 3.10, to implement the approximated activation functions on the FPGA, the FF[8], LUT[9], and DSP slices[10] are used to build the logic box[11], which takes the value x and coefficients to return $\hat{y}$. The coefficients are stored in the memory as input. The coefficients define the Taylor and PWL approximations, while the LUT approximation represents the quantization levels list. $\hat{y}$ is the output of the approximated activation functions, while y represents the actual output of the float-precision activation function. The difference between $\hat{y}$ and y is the approximation error.

The expression for the tanh function via exponential is:

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}, \tag{3.14}$$

while that for the sigmoid function reads as:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{3.15}$$

[8]FF is a basic digital storage element in an FPGA, used to store the value of a digital signal and can be used in conjunction with LUTs to implement sequential logic, such as state machines and counters.

[9]LUT is a basic building block of an FPGA used to implement equations built from Boolean logic functions, such as AND, OR, and XOR, or to store pre-calculated values for use in arithmetic or other operations.

[10]DSP blocks or slices are specialized components within an FPGA that are designed specifically for processing digital signals. They contain dedicated hardware resources such as multipliers, adders, accumulators, and registers that can perform complex mathematical operations at high speeds.

[11]FPGA uses LUTs, FF and DSP slices together to implement the digital logic, memory, and computation required by the intended applications. LUTs, FFs, and DSPs are all programmable, meaning that the user can reprogram the FPGA's logic, memory, and computation elements to suit different applications.

3.3.1 Taylor Approximation Approach

In the Taylor series approximation, the higher the degree of an approximating polynomial n, the better the approximation. The tanh Taylor series reads as:

$$\begin{aligned}\tanh x &= \sum_{n=0}^{\infty} \frac{2^{2n}(2^{2n}-1)B_{2n}}{(2n)!} x^{2n-1}, \text{ where } |x| < \frac{\pi}{2} \\ &= x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \frac{62x^9}{2835} - \dots,\end{aligned} \tag{3.16}$$

where B_{2n} denotes the Bernoulli number [169], $-a_t < x < a_t$, and a_t is the boundary of the approximation region: when x is not within $[-a_t, a_t]$, the approximation error is essential. Therefore, it is important to choose the value of a_t that maximizes performance. Empirically, the slight difference in the value of a_t can noticeably affect the performance. When x is outside the Taylor series approximation region, the value of tanh x is set to -1 or 1, according to the following expression:

$$\tanh x = \begin{cases} 1, & \text{if } x > a_t, \\ x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + \frac{62x^9}{2835}, & \text{if } -a_t < x < a_t, \\ -1, & \text{if } x < -a_t. \end{cases} \tag{3.17}$$

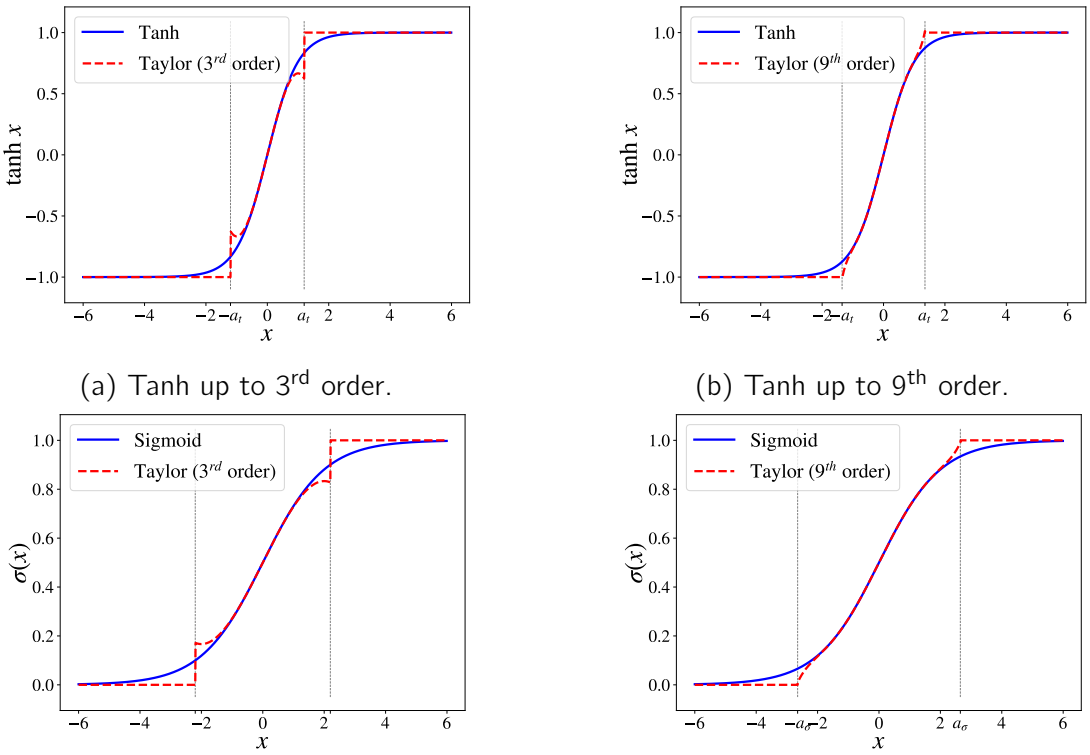
The plots for the different order Taylor approximations are given in Fig. 3.11a–3.11b. The value of a_t is the result of the grid search, which maximizes the performance of the NN-based equalizer without retraining.

The Taylor series for the sigmoid function is:

$$\begin{aligned}\sigma(x) &= \frac{1}{2} + \frac{1}{2}\tanh\frac{x}{2} \\ &= \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + \frac{x^5}{480} - \frac{17x^7}{80640} + \frac{31x^9}{1451520} - \dots,\end{aligned} \tag{3.18}$$

where $-a_\sigma < x < a_\sigma$ and a_σ is the point where the Taylor series approximation of the sigmoid starts to diverge. Similarly to tanh, the values of the sigmoid approximation in regions less than $-a_\sigma$ and greater than a_σ are set to 0 and 1, respectively, as follows:

$$\sigma(x) = \begin{cases} 1, & \text{if } x > a_\sigma, \\ \frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + \frac{x^5}{480} - \frac{17x^7}{80640} + \frac{31x^9}{1451520}, & \text{if } -a_\sigma < x < a_\sigma, \\ 0, & \text{if } x < -a_\sigma. \end{cases} \tag{3.19}$$



(a) Tanh up to 3rd order.

(b) Tanh up to 9th order.

(c) Sigmoid up to 3rd order.

(d) Sigmoid up to 9th order.

Fig. 3.11 Taylor series approximation of tanh (a) – (b) and sigmoid functions (c) – (d).

The Taylor approximation plots corresponding to Eq. (3.19), when the highest order of the polynomial is 3 and 9, are given in Fig. 3.11c–3.11d.

The performance in terms of Q-factor is evaluated when the approximation for both tanh and sigmoid functions is carried out simultaneously, with different orders of the approximating polynomial up to 9th order. The values of a_t and a_σ are chosen by using the grid search, aiming to maximize the Q-factor when replacing the exact activation functions with their Taylor series approximation without retraining the weights. The Taylor series approximation reduces the computational cost and time required to compute the activation function considerably, compared to the processing using the original function [167].

3.3.2 Piecewise Linear Approximation Approach

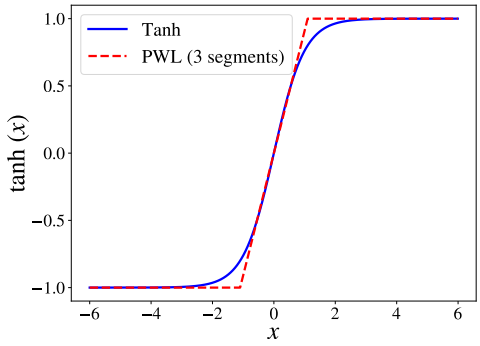
The PWL approximation, introduced in [170], is a combination of linear segments that approximates the activation or nonlinear function [130, 166]. Increasing the number of

linear segments to represent the nonlinear function allows us to achieve better accuracy. The PWL approximation is a promising method to reach a higher processing speed since it consumes fewer resources on FPGA[12] compared to the Taylor approximation: to reach higher accuracy, the Taylor approach fits the nonlinear function with high-order expressions, which results in the consumption of resources, while the PWL can reach the same level of accuracy with the use of more segments, but without employing high-order operations [166].

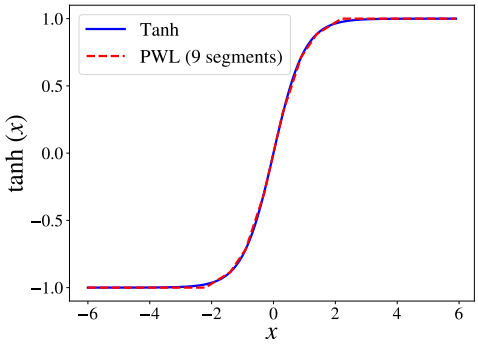
This study compares the performance of the NN-based equalizers when applying 3-, 5-, 7-, and 9-segment PWL approximations to both tanh and sigmoid[13]. The expressions for the PWL used in this study are included in Table A.1 in Appendix A.1. The corresponding plots for the equations mentioned in Table A.1 with 3 and 9 segments

[12] [24] shows that the implementation of PWL can be further optimized to have zero multipliers by simplifying the shift and addition operations.

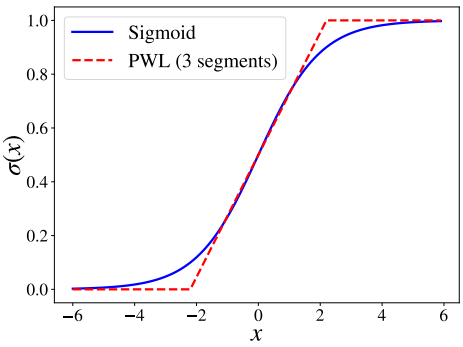
[13] Note that when the number of segments is lower than 3 segments that used to represent sigmoid or tanh in the biLSTM cell, the biLSTM model in this case is not able to learn to mitigate the approximation errors.



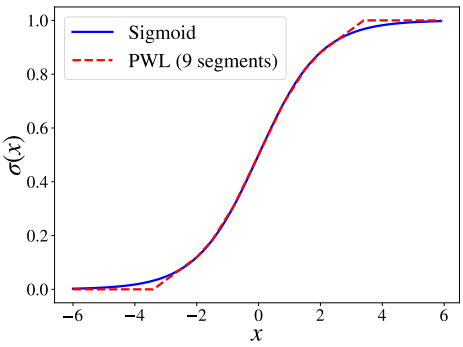
(a) Tanh with 3 segments.



(b) Tanh with 9 segments.



(c) Sigmoid with 3 segments.



(d) Sigmoid with 9 segments.

Fig. 3.12 PWL approximation of tanh (a) – (b) and sigmoid functions (c) – (d).

are depicted in Fig. 3.12a–3.12b for tanh, and Fig. 3.12c–3.12d for sigmoid. Note that grid search is used to find the coefficients for each expression, aiming to maximize the performance in terms of Q-factor, after the actual activation functions are replaced by the approximations over the trained weights, instead of minimizing the difference/areas between the exact function and the approximation curves. It is carried out because, in this case, minimizing the difference/areas between the curves noticeably degrades the Q-factor performance of the NN equalizer when the NN predicts the output with the replaced approximated activation functions.

3.3.3 Lookup Table Approximation Approach

The LUT approximation is a commonly used method for the activation functions' hardware implementation [171]. The LUT approximates the function with a limited number of uniformly distributed points. This approach offers a high-performance design, and the fastest implementation compared to other methods. At the same time, a large amount of memory is required to store the LUT on the hardware [172, 173]. The chip area requirements for the LUT approximation grow exponentially with the required approximation accuracy [173]. The number of bits used to represent values in the LUT directly affects the approximation error and the required memory size. An example of the LUT approximation of tanh with the number of bits equal to 4 is presented in Fig. 3.13.

The LUT approach is similar to traditional quantization, in which full precision values are assigned to uniform quantization levels, i.e. the value x is mapped to $\hat{x}$ which is the closest value of x in the quantization level list [174]. The LUT stores the values

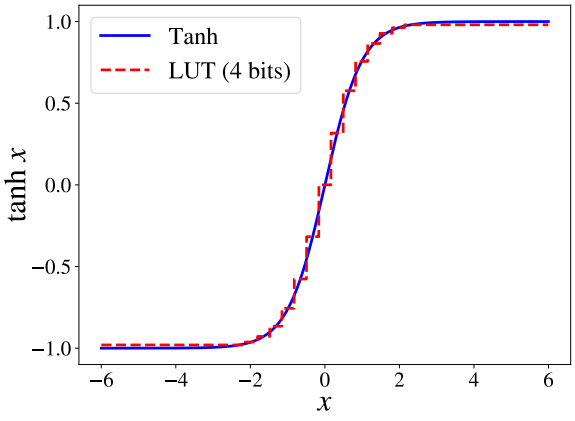


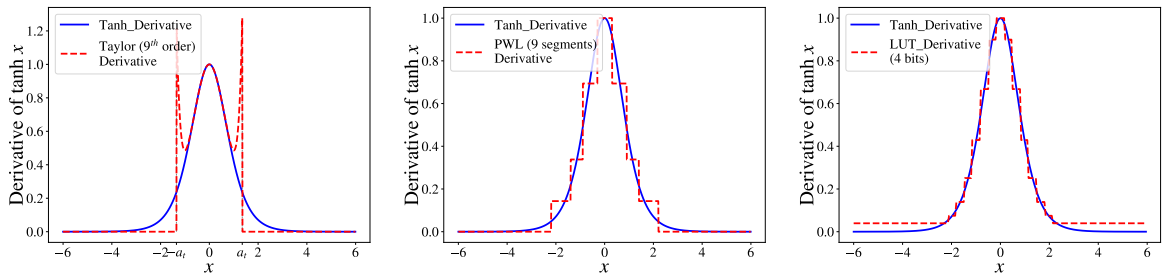
Fig. 3.13 LUT approximation of tanh function with the number of bits equal to 4.

of the quantization levels ($\hat{x}$) and their corresponding $f(\hat{x})$, in this case $\tanh(\hat{x})$ or $\sigma(\hat{x})$. The difference between the exact value $f(x)$ (the blue curve in Fig. 3.13) and the approximation $f(\hat{x})$ (the red curve in Fig. 3.13) introduces the approximation errors.

This study investigates the Q-factor performance of the model for the LUT representation of activation functions when the number of bits used ranges from 2 to 16.

3.3.4 Reducing Approximation Error through Learning via Stochastic Gradient Descent

Once the activation functions are replaced by the approximation, the NN performance can drastically drop (up to 5 dB when the approximation is the least complex). However, training the model with approximated activation functions can enhance the performance because the model learns to reduce the approximation error. SGD is the training approach applied in this work. The training can be undertaken from scratch, which means that the NN is trained when the activation functions are replaced by approximations from the beginning without any pre-assigned weights. Another approach to training is to use the weights of the model pre-trained with the true activation functions, then re-train the model after the replacement of the approximations to learn the approximation errors. The latter results in a considerably shorter training time. Only the results of the second method are reported because, empirically, training from scratch takes significantly longer to converge and sometimes can provide even worse results. It is worth noting that another available training approach is to only train the coefficients



(a) Gradient of Taylor approximation.

(b) Gradient of PWL approximation.

(c) Gradient of LUT approximation.

Fig. 3.14 The derivative of the tanh function for the approximations using (a) Taylor series with the highest order of 9, (b) PWL with 9 segments, (c) LUT approximation with the number of bits equal to 4.

of the Taylor and PWL equations without retraining the NN weights; however, in this study, the performance was not acceptable when using a low number of segments in PWL and training with this approach.

To train the NN with the approximation of the activation function via the SGD, the gradient of the approximation function must be computed. For the Taylor approximation, the Taylor series gradient is calculated with respect to the Taylor series approximation equations Eq. (3.17) for tanh and Eq. (3.19) for sigmoid. Fig. 3.14a shows an example of the derivative of the tanh approximation using the Taylor series with the highest order of 9; the gradient (red curve) is not smooth due to the polynomial nature of the Taylor series. This fact can limit the training ability, especially when training from scratch, as noted in Section 3.3.6. Concerning the PWL, the gradient is the slope of the expressions from Table A.1 (in the Appendix section). Fig. 3.14b depicts the gradient of the PWL approximation with 9 segments. Note that due to the non-differentiability of LUT, it is challenging to learn the LUT-approximated model [174]. In this work, to train the LUT, the LUTs of the gradients are generated for both sigmoid and tanh for each interval of the LUT approximations. Fig. 3.14c shows the gradient of the tanh LUT with 4 bits, corresponding to the tanh approximation in Fig. 3.13.

3.3.5 Methodology

NN Architecture

The equalizer in this work, as depicted in Fig. 3.15, contains a biLSTM layer with 35 hidden units and a linear 1D-CNN layer with a kernel size of 21 without padding and 2 filters to recover real and imaginary parts in its output. The biLSTM+CNN equalizer takes 81 symbols as inputs and retrieves 61 equalized symbols at the output. The input vector contains four features from four real values (X_I, X_Q, Y_I, and Y_Q) from X and Y polarizations, derived from real and imaginary values of $X_I + jX_Q$ and $Y_I + jY_Q$, respectively. The time domain depth is an additional dimension that characterizes the system's memory. Accordingly, the input shape can be described as (Batch size, Memory, 4).

This equalizer is pre-trained with the actual activation functions which provide the best performance. The training adopts the MSE loss estimator and the classical Adam algorithm for the stochastic optimization step [175]. The training uses a mini-batch size equal to 2001 and a learning rate equal to 0.0005, which were found by the Bayesian optimization procedure described in [16]. The training set contains 2^{20} symbols, and,

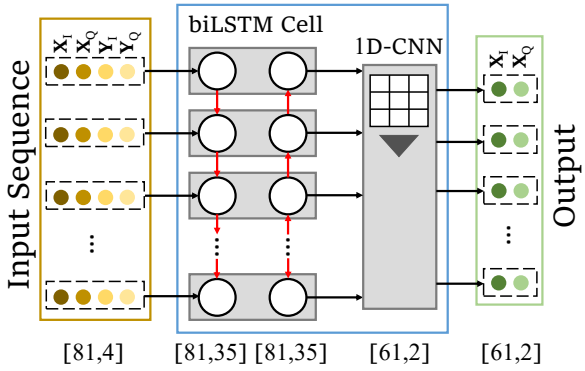


Fig. 3.15 biLSTM+CNN equalizer structure used for activation function approximation analysis.

at every epoch, 2^{18} input symbols were randomly chosen from this dataset to train the model. For the testing and validation, an unseen dataset with 2^{18} symbols was utilized.

After the model was trained with original activation functions for the best performance, the NN's activation functions (sigmoid and tanh) were replaced by the different approximation techniques. Then to alleviate the approximation errors, the weights of the NN are re-trained.

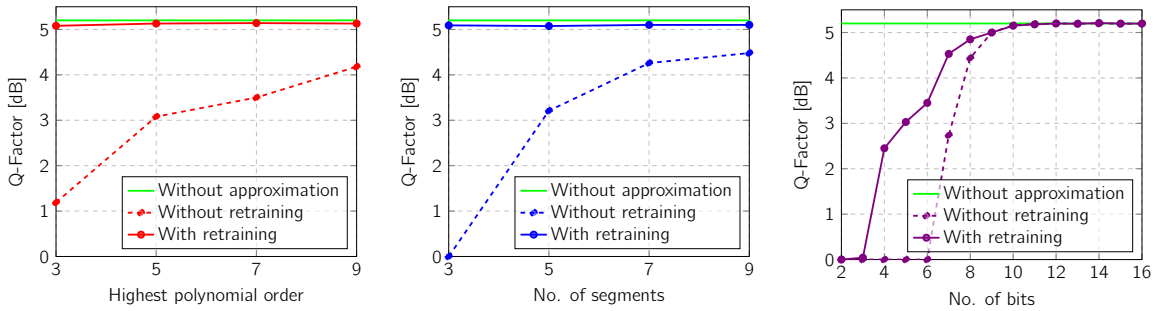
Simulation Setup

The numerical simulator created the dataset by assuming the transmission of a single-channel 34 GBd, 16-QAM DP channel along 17×70 km Large Effective Area Fiber (LEAF) spans. The signal propagation through the fiber was represented by a generalized Manakov equation split-step Fourier method [14]. The parameters of the LEAF are: the attenuation coefficient $\alpha = 0.225$ dB/km, the chromatic dispersion coefficient $D = 4.2$ ps/(nm·km), and the effective nonlinear coefficient $\gamma = 2$ (W· km)$^{-1}$. At the end of each fiber span, optical fiber losses are compensated for by an EDFA with a 5 dB noise figure. Downsampling and CDC were performed on the receiver end. The CDC was performed in the frequency domain with the transfer function of the chromatic dispersion given by [14]: $G(z,\omega) = \exp\left(-\frac{j\omega^2\beta_2 z}{2}\right)$ where ω is the angular frequency, β_2 is the group delay dispersion parameter of the fiber and z is the transmission length. Afterward, the received symbols were normalized and used as inputs of the NN.

3.3.6 Performance versus Complexity Investigation

Next, this subsection investigates the performance of the models when applying different approximation techniques for nonlinear activation functions: Taylor series, PWL, and LUT. Fig. 3.16 depicts the Q-factor in the optimal power (2 dBm)[14] after equalization for three scenarios: the original NN without approximation, the NN with approximation (without retraining), and the NN with approximation (with retraining). Note that training the NN from scratch when replacing exact activation functions with approximations takes a considerably longer time to converge than retraining the original NN after replacing floating-point activation functions with their approximations. The training from scratch with the Taylor and LUT activation approximation even results in lower eventual performance. Therefore, in this figure, only the results of the retraining approach are reported. Fig. 3.16a and Fig. 3.16b, corresponding to the Taylor series and PWL, respectively, reveal the same trend. Without training, as the complexity of the approximation increases, the NN equalizer performs clearly better. However, with training, the increasing complexity barely improves the performance: the NN is able to adjust its parameters to mitigate the approximation error and provides comparable performance to the NN without approximations. The Q-factor versus complexity (order) of approximation plots, Fig. 3.16, highlight the remarkable performance gain in all considered approaches when the model with activation functions replaced by the approximation were retrained. It can also be seen that training can mitigate the errors from the approximation. This means that even the low-order approximations, such as

[14]This work is a part of the journal paper [25].



(a) Taylor series approx. (b) PWL approximation. (c) LUT approximation.

Fig. 3.16 Q-factor versus complexity in terms of polynomial order for the Taylor approximation, pane (a), in terms of the number of segments for PWL approximation, (b), and in terms of the number of bits for the LUT, (c).

the simplest PWL with three segments, can still yield results nearly identical to those rendered by the original activation functions.

Fig. 3.16c shows the performance of the LUT approximation. When replacing the activation functions with LUT without retraining, a certain number of bits is needed to provide an acceptable Q-factor level[15]. For example, the minimum number of bits needed to provide a Q-factor greater than zero is 7 bits; 9 bits are needed to provide performance comparable to the model without approximation. On the other hand, when retraining the NN after the approximation, the Q-factor for the lower number of bits (from 3 to 7 bits) considerably increases. In this case, the non-differentiability makes the training challenging and limits the reachable performance in training, but the improvement is still noticeable when the number of bits is between 3 and 7. Fig. 3.17 shows the convergence speed of the three approximation techniques. It can be seen that the learning of Taylor and PWL is similar, whereas the retraining of LUT approximation is more difficult. Although the LUT gradient, Fig. 3.14c, and the PWL gradient in Fig. 3.14b seem interchangeable, the forward propagation of the LUT approximation is still discrete, which means that with the lower number of bits, a gap between each quantized level becomes larger. Thus, small changes that the gradient makes to update the weights might not change the quantization level to the next value. This means that the loss region is the same as it was in the last NN training interaction (trapped in a local minimum). Notably, in [176] a similar circumstance was observed; the previous

[15]Note that this study followed the LUT implementation from Ref. [171, 172] which presented the LUT with equal x-error intervals. The alternative approach (activation functions with equal y-error intervals) can be used, but in this case, there is only a slight improvement in the Q-factor when the number of bits is greater than 5, and with the retraining, the performance is very close to the x-interval approach.

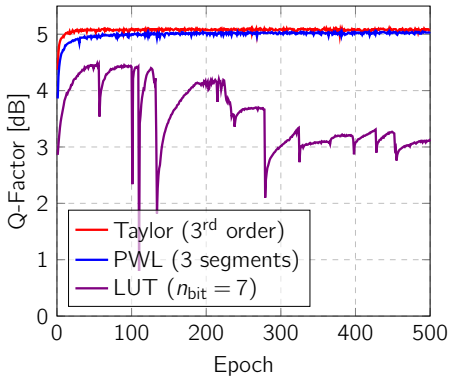


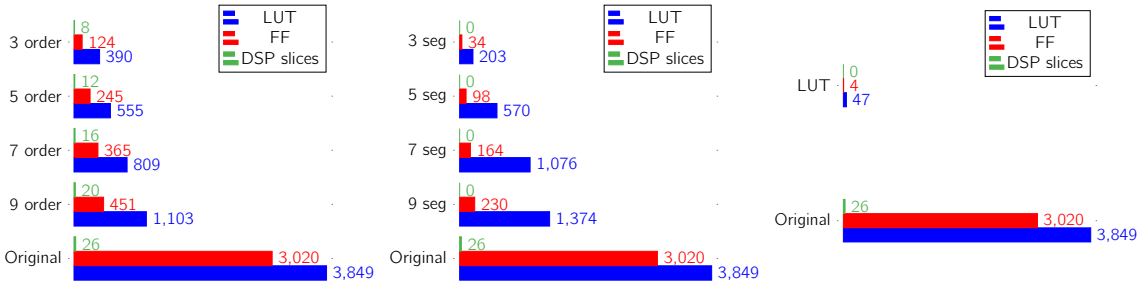
Fig. 3.17 Convergence study of the retraining to mitigate the approximation errors of Taylor series (3rd order), PWL (3 segments), and LUT ($n_{bit} = 7$) approximations.

reference also pointed to the instability of the training that can occur with a quantized activation function. In the case of PWL, the learning is more stable due to the continuity of the function's approximation, as each weight update generates a new loss value and a distinct point in the forward propagation.

In addition, as anticipated, it can be observed that when quantizing the LUT below 4 bits, no acceptable Q-factor can be reached even after the retraining. The reason for this is that when the activation function is quantized, unlike when the weights are quantized, the ability to represent the modulation of the equalized signal is limited. In this situation, the signal adopts 16 QAM, which requires at least 4 bits to represent a constellation data point. However, as can be observed, even 4 bits are insufficient in this case to preserve all the essential features for the equalization process when using the quantization of the activation function. When more bits are used, a better Q-factor can be achieved; however, more memory is then required to represent the quantization. It is worth noticing that when the number of bits is greater than 10, the Q-factor no longer improves in both scenarios (with and without retraining).

In FPGA, the resources in terms of LUT, FF, and DSP slices are used to build the logic behind the functionality of the approximation. In this work, the hardware realization of the tanh function is undertaken on the EK-VCK190-G-ED Xilinx FPGA chip[16]. The amount of resources required (in terms of LUT, FF, and DSP slices) in the FPGA when using the approximations for tanh, is compared to that when applying the actual tanh activation function in Fig. 3.18. This figure depicts the resources used to build the logic behind the functionality of each approximation. Note that the coefficients and values used in each panel of Fig. 3.10, are considered an input of the implemented

[16]The detailed explanation on FPGA realization was explained in [25]



(a) Taylor series approx. (b) PWL approximation (c) LUT approximation

Fig. 3.18 Tanh implementation complexity in terms of LUT, FF, and DSP slices for the Taylor series, PWL, and LUT approximations after the Xilinx realization pipeline.

box, which is accessed by the FPGA memory. The implementation complexity of the actual float activation functions is significantly higher than that of the approximations. In the Taylor series approximations, Fig. 3.18a, the number of FF and LUT used to implement the approximations is drastically reduced compared to the original activation functions; to be more specific, when the highest order of the polynomial is 9, the number of FF required decreases by 6.7 times, and the number of LUT required is three times smaller. In terms of DSP slices, the 9^{th} order approximation requires 6 DSP slices fewer than the original functions. As the approximation becomes simpler, the implementation requires fewer resources, as expected. For the PWL approximation, no usage of DSP slices is required for the implementation. Like in the Taylor series approximation, the number of FF and LUT required decreases noticeably. Compared to the original float-precision activation function, the PWL with 9 segments requires 2.8 times less LUT, and 13 times less FF. As the complexity decreases according to the number of segments, fewer resources are needed. Turning to the LUT approximation, it does not require any of the DSP slices as well, and the number of LUT and FF decreases by a factor of 80 and 775, respectively. Regardless of the number of bits in the quantized activation function, approximately the same amount of resources is required to implement the logic of the LUT approximation (see Fig.3.18c). The LUT approximation approach is an algorithm based on evaluating the closest value in the LUT from a certain input and determining the memory address index that corresponds to that closest value to retrieve the information. As the number of bits increases, a larger memory is needed to store the LUT approximation points, which are a quantized version of the function. However, this memory usage is not accounted for in this study because this is considered one of the inputs to the implemented box.

In conclusion, when performance, memory, and resources are considered, the PWL emerges as a viable candidate for hardware implementation, particularly, the 3-segment PWL variant with retraining. When the model learns to reduce approximation errors, the Q-factor of 3-segment PWL can reach a level comparable to that of the original activation functions; in addition, there is no need for DSP slices, resulting in more efficient use of resources than the Taylor approximation. With this, the RAM usage in the PWL is efficient because only a few coefficients of the approximation must be saved, whereas the LUT, which brings about difficulties during the retraining process, requires that all values of each quantization level be saved, resulting in an exponential increase in memory usage as the number of bits increases.

3.4 Conclusion

This chapter focused on reducing the computational complexity of NN-based equalizers while maintaining equalization performance in Q-factor. The weight clustering technique was investigated to lower the number of real multiplications of the NN-based equalizer used in digital subcarrier multiplexing systems. Additionally, approximation techniques for nonlinear activation functions of the NN were analyzed. The result demonstrated that approximation of the activation functions can reduce hardware resource usage without severely affecting accuracy. A selection of the appropriate complexity reduction strategies can lead to considerable improvements in computational efficiency. These insights set the foundation for further enhancements in parallelization and generalizability, which are discussed in the following chapters.

Chapter 4

Parallelization of Recurrent NN-Based Equalizer via Knowledge Distillation

4.1 Introduction

NN-based equalizers' practical implementation in real-world systems has been hindered by major challenges like computational complexity and the requirement for high-speed processing. As previously observed in previous works [62, 73, 177], RNNs have outperformed feedforward NNs in addressing nonlinear impairments. The feedback loop structure of RNNs (e.g., biLSTM) presents inherent parallelization challenges despite its benefit to learn the temporal sequence [178, 179]. The parallelization is necessary to enable high-throughput and low-latency hardware implementations in modern optical networks.

This chapter, based on C5 [180] and J3 [181], introduces a novel method that uses knowledge distillation (KD) to solve the parallelization problem of RNN-based equalizers. In particular, the proposed method converts the sequential biLSTM-based equalizer into a feedforward structure that can be parallelized. KD, a technique traditionally applied in classification tasks, involves transferring knowledge from a larger teacher model to a more compact student model that requires fewer computations [182]. Classification tasks involving teacher and student networks with similar topologies have been the main focus of prior KD research. Only recently has there been interest in applying KD to regression tasks and cross-architecture KD [183], as suggested in this work. This work uses KD as a structural conversion tool, allowing parallel processing and reducing inference latency, in contrast to the conventional use of KD for complexity reduction. The difficulties in implementing biLSTM in low-complexity hardware for

high-speed processing are addressed by this parallelizable structure. To further lower the computational complexity per recovered symbol, the NN-based equalizers in this work also recover multi-symbol outputs [184].

The key contributions of this chapter are as follows:

- A novel KD-based framework, as shown in Fig. 4.1, to allow parallelization of RNN-based equalizers, by transforming a biLSTM-based teacher model coupled with a 1D-CNN, proposed in [16], into a feedforward student model based on 1D-CNN,
- Comparative studies of performance in Q-factor, computational complexity, and inference latency across various architectures of the NN, namely, biLSTM, biRNN, 1D-CNN, and MLP.
- Validation of the KD framework across diverse transmission scenarios using both simulated and experimental data.

This chapter includes: first, Section 4.2 which explains the concept and benefits of parallelization of NN architecture and symbol recovery. After that, in Section 4.3,

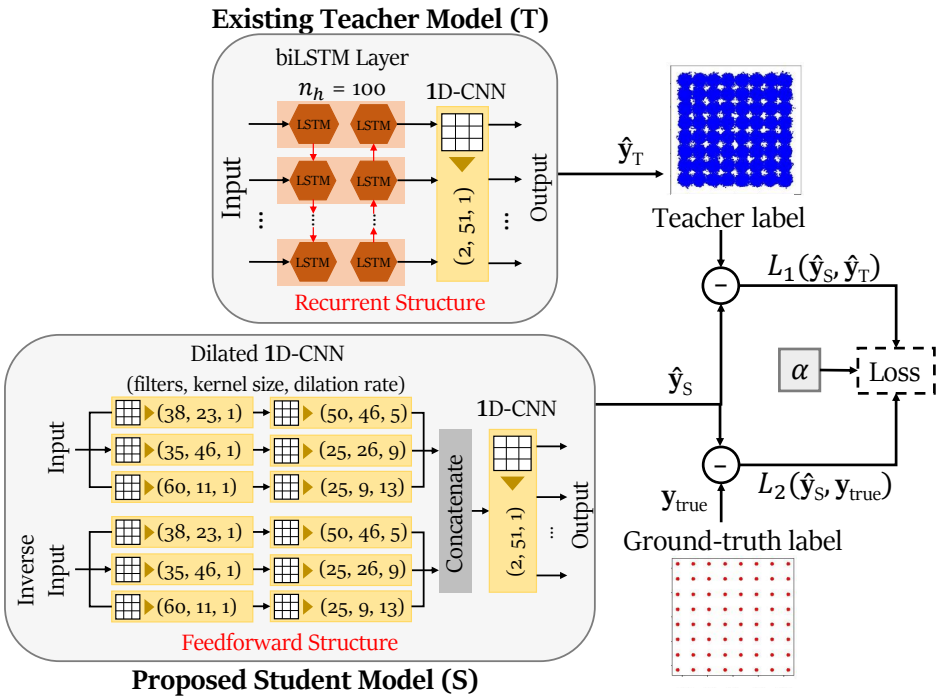


Fig. 4.1 KD framework with biLSTM+1D-CNN as a teacher model and dilated 1D-CNN as a proposed student model.

the KD framework and its background were introduced. Section 4.4 describes the experimental and numerical setups, followed by the training methodology. Next, Section 4.6 presents a comparative analysis of performance and computational trade-offs for the proposed model and other NN architectures across various transmission scenarios. Finally, Section 4.7 provides the conclusion of the key results and discusses the future work.

4.2 Parallelization

4.2.1 Parallelization of NN Architectures

Parallelization of the NN structure should be considered when designing the model. Parallelization can more efficiently leverage hardware usages like multi-core GPUs and CPUs, resulting in the acceleration of the training or inference of the NN. Each NN type offers a different degree of parallelism, depending on its unique architecture. For instance, the structure of NN can have a feedforward nature like CNNs or a recursive one like RNNs. Fig. 4.2 illustrates the recurrent structure at the top with a feedback loop, preventing parallelization, while the feedforward structure at the bottom can process multiple sets of inputs and provide multiple outputs simultaneously.

This work centers on 1D-CNNs. These networks are based on a feedforward structure, allowing input temporal sequences to be processed independently, which makes parallel operations feasible [185, 186]. In the context of signal processing, convolutional layers share similarities with finite impulse response (FIR) filters, as both rely on convolution operations. Specifically, the output at a given time step is determined solely by the current and preceding inputs. The mathematical formulation of a 1D-convolutional layer is as follows:

$$y_i^f = \phi\left(\sum_{n=1}^{n_i}\sum_{j=1}^{n_k} x_{i+j-1,n}^{in} \cdot k_{j,n}^f + b^f\right), \tag{4.1}$$

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. For the FIR filter, the equation can be summarized as:

$$y(n) = \sum_{i=0}^{N} b_i \cdot x(n-i), \tag{4.2}$$

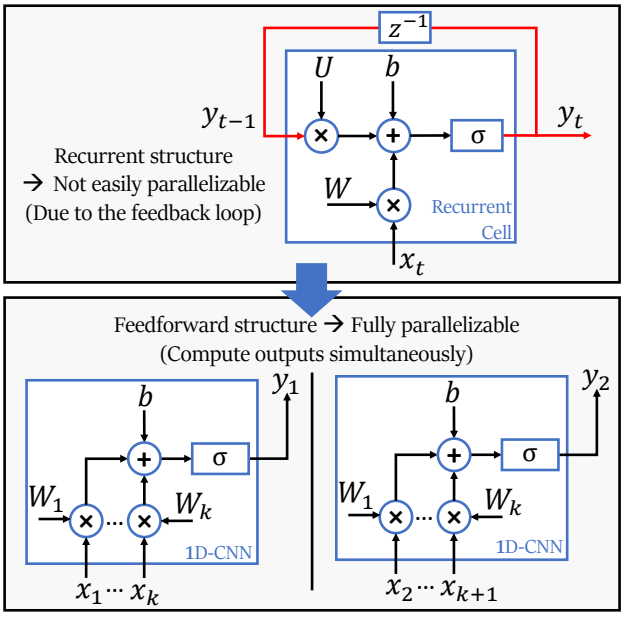


Fig. 4.2 Illustration of the parallelizability comparison between a recurrent cell (top) that is not easily parallelizable due to the feedback loop and a 1D-CNN (bottom) that can process output simultaneously.

where $y(n)$ represents the output signal, $x(n)$ is the input signal, N denotes the filter order. The FIR filter computation is fully parallelizable as it does not contain any recursive part [187–189].

For RNNs, they are particularly effective in modeling sequential data. The output at the current time step y_t depends on the current stage input x_t and the output of the previous stage y_{t-1}. The equation of the RNN for a given time step t is as follows:

$$h_t = \phi(Wx_t + Uh_{t-1} + b), \tag{4.3}$$

where ϕ is 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. Computing the output at the current stage as a function of the previous stage output introduces a recursive evaluation, which inherently requires sequential processing and prevents parallel execution. In signal processing, RNNs can be analogized to infinite impulse response (IIR) filters [190, 191]. This similarity becomes evident when comparing the equations Eq. (4.3) and Eq. (4.4). The equation of the first-order IIR filter is:

$$y(n) = bx(n) + ay(n-1), \tag{4.4}$$

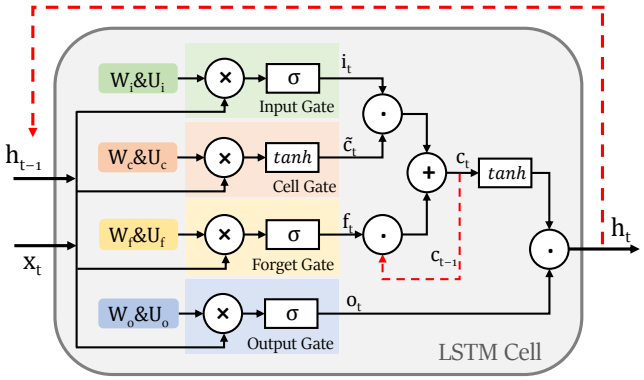


Fig. 4.3 Architecture of an LSTM cell.

where $y(n)$ refers to the output signal, $x(n)$ is the input signal, b is the feedforward filter coefficient and a denotes the feedback filter coefficient. Although Ref. [192] demonstrates the potential for unfolding the pipelined processing of IIR filters into partially parallel operations, the feedback loop remains, as illustrated in Fig. 8 of [192]. Therefore, RNN-based architectures are not entirely parallelizable. This limitation extends to other RNN variants, including LSTM networks, biLSTM networks [71], and Gated Recurrent Unit (GRU) networks.

To be more specific, this study focuses on the biLSTM model architecture. biLSTM is the network consisting of two separate LSTM layers: for forward and backward directions [193]. Because of a double recurrent setting, which cannot be fully parallelized, biLSTM is even more computationally expensive than LSTM and RNN. The architecture of an LSTM cell can be seen in Fig. 4.3. LSTM (see Eq. (2.23)) and RNN (see Eq. (2.22)) have essentially identical core properties: sequential processing and retaining past information through past hidden states. Due to the sequential processing of the recurrent setting, the model computation is very expensive due to limited parallelization [186].

This study proposed to transform the model architecture from the biLSTM to 1D-convolutional layers to enable parallel computation. Parallel computing increases the energy efficiency of the resources and reduces the time-to-solution. More importantly, parallelization allows the NN-based equalizer to be closer to the real hardware implementation. Especially, the optical networks require high-speed data transfer and the latency can be a crucial factor of the equalizers.

4.2.2 Parallelization of Recovered Symbols

Traditionally, NN-based equalizers in earlier studies [62, 194, 66] were designed to recover one symbol at a time, which means that the output of the NN model represents only one recovered symbol at each inference step. However, the single-symbol output NN-based equalizers can be computationally inefficient, as the weights and biases trained to recover one symbol may still be useful for recovering multiple symbols [195]. When taking into account the pre- and post-cursor ISI and chromatic dispersion, the input window should be wider than the output window [122]. The initial and final input symbols in the window lack the information of their neighbors, resulting in a smaller number of recovered output symbols [16]. Multi-symbol output equalizers draw attention to the research areas, previously proposed by [195, 81, 16].

In this study, the proposed NN-based equalizer adopts a multi-symbol output design to reduce computational complexity per recovered symbol. The shape of the input and the output is illustrated in Fig. 4.4a and 4.4b, respectively. The last layer of the models in this work (except for the MLP) adopts the 1D-convolutional layer as in Ref. [16]. The 1D-convolutional layer contains two filters to recover both real (I) and imaginary (Q) components of X polarization of the output signal. The size of the output window or the number of recovered symbols at each inference step is $M - n_k + 1$, where M represents the input window size that NN processes at a time, and n_k is the kernel size in the 1D-convolutional layer. In this study, the padding is set to zero, while the dilation and stride are set to one. The output size can be different depending on the padding,

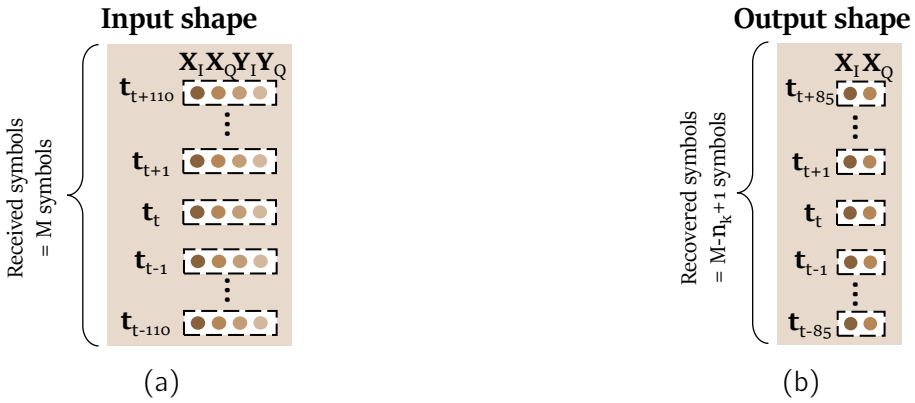


Fig. 4.4 (a) The input of the model contains M real (I) and imaginary (Q) components of both X and Y polarization the received symbols; (b) the output of the model recovers I and Q components of X polarization of $M - n_k + 1$ symbols at the output.

dilation, and stride. Note that the MLP model does not deploy a 1D-convolutional layer but it uses the neurons to recover multi-symbol output instead.

4.3 Knowledge Distillation

Generally, KD refers to a model compression technique of transferring knowledge from a complex model, known as a teacher model, to a more compact one, known as a student model, which is less computationally expensive to evaluate [182, 196]. KD allows the student model to have a comparable performance with respect to the teacher while requiring less CC. The student model exploits the teacher's predictions to assist its learning. The predictions from the teacher model referred to as "teacher labels" or "soft labels", are used to train the student model together with the ground truth labels to aid the student's learning. Most of the prior work proposed KD [197, 182, 198, 199] in a classification task. In classification, the ground truth labels are usually one-hot labels. The teacher labels contain useful information about the relative similarity of the incorrect predictions, while the one-hot labels do not provide that sort of information [200, 182, 201, 118]. For example, the teacher labels, which were the results of the Softmax function, contain probabilities of each class in a multi-class problem, whereas the one-hot label only contains one or zero for each class.

KD in a regression task is still in its infancy, but various papers, e.g., [200, 183, 202] have shown that KD can demonstrate promising results in this context. However, it is still ambiguous in some cases about how the student model can take advantage of the teacher's predictions in the regression task [200]. In this work, Fig. 4.1 demonstrated how the teacher labels contain the noise information in the constellation diagram compared to the ground truth labels. KD in regression also performs as an efficient regularizer to improve generalization. Sec. 4.6 shows the weight distribution of the student model trained with KD compared to the one without KD.

The loss paradigm for KD involves a joint loss function that considers both the loss between the teacher's and the student's predictions and the difference between the student's predictions and the ground truth labels, as described in [183]. This dual-component loss function is illustrated in Fig. 4.1 and is expressed as follows:

$$\mathrm{L}_{\mathrm{KD}} = \alpha L(\hat{\boldsymbol{y}}_S, \hat{\boldsymbol{y}}_\mathrm{T}) + (1-\alpha) L(\hat{\boldsymbol{y}}_S, \boldsymbol{y}_{\mathrm{true}}), \tag{4.5}$$

where $\boldsymbol{y}_{\text{true}}$ is the actual labels, $\hat{\boldsymbol{y}}_S$ and $\hat{\boldsymbol{y}}_\text{T}$ represents the student's and the teacher's predictions, respectively and α is the hyper-parameter to adjust the contribution for each term to the final loss. The first term of the loss function enables the student model to learn from the teacher's predictions, while the second term ensures that the student model learns directly from the ground truth. Empirically, this α parameter has a similar impact on the performance as the regularizer coefficient, as it can impact how overfitting the model is, depending on how much the student model learns from the teacher or from the ground truth. In this study, the L function is the L_2 distance (Euclidean distance). It is worth noting other functions, such as MSE, can also be adopted, but in this case, the L_2 distance provides better learning.

Typically, KD is applied when the teacher and student models share similar network topologies, with the primary objective of reducing computational complexity [118]. However, cross-architecture KD, where the teacher and student models have distinct structures, has seen limited investigation [183]. In this work, KD is leveraged to restructure the NN architecture in regression tasks. Unlike traditional KD applications that aim to reduce computational complexity in terms of RMs, this approach focuses on minimizing inference time to enhance practical deployment.

4.4 Data Generation and NN Training

4.4.1 Data Generation

The numerical simulator created the dataset by assuming the transmission of a single-channel 30 GBd, 64-QAM DP channel along 20×50 km SSMF spans. The signal propagation through the fiber was represented by a generalized Manakov equation SSFM [14]. The SSMF is characterized by the effective nonlinearity coefficient $\gamma = 1.2$ $(\text{W}\cdot \text{km})^{-1}$, chromatic dispersion coefficient $D = 16.8$ ps/(nm·km), and attenuation parameter $\alpha = 0.21$ dB/km. At the end of each fiber span, optical fiber losses are compensated for by an EDFA with a 4.5 dB noise figure. Downsampling and CDC were performed on the receiver end. The CDC was performed in the frequency domain with the transfer function of the chromatic dispersion given by [14]: $G(z,\omega) = \exp\left(-\frac{j\omega^2\beta_2 z}{2}\right)$ where ω is the angular frequency, β_2 is the group delay dispersion parameter of the fiber and z is the transmission length. Afterward, the received symbols were normalized and used as inputs of the NN.

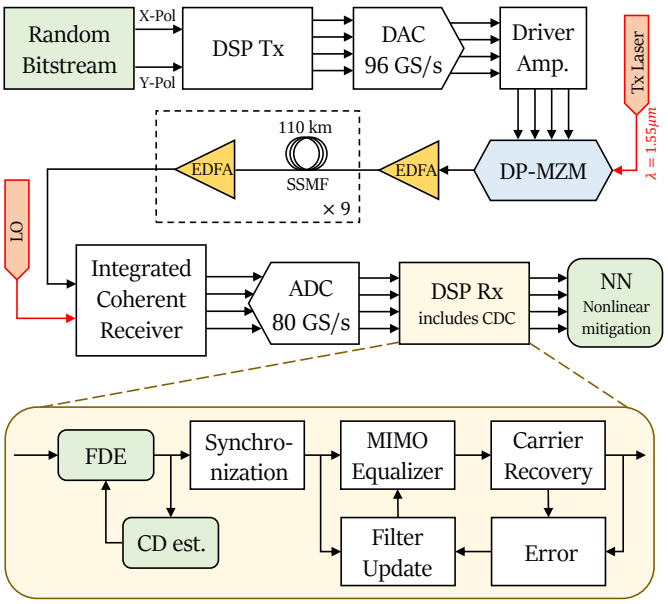


Fig. 4.5 Experimental setup where the data after the DSP Rx is fed into the NN as the input.

In this work, the experimental data were also analyzed to verify the performance of the proposed KD framework. The transmission scenario was single-channel DP-probabilistic shaped (PS)-64QAM[1] in 34.4 GBd along 9×110 km SSMF fiber spans. The experimental setup in Fig. 4.5 was detailed in Ref. [16]. At the transmitter side, a symbol sequence with a modulation scheme of 64QAM (8 bits/4D symbol) with a symbol rate of 34.4 GBd was mapped out of data bits generated by a Marsenner twister generator [203]. After that, the symbol sequence was passed through a digital RRC filter with a 0.1 roll-off factor to limit the channel bandwidth to 37.5 GHz. The filtered digital samples were resampled and fed into a DAC operating at 96 GSamples/s. The DAC outputs were then amplified by a four-channel electrical amplifier which drove a DP in-phase/quadrature MZM. The modulator modulated a continuous waveform carrier generated by an external cavity laser at a wavelength of $\lambda = 1.55\mu$m. The resulting optical signal was transmitted along 9×110 km spans of SSMF with lumped EDFA amplification with the noise figure ranging from 4.5 to 5 dB. The SSMF had α = 0.21 dB/km, D = 16.8 ps/(nm·km), and γ = 1.14 (W· km)$^{-1}$.

At the receiver side, the received optical signal was converted to the electrical domain using an integrated coherent receiver. Then, the resulting signal was sampled at

[1]To demonstrate that the NN equalizer is applicable in different transmission scenarios, the experimental data of the PS case was utilized.

80 GSamples/s using a digital sampling oscilloscope. The sampled signal was processed offline with the DSP algorithms described in [204]. The CDC was accomplished in two steps. The first step of processing involved compensating for bulk accumulated dispersion using a FDE with an FFT size of 12288 samples and an impulse response length of 3072 samples[2]. After that, the residual chromatic dispersion and dynamic impairments of channels were mitigated by the adaptive approach, MIMO equalization with an FFT size of 192 samples and an overlap size of 48 samples. Next, the carrier frequency offset was mitigated. A constant-amplitude zero autocorrelation-based training sequence was located in the received frame, and the equalizer transfer function was estimated from it. Then, the two polarizations of the signal were demultiplexed, and clock frequency and phase offsets were corrected. The carrier phase estimation was performed using pilot symbols. The resulting soft symbols were used as input for an NN equalizer. Finally, the pre-FEC BER was evaluated based on the signal obtained at the output of the NN equalizer. In this case, the NN focused on mitigating the nonlinear effects, and was not designed to replace the regular DSP, instead, the NN was applied as an extra step to the regular DSP.

4.4.2 KD Training to Solve the Parallelization Problem of Recurrent Connection

The KD framework is deployed to transform the model architecture from the biLSTM+CNN to simpler ones. Fig. 4.1 shows the KD process with the teacher and student model structure. The constellation diagrams of the teacher labels and the ground-truth labels in Fig. 4.1 indicate that the training of the student model with the teacher's predictions that contained the information on noise and some uncertainty, can result in not overly confident predictions of the student. This helps reduce overfitting and improve the generalizability of the student model. The comparison of different types of student models was carried out, namely, biRNN, 1D-CNN, and MLP. 1D-CNN and MLP have a feedforward structure, enabling parallel computation for the previously proposed biLSTM-based equalizer [16] or the teacher model. In contrast, biRNN still has a recurrent structure. However, biRNN architecture is considerably less complex than the biLSTM, thus allowing for faster computation. In this work, the biRNN is limited to only one layer in order to maintain the complexity.

[2]These values allow to compensate for up to an accumulated dispersion of 300 nm/km, which is significantly more than the one needed to compensate in a link with 9×110 km.

The teacher model is pre-trained and used only to create teacher labels. For both simulation and experiment, the training datasets were generated with random bitstream of 2^{20} symbols, however, at every epoch, 2^{18} symbols were randomly chosen from this dataset as input symbols to train the model. For testing and validation, the dataset contained unseen 2^{17} symbols. All the models in this work were trained, validated, and tested with this same size of dataset. The training of the models in both simulation and experiment was carried out for 1000 epochs. The TL is utilized to transfer the knowledge from the models trained with higher power to the model with lower power to save the training resources. A mini-batch size is 2^{10} and the mini-batch input of the NN is defined in three dimensions [62]: (B, M, 4). B is the mini-batch size. M is the memory size depending on the number of neighbor symbols N as $M = 2N+1$. The last dimension has the shape of four referring to the number of features for each symbol. Both the teacher and student models accepted four input features resulting from the in-phase and quadrature components of the complex signal (X_I, X_Q, Y_I, and Y_Q) where $X_I + jX_Q$ and $Y_I + jY_Q$ were the signals in the X and Y polarizations, respectively. The output is to recover the real and imaginary parts of multiple symbols in X polarization simultaneously. The shape of the NN output batch can be expressed as (B, $M - n_k + 1$, 2), where $M - n_k + 1$ is the number of symbols recovered at the output. The weights of the trained models were saved at the epoch where the BER of the validated dataset was the lowest, as known as the early stopping method.

To evaluate the effectiveness of the KD framework, the training of the 1D-CNN student model with KD is compared to the traditional training approach without knowledge of the teacher model, known as the student model trained from scratch. In addition, the model with the L2 regularizer [205] was also investigated to improve the generalizability of the student model trained from scratch. The student model trained from scratch has the same structure and hyper-parameters as the proposed student trained with KD.

Teacher Network Architecture

The teacher model is a biLSTM+CNN model, see Fig. 4.1, which was trained previously in [16]. The biLSTM layer has 100 hidden units (n_h), and the 1D-CNN layer adopts 2 filters (n_f) and n_k = 51 with the linear activation function. The loss function used in this model is MSE, and the optimizer is Adam with a learning rate of 10^{-3}. The input window (M) has the size of 221 input symbols and the model performs a regression task to predict the real and imaginary parts of the recovered symbols, or 171 symbols

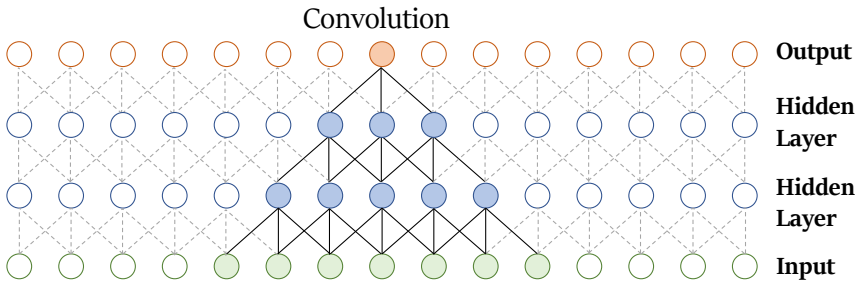
per one inference step. For the teacher model employed in the experimental setup, the model has 117 hidden units and the output window is 195 symbols. More details can be found in Ref. [16].

Student Network Architecture

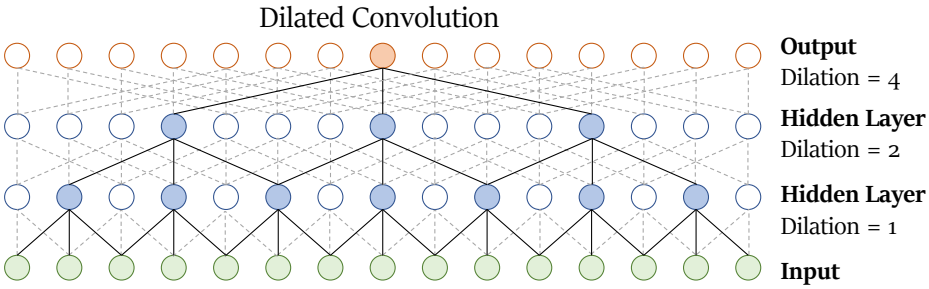
This study investigated various types of student models—1D-CNN, biRNN, and MLP[3] —to evaluate their equalization performance, computational complexity and inference speed. For both simulation and experimental setups, the student models were trained using the Euclidean distance (L_2 distance) as the loss function.

For the proposed 1D-CNN model, the dilated CNN is applied. The dilated convolution is an approach to inflate the kernel by inserting holes between its consecutive elements.

[3]Note that the hyper-paraments of all types of student models for both simulated and experimental data were optimized with the dataset when the launch power was 2 dBm which was the optimum launch power of the teacher model and these values were used for other launch power. The optimization for each type of student model has undergone approximately the same amount of time.



(a) Normal CNN with dilation equal 0.

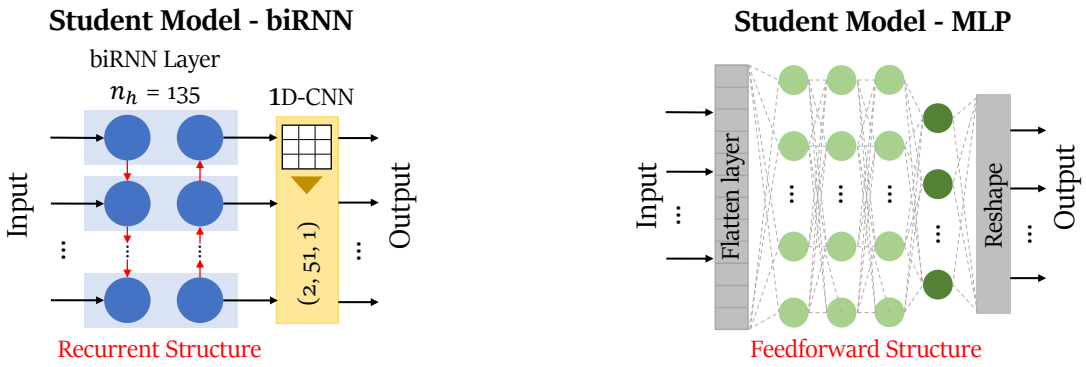


(b) Dilated CNN.

Fig. 4.6 Visualization of (a) a stack of convolutional layers; (b) a stack of 'dilated' convolutional layers.

Consequently, the network is operated on a coarser scale than with a normal convolution filter with a dilation rate equal to zero [206–208]. This approach allows the NN to deal with long-term temporal dependencies, and have larger receptive fields within only a few layers as shown in Fig. 4.6b, compared to a normal CNN in Fig. 4.6a. The dilated CNN preserves the input resolution throughout the network [209]. The dilated convolutions demonstrated the longer receptive field in a cheaper way than the LSTM [206]. In Ref. [183], the KD student model with dilated CNN architecture shows promising results when the teacher model is the LSTM architecture and the data is in the form of a time-series in the regression task. To mimic biLSTM, which learns the input data in forward and backward directions, the 1D-CNN student model learns both directions of the training data. The backward direction input means the input sequence (forward direction) in reverse time order. The last 1D-CNN layer of both the teacher and the student has the same parameters. The Bayesian optimizer [62] is used to optimize the hyper-parameter values of the student model. The estimated optimal values found by the Bayesian optimizer are depicted in Fig. 4.1. Note that (38, 23, 1) means that the 1D-CNN layer operates with 38 filters, a kernel size of 23, and a dilation rate of 1. The alpha value is 0.903. The activation function of the dilated 1D-CNN part is LeakyRelu [210]. Note that the architectures and parameters of the student models in the simulation and experimental setup are the same, apart from the second layer of the 1D-CNN, instead of 25 filters, it has 33 and 34 filters to maintain the output dimensions.

The vanilla RNN is the simplest variant of the recurrent-based models [211]. This RNN student model adopts one layer of biRNN with 135 hidden units followed by a



(a) (b)

Fig. 4.7 Different student architectures: (a) biRNN model as a student model; (b) MLP model as a student model.

1D-CNN layer with 2 filters and a kernel size of 51 as the teacher model, see Fig. 4.7a. Stacking the biRNN layers is avoided to maintain the inference speed, therefore, the performance can be limited. The training was carried out with the alpha value of 0.611, optimized by Bayesian optimizer.

The last student model type is the MLP, shown in Fig. 4.7b, consisting of three hidden layers with the hidden units of 401, 510 and 510, respectively, and an output layer of 342 neurons. The output layer is reshaped to match the aforementioned output window shape of the data. The neurons in the hidden layers have hyperbolic tangent (tanh) as an activation function, while the output layer deploys a linear function. The alpha value is 0.8.

4.5 Computational Complexity Evaluation Metrics

This section presents the formulas of the complexity metrics (RM, BOP, and NABS) explained in Chapter 2, to evaluate the complexity of the NN models in this chapter. The detailed explanation of each equation was discussed in [55].

Dense Layer

Starting with the MLP model explained in Section 2.2.1, an MLP model consists of multiple dense layers. RM of a dense layer can be calculated as:

$$\mathrm{RM}_{\mathrm{Dense}} = n_n n_i, \tag{4.6}$$

where n_i is the number of features in the input vector and n_n represents the number of neurons in the layer. After that, we move on to calculating BOP to take into account the bitwidth or the precision. BOP of a dense layer includes the costs of both multiplications and additions. $\mathrm{BOP}_{\mathrm{Mul}}$ corresponds to the BOP of vector-matrix multiplication and $\mathrm{BOP}_{\mathrm{Bias}}$ represents bias addition [55]:

$$\mathrm{BOP}_{\mathrm{Mul}} = n_n\left[n_i b_w b_i + (n_i - 1)(b_w + b_i + \lceil \log_2(n_i) \rceil)\right], \tag{4.7}$$

$$\mathrm{BOP}_{\mathrm{Bias}} \approx n_n(b_w + b_i + \lceil \log_2(n_i) \rceil), \tag{4.8}$$

where b_w is the weight bitwidth and b_i is the input bitwidth. For the convenience of the upcoming expressions, short notations are defined:

$$\text{Mult}(n_i, b_w, b_i) = n_i b_w b_i + (n_i - 1)\big(b_w + b_i + \lceil \log_2(n_i) \rceil\big), \tag{4.9}$$

$$\text{Acc}(n_i, b_w, b_i) = b_w + b_i + \lceil \log_2(n_i) \rceil. \tag{4.10}$$

The Acc term represents the actual bitwidth of the accumulator required for MAC operation. The BOP of a dense layer can be calculated as:

$$\begin{aligned} \text{BOP}_{\text{Dense}} &= \text{BOP}_{\text{Mul}} + \text{BOP}_{\text{Bias}} \\ &\approx n_n n_i \big[b_w b_i + (b_w + b_i + \lceil \log_2(n_i) \rceil) \big] \\ &\approx n_n n_i \big[b_w b_i + \text{Acc}(n_i, b_w, b_i) \big]. \end{aligned} \tag{4.11}$$

For the last metric, NABS of the dense layer can be formulated as [55]:

$$\begin{aligned} \text{NABS}_{\text{Dense}} &\approx n_n n_i \big[X_w \text{Acc}(n_i, b_w, b_i) + \text{Acc}(n_i, b_w, b_i) \big] \\ &\approx n_n n_i (X_w + 1) \text{Acc}(n_i, b_w, b_i), \end{aligned} \tag{4.12}$$

where X_w represents the number of adders required at most to perform the multiplication. To be more specific, for the uniform quantization, we have $X_w = b_w - 1$. This is because in the binary system, multiplying can be represented by a shift and adders, and the number of adders needed at most is $b_w - 1$. In this chapter, only the uniform quantization is assumed.

1D convolutional layer

Next, for the 1D convolutional layer as formulated in Eq. (2.20) and explained in Section 2.2.1, the output size of the 1D-convolutional layer is:

$$\text{OutputSize} = \left\lfloor \frac{n_s + 2\,padding - dilation(n_k - 1) - 1}{stride} + 1 \right\rfloor, \tag{4.13}$$

where n_s is the input time sequence size and n_k is the kernel size. The RM of a 1D-convolutional layer can be calculated as follows:

$$\text{RM}_{\text{CNN}} = n_f n_i n_k \cdot \text{OutputSize}, \tag{4.14}$$

where n_f is the number of filters or the output dimension. The BOP for a 1D-convolutional layer, after considering the multiplications and additions, can be defined as [55]:

$$\begin{aligned} \mathrm{BOP}_{\mathrm{CNN}} = \mathrm{OutputSize} \cdot n_f \mathrm{Mult}(n_i n_k, b_w, b_i) \\ + n_f \mathrm{Acc}(n_i n_k, b_w, b_i). \end{aligned} \tag{4.15}$$

The NABS of a 1D-convolutional layer is given by [55]:

$$\begin{aligned} \mathrm{NABS}_{\mathrm{CNN}} = \mathrm{OutputSize} \cdot n_f \big[n_i n_k (X_w + 1) - 1 \big] \\ \cdot \mathrm{Acc}(n_i n_k, b_w, b_i) \\ + n_f \mathrm{Acc}(n_i n_k, b_w, b_i). \end{aligned} \tag{4.16}$$

Vanilla Recurrent Neural Network

The equation of Vanilla RNN was defined and explained in Section 2.2.1. The RM of a vanilla RNN is given by:

$$\mathrm{RM}_{\mathrm{RNN}} = n_s n_h (n_i + n_h), \tag{4.17}$$

where n_h notes the number of hidden units. The BOP for a vanilla RNN is presented as [55]:

$$\begin{aligned} \mathrm{BOP}_{\mathrm{RNN}} = n_s n_h \mathrm{Mult}(n_i, b_w, b_i) \\ + n_s n_h \mathrm{Mult}(n_h, b_w, b_a) \\ + 2 n_s n_h \mathrm{Acc}(n_h, b_w, b_a), \end{aligned} \tag{4.18}$$

where b_a is the activation function bitwidth. Like other network types, the NABS of vanilla RNN can be computed from its BOP equation by transforming the multiplication to the number of adders needed at most (X) [55]:

$$\begin{aligned} \mathrm{NABS}_{\mathrm{RNN}} = n_s n_h \big[n_i (X_w + 1) - 1 \big] \mathrm{Acc}(n_i, b_w, b_i) \\ + n_s n_h \big[n_h (X_w + 1) + 1 \big] \mathrm{Acc}(n_h, b_w, b_a). \end{aligned} \tag{4.19}$$

Long Short-Term Memory

The LSTM layer and its formulas were discussed in Section 2.2.1. The RM of an LSTM layer is:

$$\mathrm{RM}_{\mathrm{LSTM}} = n_s n_h (4 n_i + 4 n_h + 3), \tag{4.20}$$

where n_h is the number of hidden units in the LSTM cell. The BOP for an LSTM layer includes the bitwidth of the operands and the number of additions. Consequently, the BOP can be defined as [55]:

$$\begin{aligned} \text{BOP}_{\text{LSTM}} = & 4n_s n_h \text{Mult}(n_i, b_w, b_i) \\ & +4n_s n_h \text{Mult}(n_h, b_w, b_a) \\ & +3n_s n_h b_a^2 \\ & +9n_s n_h \text{Acc}(n_h, b_w, b_a). \end{aligned} \tag{4.21}$$

Lastly, the NABS of LSTM layer is described as [55]:

$$\begin{aligned} \text{NABS}_{\text{LSTM}} = & 4n_s n_h \big[n_i(X_w+1)-1\big]\text{Acc}(n_i, b_w, b_i) \\ & +4n_s n_h \big[n_h(X_w+1)+1\big]\text{Acc}(n_h, b_w, b_a) \\ & +6n_s n_h b_a. \end{aligned} \tag{4.22}$$

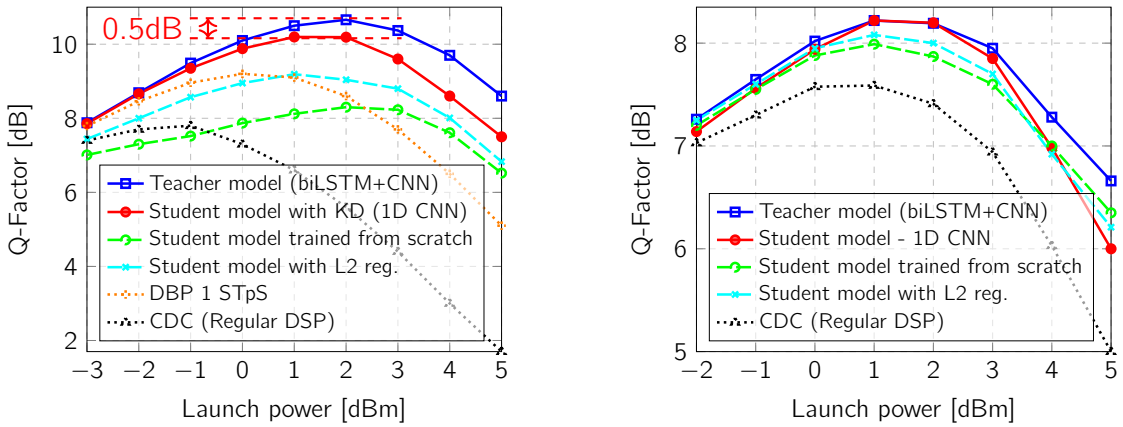
Please note that when computing the complexity for the bi-directional layer, such as bi-RNN, the complexity of a bidirectional layer is twice the unidirectional layer.

4.6 Results and Discussion

4.6.1 Equalization Performance

The proposed student model (1D-CNN) trained using the KD framework is compared against the teacher model (biLSTM+CNN), the student model trained from scratch without KD (using the same settings), and the student model trained from scratch with an L2 regularizer [205]. The optimum L2 coefficient depends on the launch power. At 2 dBm launch power, the optimum L2 coefficient found by grid search is 10^{-4} for simulated data and 5×10^{-6} for experimental data. Fig. 4.8a depicts Q-factor vs. launch power for different types of NN-based equalizers in the simulated data. In all NN-based equalizers, an improvement of the optimum launch power was achieved. It can be observed that the Q-factor performance of the feedforward student model with KD drops by 0.5 dB compared to the recurrent teacher model at its optimal launch power (2 dBm). With KD, the performance of the student model is comparable to that of the teacher model in the linear transmission regime, but the student's performance degrades slightly as the launch power increases. However, when training the student model from scratch without KD, the model suffers from overfitting, resulting in a noticeable

degradation of the peak performance by 2.4 dB at its optimal power. Training the student model with the L2 regularizer, which helps enhance the generalization capability, improves the performance of the NN compared to training it from scratch only, but still does not reach a similar performance level as the one achieved when the student model is trained with KD. The performance achieved using DBP 1 STpS and CDC are also shown for reference.



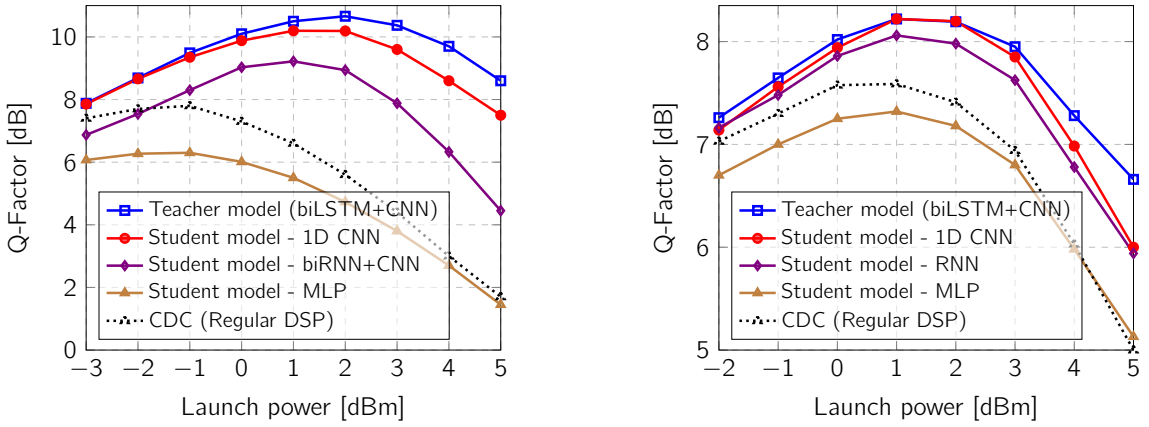
(a) SC-DP 30GBd; 64QAM; 20×50km SSMF link (Simulation).

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

Fig. 4.8 Q-factor as a function of the launch power for the NN-based equalizers obtained via KD, compared to the original (teacher) model, CDC in different transmission scenarios.

With the experimental setup, the optimal performance of the student model with KD, shown in Fig. 4.8b, was comparable to the teacher model. The performance drop was not observed in the experimental data. However, the student model trained from scratch did not suffer from severe overfitting as in the simulated data, but it still could not reach the same Q-factor level as the teacher model or the student model trained with KD. In the case that the student model was trained from scratch with an L2 regularizer, the performance was improved slightly. When the model is not significantly overfitting, the L2 regularizer parameter needs to be carefully selected. In this experimental setup, the weaker regularization parameter was preferred (5×10^{-6}), to prevent the regularizer from excessively penalizing the weights, allowing the model to still learn meaningful patterns from the data [212]. This result demonstrated that the proposed student network trained with KD was highly effective in the experimental data and the KD also maintained superior performance compared to the student models trained from scratch. However, it can be observed that at the higher launch power, the

performance gap was larger. At the launch power of 5 dBm, the model with KD and the student model with L2 regularize performed worse than the student model trained from scratch. This can be because the optimization process was carried out with the dataset with the launch power of 2 dBm, resulting in a sub-optimal performance at 5 dBm and overly constrained weight distribution.



(a) Simulation. (b) Experiment.

Fig. 4.9 Comparison of different architectures (biRNN+CNN, 1D-CNN and MLP) of the student model to the teacher model, CDC and 1 STpS DBP in: (a) simulation and (b) experiment.

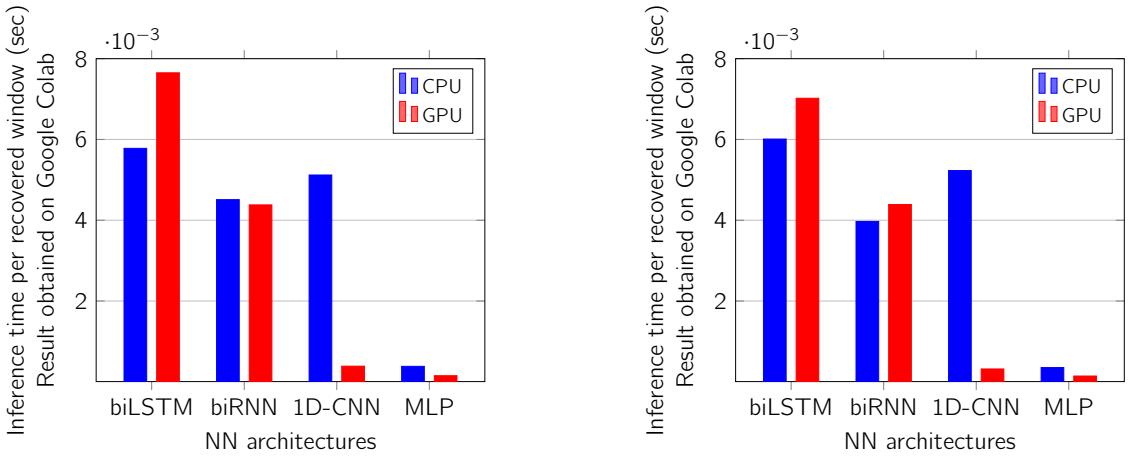
To demonstrate the effectiveness of the proposed 1D-CNN student model, the 1D-CNN is compared against the biRNN and MLP as student models, shown in Fig. 4.9a and Fig. 4.9b for the simulated and experimental data, respectively. The performance of the biRNN and MLP students was not comparable to the 1D-CNN model. In the simulation, the biRNN model had around 1.4 dB Q-factor drop from the teacher model but still outperformed the CDC and showed the improvement of the optimum launch power. In the experiment, the biRNN revealed the same behavior as in the simulation. Even the biRNN has a recurrent structure, but as the implementation was limited to only one layer to limit the inference latency, the model did not learn well. In addition, instability during the training was present, which can result from vanishing/exploding gradients. In the case of the MLP as a student model[4], the performance was poorer than that of the CDC in both simulation and experiment. The MLP was not the most suitable architecture for nonlinear mitigation when considering the performance [89],

[4]Note that during the optimization process, attempts were made to increase the number of layers to determine if this would enhance performance; however, the MLP reached a point where adding more hidden layers no longer improved performance.

especially for recovering the multi-symbol output as in this case. This can be explained by considering that the MLP lacks temporal information handling by design. The MLP does not take the sequence of data points into account, which limits its capability to effectively model time series dynamics.

4.6.2 Inference Speed Performance

For this analysis, the inference speed was tested with the simulation data and the experimental data. Note that both the teacher and the student models recover 171 symbols per inference step in the simulation, and 195 symbols per inference step in the experiment. The inference time analysis was carried out by using CPU (Intel Xeon Processor 2.20 GHz) and GPU (Tesla T4) on Google Colab [213][5] as the inference engine. In this analysis, the size of the test set was 800, and the batch size was 8. As can be seen from Fig. 4.10a for the simulation and Fig. 4.10b for the experiment, the biLSTM-based NNs require the longest inference time in both CPU and GPU as inference engines.



(a) Simulation.

(b) Experiment.

Fig. 4.10 Inference time of teacher and different student models when applied testing data obtained from (a) simulation; (b) experiment.

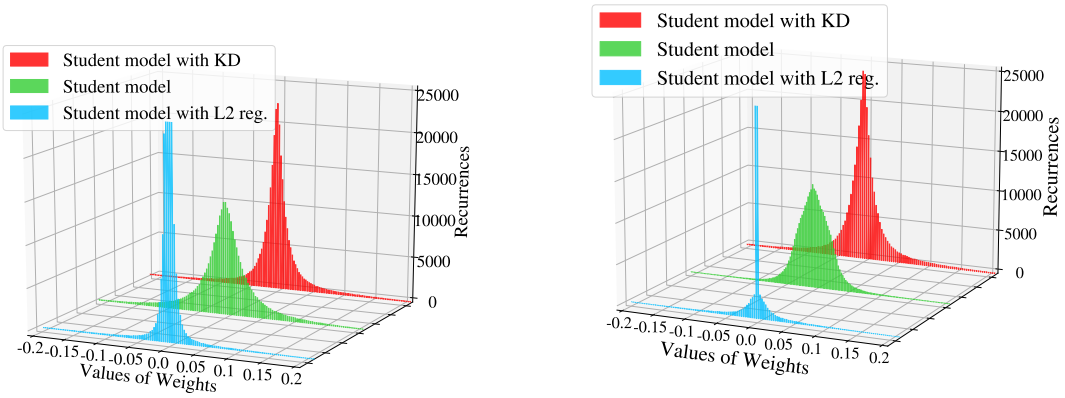
The biRNN presented lower inference latency than the biLSTM due to its simpler architecture. The inference time of the recurrent-based NN in the CPU and the GPU did not differ significantly. In contrast, the feedforward NNs (1D-CNN and MLP) have

[5]Note that, in this study, the GPU-accelerated library of primitives for deep NNs cuDNN (NVIDIA CUDA® Deep NN library) was not considered for the inference in the GPU.

significantly lower inference latency in the GPU than that of the CPU, especially the 1D-CNN model. 1D-CNN model when the CPU was used as the inference engine experienced similar latency as the recurrent-based NN. However, with the feedforward nature of the 1D-CNN that allows parallelization, the GPU can demonstrate remarkably faster computation. The MLP's inference time is shorter than the 1D-CNN as it has a simpler structure and operations. The recurrent networks (biLSTM and biRNN) experienced longer inference latency in the GPU compared to the CPU. This can happen due to the non-easily parallelizable recurrent structure and the complexity of the model. GPUs are specialized for efficient parallel computing, handling complex models with multiple layers and parameters. The GPUs are generally faster when the computation is parallelizable and involves matrix multiplications, while in some other types of computations, the GPU can be slower. The parallel computing ability of the GPU can fully exploit the parallelizability of the feedforward structures of 1D-CNN and MLP. Overall, the proposed 1D-CNN student model provided the most reasonable trade-off between performance and inference speed. The parallelization of the proposed feedforward equalizer and its savings in latency are key to the real-time hardware implementation of NN-based equalizers.

4.6.3 Roles of Knowledge Distillation

This subsection discusses the features associated with the KD-trained model. For this purpose, the weight distribution of the student model trained with different approaches in Fig. 4.11a for the simulation and Fig. 4.11b for the experiment are reported. In both figures, compared to the student model trained from scratch, the student model with KD has a more regularized weight distribution: the weights are more concentrated around zero. This characteristic helps reduce the model's variance and overfitting. The optimal value of α in the KD loss function is 0.903, which means that the student model learns 90.3% from the teacher labels and the rest comes from the ground-truth labels. This fact demonstrates the effectiveness of the teacher labels in the student's learning. The teacher constellation/labels depicted in Fig. 4.1 show that the teacher also provides helpful information on the noise, whereas this information cannot be encoded in the ground-truth labels (which contain only real values). The weight distribution of the student model with KD and the improvement in Q-factor, compared to the training of the 1D-CNN without KD, both support the concept of using teacher labels as efficient regularizers [182].



(a) Simulation. (b) Experiment.

Fig. 4.11 Comparison of weight distributions of the student model trained with KD, the student model trained from scratch, and the student model trained with L2 regularizer, using the data obtained from: (a) simulation; (b) experiment.

KD can work as an efficient regularizer to allow the model to generalize well with unseen data. In this regard, the KD framework provides the adequacy of the NN weight constraints, in contrast, too strict or too weak L2 regularizer parameters, may not provide significant benefits. However, the KD framework also shows some limitations. The training complexity is increased because both the teacher and the student models need to be trained. For example, when the transmission scenario changes, the teacher model needs to be trained first before the student one with KD can be trained effectively. This can be time and resource-consuming. During the training, KD involves learning via the teacher's predictions, resulting in a more complex training process. Moreover, the student model relies heavily on the accuracy of the teacher model. When considering the optimization process, to obtain the best performance, both the parameters of the teacher and the student need to be optimized. A teacher with good performance is necessary for the student's learning.

4.6.4 Complexity Comparison of Different NN-Based Equalizers

In this section, we evaluate the performance versus computational complexity of the NN-based equalizers after applying KD framework to train different student models (biRNN+CNN, 1D-CNN, and MLP) with the knowledge of teacher model (biLSTM+CNN). The computational complexity in terms of the number of trainable

parameters, RM, BOPs, NABS, and inference latency is compared. As mentioned earlier, KD in general context is used to reduce the computational complexity in terms of the NN parameters, however, in this work, the KD framework is used to recast the NN structure from biLSTM-based (recurrent-based) to a feedforward-based equalizer to allow parallelization in processing. Therefore, KD was applied to focus on enabling parallelization and reducing the inference latency rather than reducing the RM, BOPs, or NABS.

Data/ Output Shape	Performance Metric	biLSTM+CNN (Teacher)	biRNN+CNN (Student)	1D-CNN (Student)	MLP (Student)
Simulation 171 Symbols	Q-factor	**10.66**	9.22	10.19	6.3
	No. Trainable Parameters	1.04×10^{5}	$\mathbf{6.53}\times10^{4}$	2.93×10^{5}	9.95×10^{5}
	RMpS	1.29×10^{5}	7.60×10^{4}	3.73×10^{5}	$\mathbf{5.81}\times10^{3}$
	BOPpS	1.67×10^{8}	1.14×10^{8}	8.03×10^{8}	$\mathbf{1.25}\times10^{7}$
	NABSpS	3.2×10^{8}	2.10×10^{8}	1.27×10^{9}	$\mathbf{1.96}\times10^{7}$
	CPU Inference Time per Window	5.78×10^{-3}	4.51×10^{-3}	5.12×10^{-3}	$\mathbf{3.83}\times10^{-4}$
	GPU Inference Time per Window	7.65×10^{-3}	4.38×10^{-3}	3.87×10^{-4}	$\mathbf{1.51}\times10^{-4}$
Experiment 195 Symbols	Q-factor	8.22	8.06	**8.22**	7.32
	No. Trainable Parameters	1.27×10^{5}	$\mathbf{5.24}\times10^{4}$	3.10×10^{5}	1.02×10^{6}
	RMpS	1.42×10^{5}	5.71×10^{4}	3.43×10^{5}	$\mathbf{5.22}\times10^{3}$
	BOPpS	1.73×10^{8}	7.93×10^{7}	7.38×10^{8}	$\mathbf{1.12}\times10^{7}$
	NABSpS	3.39×10^{8}	1.50×10^{8}	1.17×10^{9}	$\mathbf{1.76}\times10^{7}$
	CPU Inference Time per Window	6.01×10^{-3}	3.97×10^{-3}	5.23×10^{-3}	$\mathbf{3.53}\times10^{-4}$
	GPU Inference Time per Window	7.02×10^{-3}	4.39×10^{-3}	3.18×10^{-4}	$\mathbf{1.43}\times10^{-4}$

Table 4.1 Summary of the performance versus complexity of different architectures of NN-based equalizers after applying KD (biLSTM+CNN as a Teacher model, biRNN+CNN, 1D-CNN and MLP as Student models), where the bitwidth b_i = 64, b_w = 32, and b_a = 32. The RM, BOP and NABS are reported "per equalized symbol".

Table. 4.1 shows the summary of the performance versus complexity of these two NN architectures when the precision is the default values from Tensorflow. The bitwidth of the input (b_i) is 64 bits, and the bitwidth of the weights (b_w) and activation function (b_a) is 32 bits. It can be seen that the Q-factor of the student model is slightly lower than that of the teacher model when trained with the simulated data mentioned above. Note that in the experiment, this KD approach did not show the performance degradation in the student model. Even though the number of trainable parameters,

RM, BOPs, and NABS of the student model are also higher than the teacher model, the inference latency of the student is actually lower. This highlights the importance of the "parallelization" strategy, which helps reduce the complexity of the processing at the hardware synthesis level. This is crucial for real-world implementation. For the training phase in this case, the trainable parameters can indicate the complexity to some extent; however, empirically, the biLSTM+CNN has a longer training time, even though the number of trainable parameters is lower than the 1D-CNN model. This occurs because the recurrent structure of the biLSTM prevents the computation from being fully parallelizable. At each time step of the calculation, the recurrent structure takes into account the output of the previous time step. This sequential nature makes the training and inference longer.

Fig. 4.12 shows BOPs and NABS as a function of the bitwidth of the weights (b_w) of the NN. It can be observed that the NABS grows with a steeper slope than the BOPs when the b_w increases. This fact highlights that towards the implementation of resource-constrained devices or hardware accelerators, both metrics should be considered carefully, because if only BOPs is assessed at higher precision of the weights, while BOPs fits the requirement, NABS which escalates faster might exceed the requirement of the implementation.

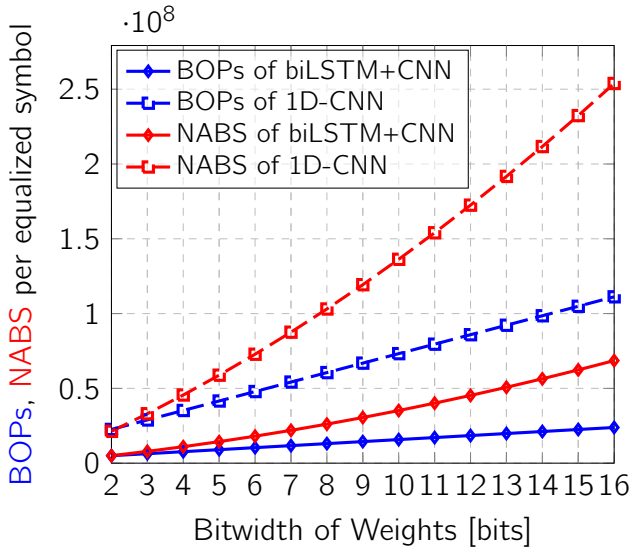


Fig. 4.12 BOPs and NABS per equalized symbol as a function of bitwidth of weights of biLSTM+CNN and 1D-CNN models.

4.7 Conclusion

In this chapter, the knowledge distillation technique has been proposed as an efficient tool to achieve the parallelizability of the recurrent-based equalizers. In this study, KD transfers the knowledge from a recurrent-connection-based biLSTM equalizer to a parallelizable feedforward 1D-CNN. This approach enables the parallelization of signal processing, allowing us to essentially simplify the hardware implementation of NN models. The effectiveness of the KD approach was tested with both simulated and experimental data. The proposed 1D-CNN model was compared against other NN architectures to verify the performance in terms of Q-factor and inference time. In addition, the characteristics of the KD approach on how it assists the student's learning and the limitations of the KD are highlighted. It has also been shown that the proposed feedforward equalizer obtained with KD, results in a significantly reduced signal processing latency compared to the original biLSTM model. In the experimental setup, the student model can perform at the same level as the teacher at the optimal launch power, while in the simulated data, the student slightly reduces the maximum Q-factor by 0.5 dB. In conclusion, the student model can provide 2.2 dB gain compared to the CDC with an improvement of the optimum power by 3 dB in simulated data, while having a 0.7 dB gain with 1 dB increment in optimum launch power in the experimental data.

Chapter 5

Generalizability Improvement via Multi-Task Learning

5.1 Introduction

Apart from the computational complexity and the parallelization, which are the main challenges of the NN-based equalizers, generalizability remains one of the major considerations of NN-based equalizers and attracts more attention [100, 214]. The real-world optical transmission systems are highly dynamic and subject to change over time. This can result in different channel settings, different values of accumulated chromatic dispersion [107], or the presence of channel distortion. Therefore, the equalizers in the receiver or transmitter require reconfiguration and must be adjustable to compensate for the variation of impairments as the channel characteristics change.

This chapter investigates the use of multi-task learning (MTL) to enhance the generalizability and flexibility of the NN-based equalizer. Section 5.2 discusses the concept of MTL and the difference between MTL and the traditional training approach or single-task learning (STL). Section 5.3, based on C3 [108], demonstrates the effectiveness of an MTL-based NN equalizer, which not only improves the equalization performance but also works efficiently in different transmission regimes and scenarios, leading to more generalizable and flexible solutions. Section 5.4, based on J2 [82], shows the experimental validation of the MTL applied in coherent-detection WDM systems, allowing flexibility when the channel spacing varies.

5.2 Multi-Task Learning

STL is a commonly used approach to train NNs. STL refers to the training in which the NN learns the representation of the function to provide the output of a "specific" task [215]. One advantage of STL is that it allows the NN to focus solely on a specific task, usually leading to very good performance in that task. However, the NN may behave poorly when applied to different tasks (e.g., when the transmission scenario of interest is not included in the initial training dataset). As shown in Fig. 5.1, if STL is used for channel equalization in different transmission scenarios, multiple NN models (multiple sets of trained weights) are usually required to provide acceptable performance. To reduce the training complexity in (the training time and resources required) of STL training, transfer learning (TL) can be applied [98]. TL adapts the knowledge acquired in one task to the different tasks (different channel spacing). It is worth noting that to be more effective, the knowledge should be transferred from the transmission scenario with higher nonlinearity to the scenario with lower nonlinearity, for example, from higher launch power to lower launch power. This is because the model learns and adapts better from a more severe nonlinear impairment situation. Even though TL reduces the training complexity, it does not contribute to the inference/implementation complexity, as the multiple sets of weights of the NNs are still required for different transmission scenarios.

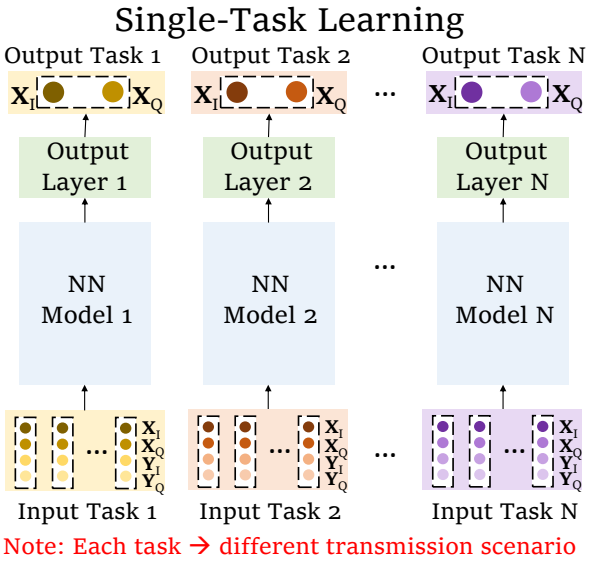
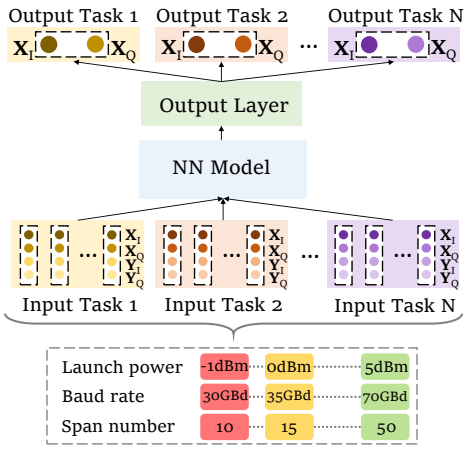
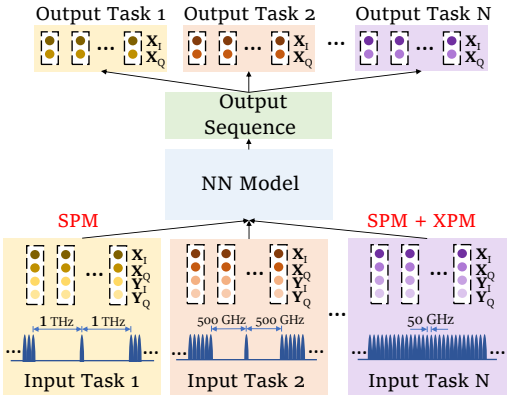


Fig. 5.1 STL: multiple models are required for multiple transmission scenarios.

In MTL, the NN is trained with multiple datasets from multiple related tasks, see Fig. 5.2. In this case, the common representations learned from different but related tasks are shared [216, 215]. The tasks refer to different transmission scenarios. With the MTL, the model learns the different datasets randomly at each training epoch, allowing the model to gain broader knowledge. As depicted in Fig. 5.2, MTL enables a single NN to equalize the signal in different transmission scenarios by the joint training on the datasets from different transmission scenarios. Fig. 5.2a shows the MTL applied when varying the ranges of launch power, symbol rate, and transmission distance in a single-channel system. On the other hand, Fig. 5.2b illustrates the MTL model applied in WDM systems to learn different degrees of nonlinearities (SPM and XPM) through different channel spacings. MTL allows the NN to generalize better by using the domain-specific information contained in the different related tasks [215]. By training a model to solve multiple tasks simultaneously, it can learn representations that generalize better across tasks [217, 218]. In the case that the different tasks are sufficiently related, the shared representation helps improve generalization, even for tasks that might not perform as well in their STL. As a result, it can lead to better performance than learning the tasks independently [219]. While MTL has potential, it does not always perform better than STL, leading to a trade-off between the performance of individual tasks and the overall performance of the model. Success in MTL depends heavily on appropriate



(a) MTL used in single channel systems where different tasks refer to the transmission scenarios with different launch powers, baud rates, and numbers of spans.



(b) MTL used in WDM systems where different tasks refer to the transmission scenarios with different channel spacings between the channel under test and the neighboring channels.

Fig. 5.2 MTL: only one model is required for various situations.

task selection and the degree of information sharing between tasks has to be carefully controlled. Too much sharing can cause a negative information transfer, resulting in performance degradation for each task [216]. Besides the generalization feature enabled by the MTL, it reduces hardware costs. In fact, the shared weights are fixed, which results in the simplification of the multipliers [107].

5.3 MTL in Adaptive Transmission

In this section, MTL [215] is proposed to calibrate the NN-based equalizer used for different transmission conditions in coherent systems. MTL leverages shared representations to enhance the adaptability of NN-based equalizers across different system configurations of single-channel transmission systems. This approach does not require retraining or additional data when the channel conditions change. The considered transmission setup is altered by changing the symbol rate (R_S) and launch power (P) of data channels and the transmission distance (number of spans, N_{Span}). For the MTL, the NN is trained with different datasets resulting from the combination of different transmission setups (to share the weights and biases).

5.3.1 NN Architecture and Training

The NN architecture, depicted in Fig. 5.3, contains a stack of four biLSTM layers with 100 hidden units in each layer coupled with a dense output layer of 2 neurons to deliver the real and imaginary values for the X-polarization. The biLSTM was selected because it outperformed other types of NNs when used for nonlinear compensation [62, 73]. The model took four input features resulting from the in-phase and quadrature components of the complex signal (X_I, X_Q, Y_I, and Y_Q) where $X_I + jX_Q$ and $Y_I + jY_Q$ were the signals in the X and Y polarizations, respectively. A set of 141 input symbols was fed to the NN to recover one symbol at the output. A new set of synthetic data of size 2^{18} was randomly created with different system parameters and used in each training epoch to allow the model to learn different transmission scenarios. The entire training was carried out with a mini-batch size of 2000, and a learning rate of 0.001. The MSE loss estimator and the classical Adam algorithm [175] were applied when training the weights and biases. After the training, the models were evaluated by unseen test sets of size 2^{17} for each testing scenario.

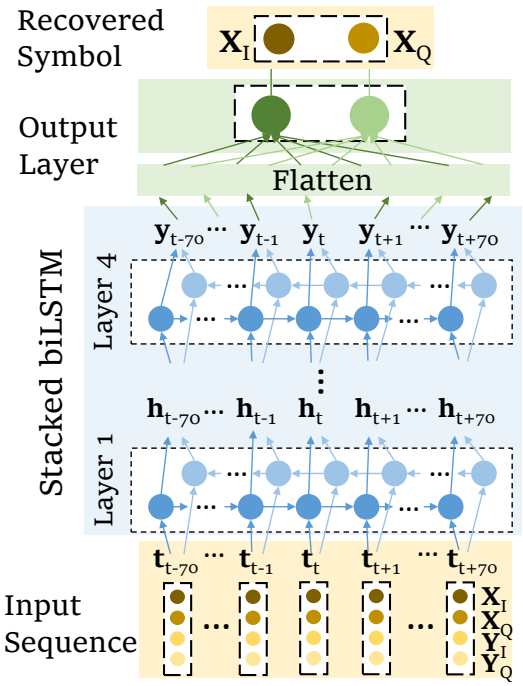


Fig. 5.3 Equalizer architecture with 4-layer biLSTM and a dense layer

The transmission scenarios include R_S ranging from 30 to 70 GBd, number of spans ranging between 10 and 50 (with fixed 50 km span length), and launch power ranging between -1 and 5 dBm. The NNs were trained with MTL or STL as follows:

1. MTL trained for 1000 epochs with datasets including different N_{Span}, but fixed R_S = 40 GBd and P = 5 dBm.

2. MTL trained for 1000 epochs with datasets including different P, but fixed N_{Span} = 50 and R_S = 40 GBd[1].

3. MTL trained for 1000 epochs with datasets including different R_S but fixed N_{Span} = 50 and P = 5 dBm.

4. MTL trained for 1200 epochs with datasets including different combinations of N_{Span}, R_S, and P. This NN is referred to as the "Universal model"[2].

[1]This model has one extra input feature, which is the launch power. The model learns the data during the training using a normalized launch power. Therefore, it could not learn to generalize well without knowing the actual launch power.

[2]Here, the values of R_S and N_{Span} are randomly selected from the list of possible baud rate values with 5 GBd increment and the list of span number with the increments of 5 spans, respectively, to decrease the possible number of combinations for the NN's learning.

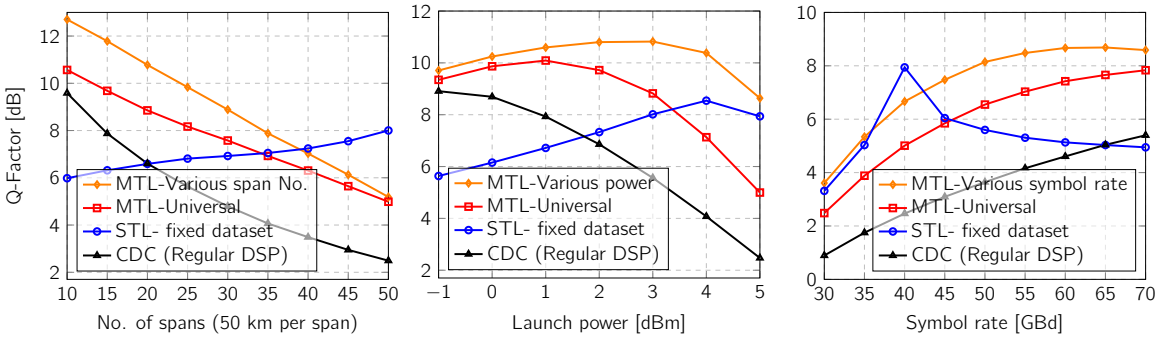
5. STL (without MTL) trained for 1000 epochs with fixed parameters: R_S = 40 GBd, N_{Span} = 50 and P = 5 dBm.

5.3.2 Numerical Setup

The dataset was obtained by numerical simulation assuming the transmission of a single 16-QAM DP channel along the SSMF. The signal propagation through the fiber was represented by a generalized Manakov equation using the GPU-accelerated split-step Fourier method [220]. The SSMF is characterized by the effective nonlinearity coefficient γ = 1.2 (W· km)$^{-1}$, chromatic dispersion coefficient D = 16.8 ps/(nm·km), and attenuation parameter α = 0.21 dB/km. At the end of each fiber span, the optical fiber losses were compensated by an EDFA with a noise figure of 4.5 dB. Downsampling and CDC were performed on the receiver end. Afterward, the received symbols were normalized and used as inputs of the NN.

5.3.3 Results and Discussion I

We considered MTL for multiple symbol rates, transmission distances, and launch powers. To evaluate equalization performance and generalizability, the MTL models were compared to CDC and the STL model trained with a fixed dataset.



(a) Fixed $P = 5$ dBm and $R_s =$ 40 GBd

(b) Fixed $N_{Span} = 50$ and $R_s =$ 40 GBd

(c) Fixed $N_{Span} = 50$ and $P =$ 5 dBm

Fig. 5.4 Q-factor resulting from using MTL (orange and red) and STL model (blue) in the following test cases; (a) when the transmission distance changes but the launch power and symbol rate are set to 5 dBm and 40 GBd, respectively; (b) when the launch power changes but the number of span and symbol rate are set to 50 and 40 GBd, respectively; (c) when the symbol rate changes but the number of spans and launch power are set to 50 and 5 dBm, respectively.

Variation of transmission distance: Fig. 5.4a shows the optical performance for different reaches considering a fixed launch power of 5 dBm and a signal baud rate of 40 GBd. The STL model performed the best when N_{Span} was 50 (because it was trained for this specific transmission scenario), significantly outperforming the remaining approaches. However, its performance was significantly impacted in the shorter reaches as it could not generalize. On the other hand, the MTL trained with different N_{span} showed much better performance than STL for the shorter reaches, achieving a better Q-factor (about 3 dB Q-factor improvement) than CDC only for all considered scenarios. The universal MTL model also showed better performance than the CDC alone, leading to a maximum Q-factor improvement of about 2.5 dB at 50×50 km.

Variation of launch powers: Fig. 5.4b depicts the Q-factor as a function of the launch power for a fixed R_S of 40 GBd and transmission distance of 50×50 km. Again, the STL model showed the best gain for launch powers close to the one it was trained with (5 dBm), but revealed quite poor results for the remaining launch powers. In contrast, the universal MTL model enabled a Q-factor improvement exceeding 2 dB for the most relevant launch powers. The MTL, trained with various P but fixed N_{Span} and R_S, revealed the best performance, enabling a Q-factor improvement exceeding 4 dB for the most relevant launch powers. Interestingly, we can see that, at 5 dBm, the MTL outperformed STL. The reason for this may be that the STL is overfitting and cannot adapt to the unseen test data as effectively as the MTL model, which is more generalized. Ref. [180] supported the claim that a more generalized model can perform better.

Variation of symbol rates: Fig. 5.4c illustrates the Q-factor as a function of the data signal baud rate for a fixed transmission distance of 50×50 km and launch power of 5 dBm. STL led to very good results for the 40 GBd transmission scenario (training scenario) but showed very poor generalization capability. The MTL, trained with multiple R_S but fixed N_{Span} and P, enabled a Q-factor improvement of up to 4.5 dB with respect to the CDC only, whereas the universal MTL model showed up to 2.5 dB improvement. The MTL provided a good gain in most cases.

The aforementioned results show that, although STL may lead to outstanding performance in specific transmission conditions, it is not suitable for real-world system application because it lacks the adaptability to dynamic optical network parameters. MTL overcomes this limitation, allowing the equalizer to be more flexible, but at the cost of small performance degradation compared to models trained only for a specific task.

Multi-task learning is proposed to allow a "single" NN-based equalizer, without retraining, to recover received symbols when the transmission scenarios change. The results showed that the MTL can provide up to 4 dB improvement in Q-factor with respect to CDC alone even if the transmission distance, launch power, and symbol rate vary, thus highlighting the adaptability of the MTL NN-based equalizer to the real-world dynamic optical network.

5.4 Experimental Validation of MTL Applied in WDM Systems

In this work, we trained the NN on different datasets with various channel spacing (different degrees of XPM) in WDM systems. The channel spacing for each dataset is 1000, 500, 400, 300, 200, 100, and 50 GHz from CUT, see Fig. 5.5. In this way, we can observe how the NN tackles varying levels of XPM, focusing on the nonlinear regime. We use MTL to train the NN to attain generalizability. The MTL, illustrated in Fig. 5.2b, enables a single NN to perform in different channel spacing scenarios, as it does not require retraining for different band spacing. In the MTL, the NN is trained jointly with multiple datasets from multiple related tasks; the tasks refer to different channel spacing. MTL leverages shared representations to enhance the generalizability of the NN across the different intensities of XPM. With the MTL, the model learns the different datasets with random channel spacing at each training epoch, allowing the model to gain broader knowledge. We conducted a comparative analysis to assess the MTL's performance against the traditional STL model. The experimental data were collected from three different setups: 9×50 km TWC, 23×50 km SSMF and 12×50 km LEAF.

5.4.1 NN Architecture and Training

The NN architecture described in Fig. 5.5, includes a 1D-CNN layer with n_f filters, a kernel size of 3 and LeakyReLU[3] activation, followed by a biLSTM layer with n_h units, and another 1D-CNN layer with 2 filters to recover real and imaginary parts of X-polarization output. The hyperparameters used to train the NN for each transmission scenario (different fiber types and lengths) are summarized in Table 5.1. The NN

[3]Unlike ReLU, LeakyReLU allows a small gradient for negative values to ensure that the neurons continue learning and helps with the weight updates.

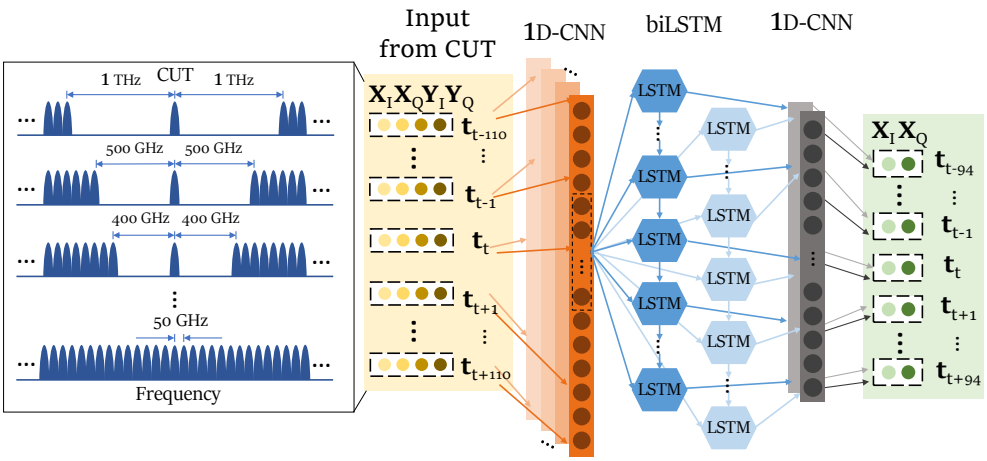


Fig. 5.5 NN equalizer architecture with 1D-CNN and biLSTM layers, taking WDM data of CUT with different channel spacing as an input.

structure with 1D-CNN and biLSTM has proven superior in nonlinear compensation compared to other NN types [221, 73, 89]. The NN received 221 input symbols to simultaneously recover 189 or 191 output symbols, depending on the scenario (see Table 5.1). The training dataset contained 2^{19} independent symbols and, at every epoch, 2^{18} symbols were randomly picked for training. For the testing and validation, a never-before-seen dataset with 2^{17} symbols was utilized. The datasets were created using a pseudo-random binary sequence (PRBS) of order 32. The training was carried out with a mini-batch size, a learning rate shown in Table 5.1, MSE loss, and the Adam algorithm. We also adopted early stopping and data augmentation, adding slight random noise to the training data to avoid overfitting. The optimal hyper-parameters were found by the Bayesian optimizer, given the range of acceptable complexity. The Bayesian optimizer was applied specifically to optimize the hyperparameters for the STL model trained on the 50 GHz channel spacing, due to its most exposure to nonlinear impairments like XPM. Once the optimal hyperparameters were found for the STL model, we applied the same set of hyperparameters for training the MTL model. The model received four input features derived from the in-phase and quadrature components of the complex signal (X_I, X_Q, Y_I, and Y_Q) from X and Y polarizations, respectively. This proposed NN only considers the data from the CUT without exploring the information from adjacent channels. The models with MTL were trained for 1500 epochs (can compensate for all channel spacings considered), whereas the traditional training models were trained for 1000 epochs (to compensate for one channel spacing). To reduce the training complexity of STL training, TL was applied [98]. TL adapts the knowledge

acquired in one task to the different tasks (different channel spacing). We transferred the knowledge from 50 GHz to the larger channel spacing as the model learns and adapts better from a more severe XPM situation. Even though TL reduces the training complexity, it does not contribute to the inference/implementation complexity, as the multiple sets of weights of the NNs are still required for different transmission scenarios.

Transimission	Output Window Size	n_f of 1D-CNN	n_h of LSTM	Learning Rate	Batch Size
TWC 9×50 km	189 Symbols	117	98	9.082×10^{-4}	3786
SSMF 23×50 km	189 Symbols	99	48	1.787×10^{-3}	1823
LEAF 12×50 km	181 Symbols	70	68	5.623×10^{-4}	1000

Table 5.1 Training hyperparameters of each transmission scenario.

5.4.2 Experimental Setup

Fig. 5.6 depicts the experimental setup, as in Ref. [89], where the 16-QAM DP 34.4 GBd symbol sequence was mapped from the data bits generated by a $2^{32}-1$ order PRBS at the transmitter. Then, the channel bandwidth was limited to 37.5 GHz by the RRC filter with 0.1 roll-off. The processed digital samples were passed to a DAC operating at 88 Gsamples/s. The DAC outputs were amplified using a four-channel electrical amplifier, which drove a DP in-phase/quadrature MZM, which in turn modulated a continuous waveform carrier generated by an external cavity laser operating at $\lambda = 1.55$ μm. The resulting optical signal was transmitted over various transmission scenarios as follows:

- Experiment 1: Transmission over 9×50 km TWC fiber spans with EDFA, with up to 95 neighboring channels (100G QPSK, 50 GHz ITU grid). The ADC operated at 50 Gsample/s.
- Experiment 2: Transmission over 23×50 km SSMF fiber spans with EDFA, with up to 40 neighboring channels (100G QPSK, 50 GHz ITU grid). The ADC operated at 88 Gsample/s.
- Experiment 2: Transmission over 12×50 km LEAF fiber spans with EDFA, with up to 40 neighboring channels (100G QPSK, 50 GHz ITU grid). The ADC operated at 88 Gsample/s.

We consider the WDM channels with different channel spacing, see Fig. 5.5. Table 5.2 shows the fiber parameters in terms of attenuation coefficient (α), dispersion coefficient (D) and nonlinear coefficient (γ).

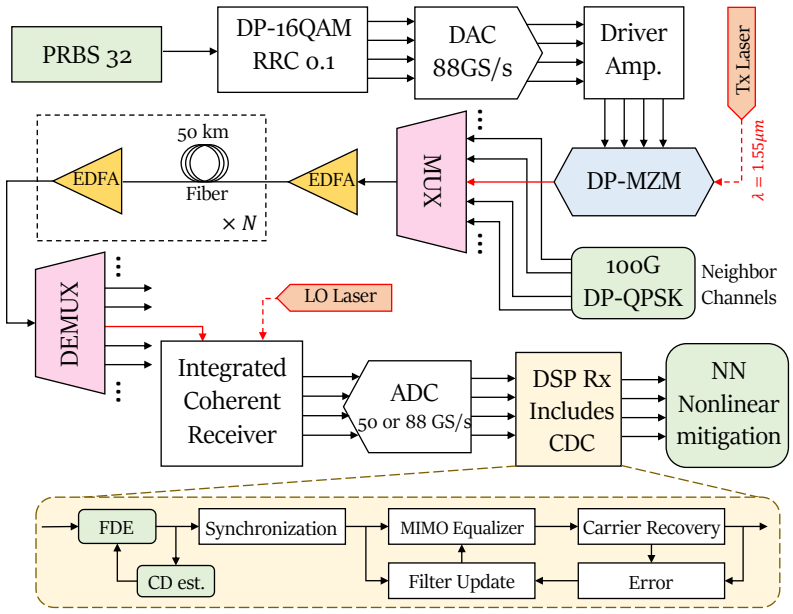


Fig. 5.6 Schematics of the experimental setup used in this work, where the input of the NN is the soft output (before decision unit) of the DSP Rx.

Fiber Type	α [dB/km]	D [ps/(nm· km)]	γ [(W· km)$^{-1}$]
TWC	0.23	2.8	2.5
SSMF	0.195	17.45	1.13
LEAF	0.2	3.9	1.25

Table 5.2 Fiber parameters of TWC, SSMF and LEAF used in the experiment.

At Rx, an integrated coherent receiver converted the signal to the electrical domain, which was sampled using a digital sampling oscilloscope. The signal underwent offline DSP processing as in Ref. [204], including CDC, MIMO equalization, carrier frequency offset correction, clock recovery, and a pilot-aided carrier recovery[4]. The resulting symbols were fed as an NN input for nonlinear mitigation. Finally, at the output of the NN, the pre-forward error correction (pre-FEC) Q-factor was evaluated.

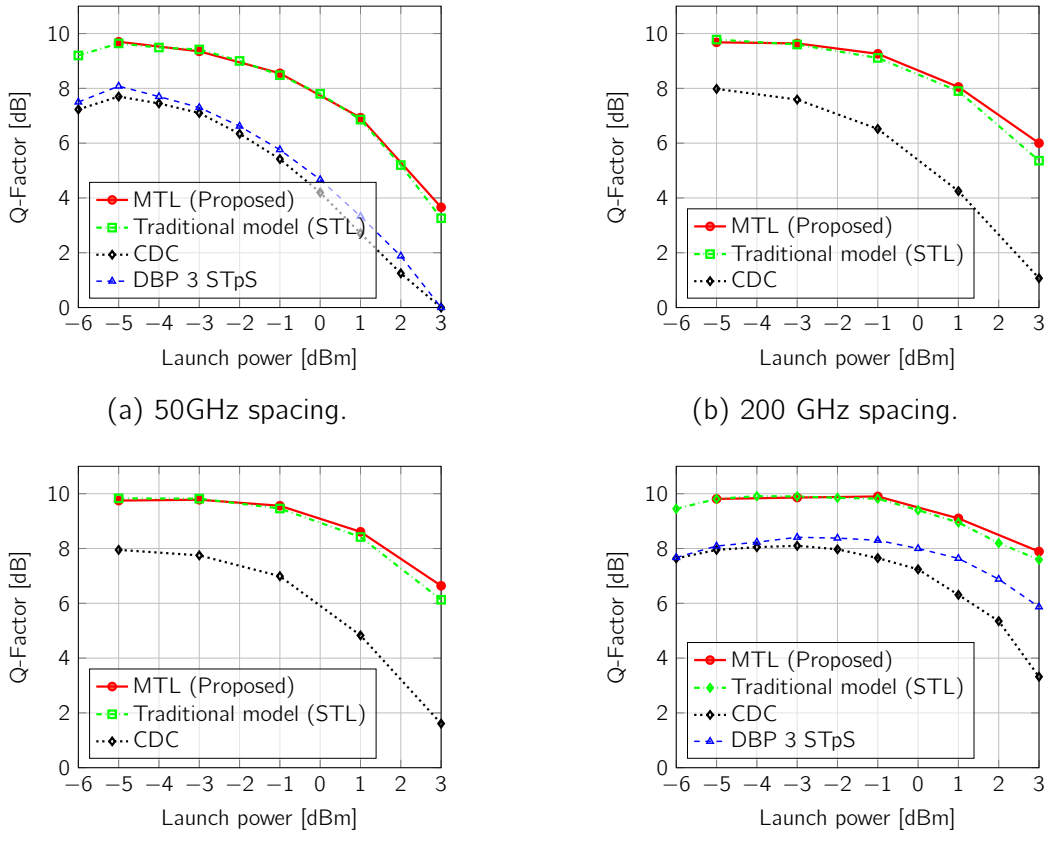
5.4.3 Results and Discussion II

Experiment 1: 9×50 km TWC with up to 95 neighboring channels

Starting with Experiment 1 with 9×50 km TWC, Fig. 5.7 presents the Q-factor as a function of launch power for different channel spacing. Albeit the MTL was

[4]The carrier recovery block in Fig. 5.6 includes both frequency offset compensation and carrier phase recovery

not optimized for the specific cases like in the STL, the MTL models still slightly outperformed the traditional STL approach, especially when the launch power was higher, or the regime was extremely nonlinear. This advantage of MTL was also reported in [219]. We attribute this to the MTL's ability to acquire knowledge across all XPM levels simultaneously. This result shows that the MTL models partially learn the complex patterns associated with the XPM, enhancing the Q-factor compared to the CDC. The DBP with 3 STpS did not perform well, providing only a small gain because the DBP only compensated for SPM; TWC's high nonlinearity coefficient made XPM more significant [164]. The NNs also show the potential to mitigate real-world component-induced impairments beyond Kerr nonlinear effects. The component-induced impairments can be the effects of the transceivers (ADC/DAC, drive amplifier, or MZM), also observed in [25].



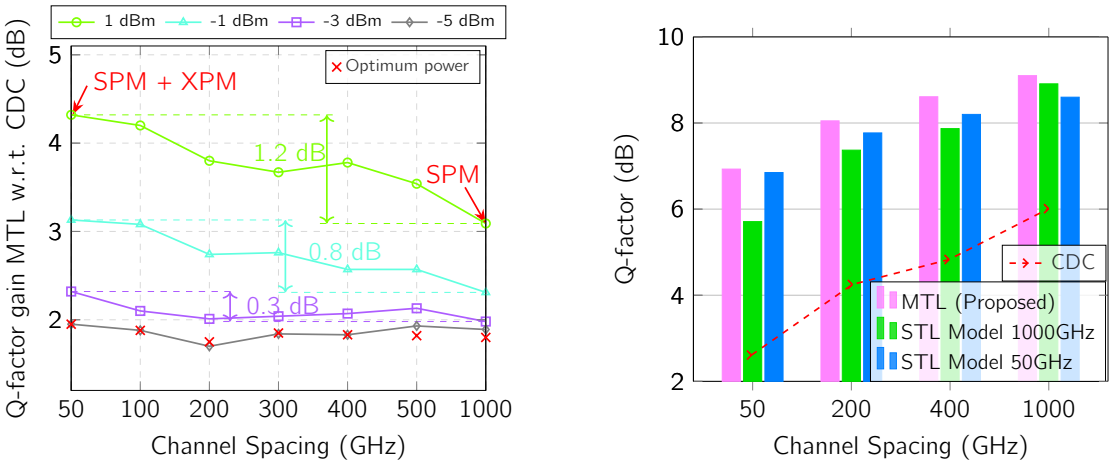
(a) 50GHz spacing.

(b) 200 GHz spacing.

(c) 400 GHz spacing.

(d) Single Channel.

Fig. 5.7 For the 9×50 km TWC fiber transmission, Q-factor versus the launch power for the NN trained with MTL and STL, compared to the CDC and DBP 3 STpS, evaluating when the channel spacing was 50, 200, 400 and 1000 GHz.



(a) (b)

Fig. 5.8 For the 9×50 km TWC fiber transmission, (a) Q-factor gain for MTL models with respect to the CDC performance; (b) Q-factor at 1 dBm launch power of MTL compared to the CDC and two models trained solely with datasets of 50 GHz and 1000 GHz, respectively.

Fig. 5.8a depicts the Q-factor improvement achieved by MTL compared to the CDC for different channel spacings. Considering the optimum power when the channel spacings were 50 GHz and 1000 GHz, the gain of the 50 GHz scenario was around 0.2 dB higher than in the case of 1000 GHz spacing. To assess the performance and robustness of the MTL, we consider the launch power where the fiber nonlinearity started to play a noticeable role, i.e., exceeding -3 dBm. For a narrower channel spacing (SPM and XPM are essential), the gain was even more pronounced than that for a wider channel spacing (where the XPM impact becomes less pronounced). With the increase in power, the difference between the gain at 50 GHz and 1000 GHz channel spacings also increased: the difference was 0.3 dB at -3 dBm launch power, 0.8 dB at -1 dBm launch power and 1.2 dB at 1 dBm launch power. This observation shows that the NN trained with MTL efficiently compensates both SPM and XPM.

	CDC	DBP 3 STpS	Original MTL	MTL with 8 WC
Q-factor [dB]	7.81	8.08	9.7	8.9
RMpS	109	2928	209884	5930

Table 5.3 For the 9×50 km TWC fiber transmission, Q-factor and computational complexity of different methods when considering 50 GHZ spacing at its optimum performance.

To emphasize the NN's adaptability and efficiency in mitigating XPM, Fig. 5.8b shows that at the 1 dBm launch power for inference, the MTL model outperforms both traditional STL models trained on 50 GHz and 1000 GHz channel spacings and the CDC. This showcases the MTL's superior generalization capability across different scenarios. While the STL models trained for specific spacings perform well for the cases matching the specific training conditions, they perform worse when applied to other scenarios. Even though the model trained with 50 GHz spacing data had only a slight drop in Q-factor compared to the MTL, this difference may be larger for other launch powers. Notably, the model trained on 50 GHz spacing generalizes better, likely due to greater exposure to XPM dynamics in dense WDM (DWDM) systems, unlike the 1000 GHz model that exhibits reduced adaptability due to limited XPM exposure.

Tab. 5.3 shows the Q-factor and computational complexity in terms of number of RMpS [55] for CDC, DBP with 3 STpS, original MTL model, and the reduced complexity MTL model using weight clustering technique, detailed in Section 3.2. The weight clustered model groups similar weights into clusters and shares a single value within each cluster; in our case, we used 8 clusters. The weight clustered model was included to demonstrate the possibility of reducing the computational complexity in the MTL NN-based equalizer. While DBP offers about half the complexity of the MTL model with 8 WC, it only compensates for SPM and not XPM. We can notice the trade-off between the Q-factor and complexity. The numbers reported in Tab. 5.3 were assessed using the dataset considering the optimum launch power of the 50 GHz channel spacing scenario. Note that the NN in this work was not optimized for lower complexity.

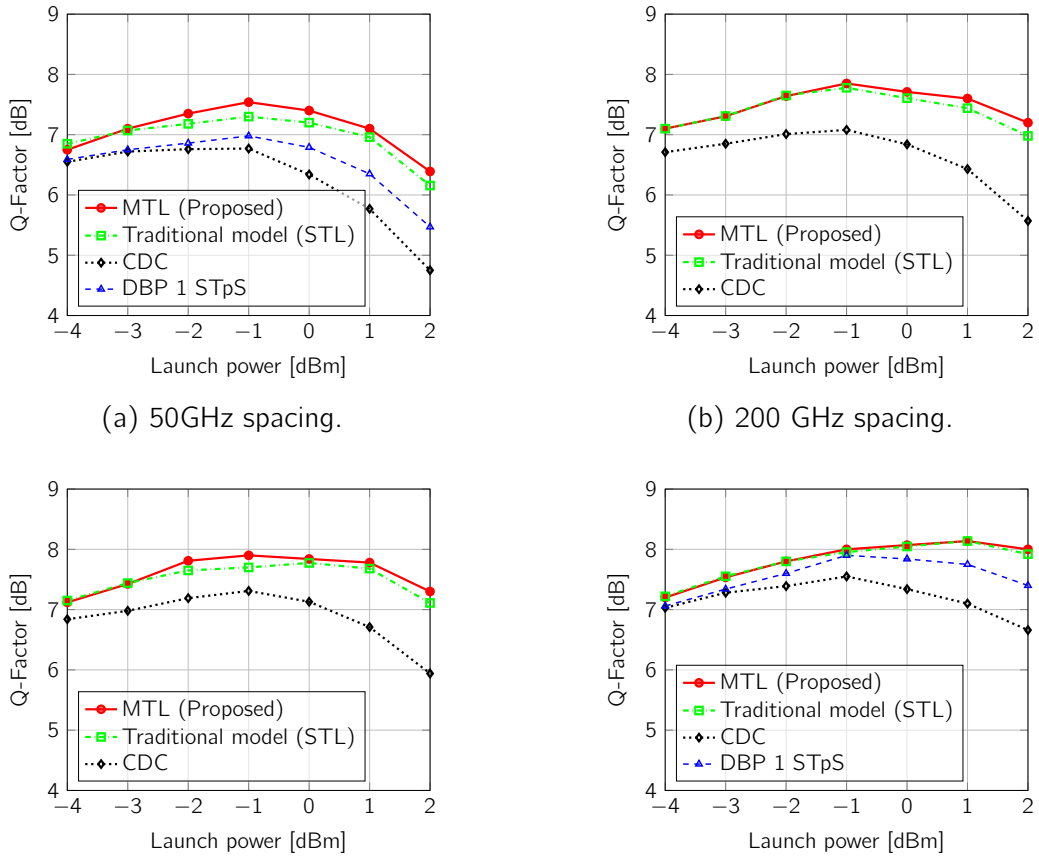
Experiment 2: 23×50 km SSMF with up to 40 neighboring channels

Fig. 5.9 demonstrates the Q factor as a function of launch power for Experiment 2, with 23×50 km SSMF. Overall, the NN-based equalizers, both the MTL and STL outperformed the CDC and DBP with 1 STpS. The MTL clearly outperforms the STL model when the channel spacing is 50 GHz and performs as well as the STL in the other cases. Specifically, the MTL improved the Q-factor by up to 0.77 dB. The classical nonlinearity mitigation method, like DBP, did not provide such significant gain in the DWDM scenario (50 GHz channel spacing) but can perform better in single-channel transmission. This is because the DBP in this work can only compensate for SPM, and the SPM is the major nonlinear impairment in the single-channel transmission. Note

that, in this experiment, increasing the number of steps per span in the DBP did not provide reasonable improvement.

Regarding the gain for the MTL with respect to the CDC, Fig 5.10a showed that the DWDM systems exhibited a higher gain in Q-factor compared to the single-channel transmission. When comparing the gain of the MTL with respect to the CDC in their optimum power, the higher gain in the scenario with 50 GHz can also be observed. It can be implied that the NN-based equalizers can partially mitigate the XPM. The results are similar to Experiment 1.

Fig 5.10b confirms the flexibility of the MTL-based equalizers. All the models are tested at the launch power of 1 dBm. This MTL requires no retraining to be able to compensate for the nonlinear impairments in the transmissions with different channel spacings. The STL model trained only with 50 GHz channel spacing can generalize



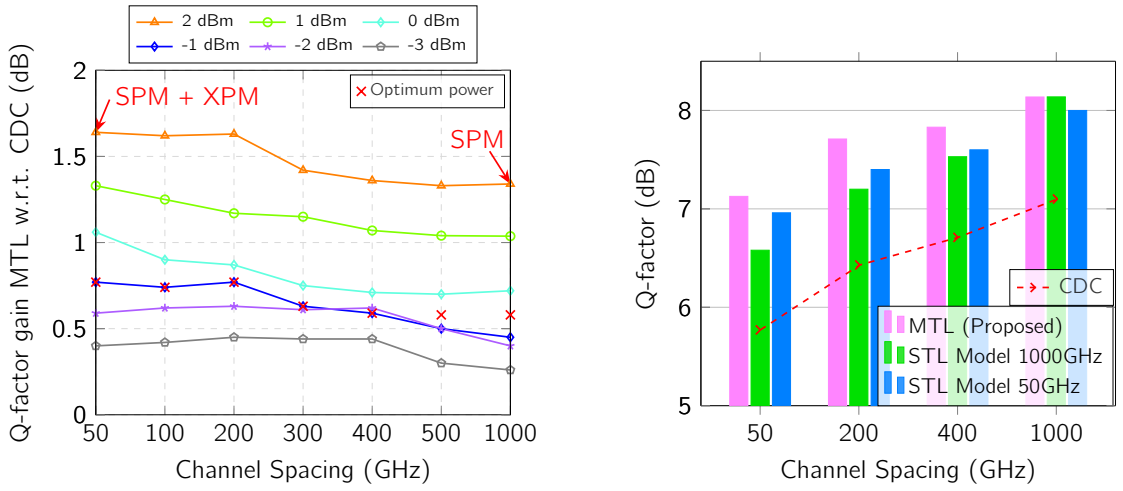
(a) 50GHz spacing.

(b) 200 GHz spacing.

(c) 400 GHz spacing.

(d) Single Channel.

Fig. 5.9 For the 23×50 km SSMF fiber transmission, Q-factor versus the launch power for the NN trained with MTL and STL, compared to the CDC and DBP 1 STpS, evaluating when the channel spacing was 50, 200, 400 and 1000 GHz.



(a) (b)

Fig. 5.10 For the 23×50 km SSMF fiber transmission, (a) Q-factor gain for MTL models with respect to the CDC performance; (b) Q-factor at 1 dBm launch power of MTL compared to the CDC and two models trained solely with datasets of 50 GHz and 1000 GHz, respectively.

better than the STL model trained only with single-channel data because the STL model with 50 GHz spacing learned more nonlinearity from both SPM and XPM.

Lastly, even if the complexity is not the main focus of this chapter, weight clustering was applied in this case to demonstrate the possibility of compressing the MTL model. Table 5.4 concludes the Q-factor and complexity of CDC, DBP with 1 STpS, original MTL and weight-clustered MTL with 8 clusters at their optimum performance (at -1 dBm launch power). With 8 WC, the RMpS is reduced drastically compared to the original model. The MTL model experienced a performance drop when the complexity was reduced. The DBP in this case is cheaper in complexity than the MTL model with 8 WC, however, the performance of the DBP cannot be improved even with a higher number of STpS. In addition, the MTL can be optimized further for lower complexity.

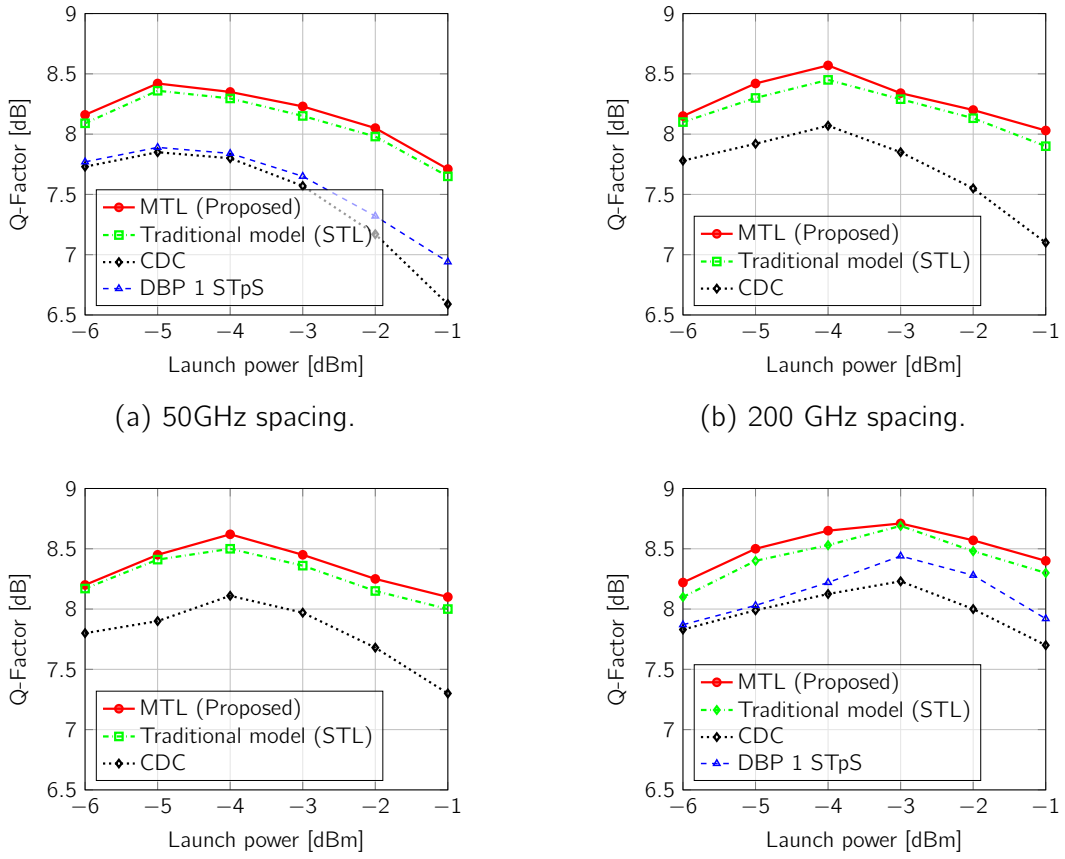
	CDC	DBP 1 STpS	Original MTL	MTL with 8 WC
Q-factor [dB]	6.77	6.98	7.54	7.35
RMpS	146	2430	70222	4120

Table 5.4 For the 23×50 km SSMF fiber transmission, Q-factor and computational complexity of different methods when considering 50 GHZ spacing at its optimum performance.

Experiment 3: 12×50 km LEAF with up to 40 neighboring channels

Similarly to the previous two experiments, in Experiment 3 with 12×50 km LEAF fibers, the MTL also revealed superior performance compared to other methods, including CDC, DBP with 1 STpS and the STL model, see Fig. 5.11. The MTL provided up to 0.6 dB Q-factor improvement compared to the CDC, while the DBP struggled to enhance the Q-factor, especially in the DWDM transmission.

Regarding the gain for the MTL with respect to the CDC, it can be observed in Fig 5.12a that the gain in Q-factor is slightly higher in the WDM system with a 50 GHz channel gap than in the single channel transmission. The result of this experiment validates the assumption made previously that the MTL model can partially mitigate for the XPM.



(a) 50GHz spacing.

(b) 200 GHz spacing.

(c) 400 GHz spacing.

(d) Single Channel.

Fig. 5.11 For the 12×50 km LEAF fiber transmission, Q-factor versus the launch power for the NN trained with MTL and STL, compared to the CDC and DBP 1 STpS, evaluating when the channel spacing was 50, 200, 400 and 1000 GHz.

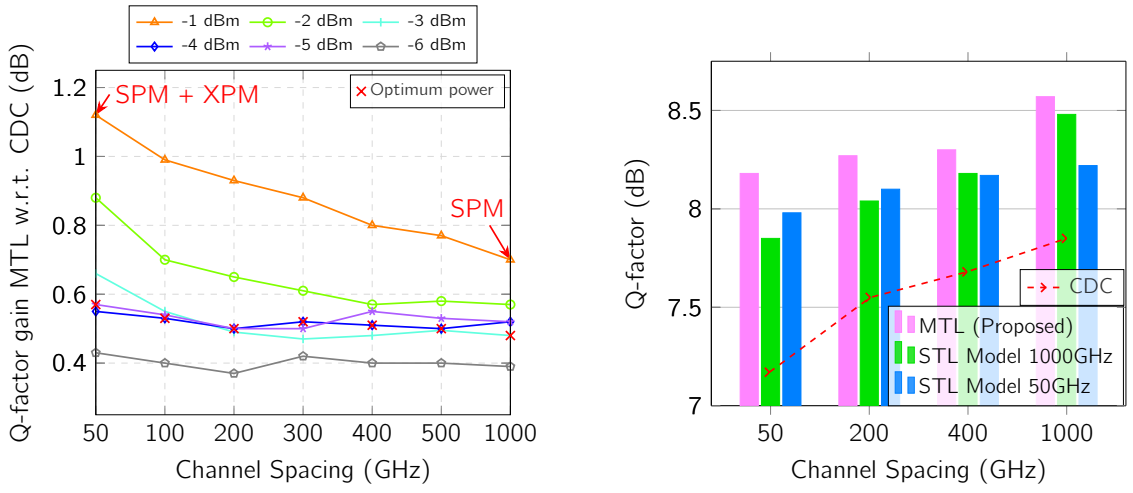


Fig. 5.12 For the 12×50 km LEAF fiber transmission, (a) Q-factor gain for MTL models with respect to the CDC performance; (b) Q-factor at -3 dBm launch power of MTL compared to the CDC and two models trained solely with datasets of 50 GHz and 1000 GHz, respectively.

Fig 5.12b shows the generalizability when the models were tested at a launch power of -3 dBm. The finding indicates similar trends as in the previous experiments, demonstrating that the MTL model has the most generalizability. The STL model trained with 50 GHz spacing appears to be more flexible than the one trained with 1000 GHz spacing, as the former learned from the dataset containing more nonlinearity.

Table 5.5 shows the comparison of the Q-factor and complexity in RMpS of different approaches. The results are similar to the previous two transmission scenarios. The model with 8 WC exhibited a significant reduction in RMpS compared to the original NN model, at a cost of 0.8 dB drop in Q-factor. Even the DBP with 1 STpS required less RMpS than the model with 8 WC, it can only provide 0.04 dB in the equalization

	CDC	DBP 1 STpS	Original MTL	MTL with 8 WC
Q-factor [dB]	7.85	7.89	8.42	8.18
RMpS	144	1252	97792	3980

Table 5.5 For the 12×50 km LEAF fiber transmission, Q-factor and computational complexity of different methods when considering 50 GHZ spacing at its optimum performance.

performance gain compared to the CDC. Note that the model in this work was not optimized for low complexity.

In Subsection 5.3, the MTL model was used to equalize symbols when the channel characteristics varied, and a trade-off between the Q-factor of individual tasks and the overall performance was observed. In the current study, the MTL model applied for varying XPM levels did not demonstrate the same trade-off. Instead, the MTL framework contributed to enhancing each specific task's performance. The MTL's performance boost and its ability to work without retraining are notable advantages compared to employing several STL models for different channel spacings in traditional training. Additionally, in this study, our NN leveraged the dataset from the same experimental setup and a similar NN architecture as in Ref. [89], but our model here was designed to recover multiple output symbols simultaneously. Multi-symbol output not only reduces the computational complexity per symbol but also reveals superior performance compared to its single-symbol counterpart.

In conclusion, we employed MTL to develop an NN-based equalizer capable of mitigating nonlinear impairments in coherent-detection DWDM systems. Learning from the experimental data and leveraging the MTL concept, our MTL NN-based equalizer demonstrates excellent adaptability to varying levels of SPM and XPM. The proposed MTL model has remarkable generalizability, as we need only a "single" model without the necessity of retraining for different channel spacing or diverse levels of XPM. We showed that the MTL model outperforms the CDC, DBP with 3 STpS, and the "traditional" single-task model. The MTL can optimize the performance across all studied tasks without compromising the performance of individual tasks; moreover, we revealed that the MTL can even enhance the performance for specific tasks. Additionally, we explored and assessed the potential for reducing the computational complexity of our MTL-based model, making it more viable for real-world optical communication networks, and demonstrated the possibility of essential complexity reduction for our MTL equalizer without a significant penalty in its performance.

5.5 Conclusion

This chapter explored the application of MTL to improve the generalizability of NN-based equalizers. The MTL was investigated in the single-channel transmission when the symbol rate, transmission distance, and launch power were changing, and in the WDM systems where the channel spacing was altering. From the findings, it can be confirmed

that MTL enables flexibility of the NN-based equalizers across various transmission scenarios. The MTL model utilizes only a single model for different related tasks and does not require retraining. When the tasks are related enough, MTL could potentially outperform STL in Q-factor performance. The results also suggest the possibility of integrating MTL with other complexity reduction techniques, like weight clustering and transfer learning, to further optimize computational efficiency.

Chapter 6

Conclusion and Future Works

6.1 Review of Thesis

To battle the capacity crunch that can occur in the near future, nonlinearity mitigation techniques are required to enhance the capacity of the optical transmission systems. Neural networks (NN) have gained more attention in nonlinearity compensation tasks due to their universal approximation capability. Despite the effectiveness in equalization performance, NN-based equalizers still face some major limitations: computational complexity, parallelization, and generalizability. This thesis examined methods to enhance the computational efficiency of NN-based equalizers in coherent optical communication systems. The focus has been on exploring and proposing possible solutions for the aforementioned three aspects of the challenges of NN-based equalizers.

First, to lower computational complexity, weight clustering has been proposed to substantially lower the number of real multiplications per equalized symbol (RMpS) of the NN-based equalizers used in digital subcarrier multiplexing (DSCM) systems. With this approach, the complexity in terms of RMpS was reduced by up to 97% compared to the original NN, and up to 91% with respect to the NN based on the perturbation analysis. It also offers a reduction of up to 34% in RMpS, compared to the standard digital backpropagation (DBP) 1 STpS. In addition, the different approaches to approximate the nonlinear activation functions of NN were assessed in the single-channel systems to reduce the hardware resources required for the implementation. The findings revealed that the approximated activation functions required significantly fewer hardware resources than the original ones, while maintaining satisfactory performance. This was because the approximation errors were mitigated by the learning of the NNs.

Next, to address the challenge of parallelizability of the promising recurrent NN-based equalizers in high-speed optical communication, knowledge distillation has been employed to transform recurrent NN structures into more hardware-friendly feedforward architectures. With this technique, the feedforward model demonstrated a reduction in inference latency, without a major drop in Q-factor performance. The results, validated with both the simulated and experimental data, showed that the knowledge distillation acted as an efficient regularizer to avoid overfitting when training the NN.

Lastly, tackling the generalizability problems of the NN is crucial to accommodate the dynamic optical systems. Multi-task learning (MTL) was proposed to enhance the flexibility of the NN-based equalizers, so that a single MTL model can mitigate the nonlinearity across different transmission scenarios, without retraining. MTL was tested in both single-channel and WDM transmissions. For the single-channel transmission, the MTL showed an improvement in Q-factor over the linear compensation even when the transmission length, symbol rate and launch power varied. For the WDM systems, experimental validation showed that MTL-based equalizers outperformed traditional single-task models and DBP in performance when channel spacing changed.

In summary, the findings of this thesis contribute to the research area of NN-based equalization by offering a systematic approach to alleviate the limitations of the NN-based equalizer, while preserving robust performance. The complexity reduction techniques were demonstrated with the use case from both simulated and experimental data. These contributions allow a step closer to efficient and scalable NN deployments in real-world optical networks.

6.2 Future Work Direction

In spite of the results and strategies presented in this thesis, several open research directions remain. These open problems should be investigated to further improve the efficiency and practicality of NN-based equalizers in optical communication systems. Possible directions for further research are outlined in the following areas:

- NN-based equalizers should be studied further to be used in WDM and DSCM systems. These systems represent real-world optical networks beyond single-channel transmission. Some studies, including this thesis, have already presented the potential of NN in these systems. However, existing work is limited to only some specific study cases. These WDM and DSCM networks are more complex than

the single-channel systems, due to the higher nonlinearity affected by the neighboring channels/subcarriers. Further research should investigate efficient training strategies and the dependence of the performance of the channel/subcarrier under test on the information of the neighboring channels/subcarriers.

- Other computational complexity reduction techniques should be studied more deeply to explore more opportunities for low-complexity NN-based equalizers. Different approaches can be combined to understand if the NN could still maintain the performance while reducing complexity.

- Meta-learning [222] and multi-task learning are promising research directions to enable flexible and generalizable NN-based equalizers. These techniques have an opportunity to work together such that the multi-task model can learn to adapt to an unseen task in fewer steps. This multi-task meta-learning approach has already been proposed in the image tasks [223].

- Not only the time sequence of the complex symbols, but also the different types of inputs of NN should be investigated. Examples of the input types could be spectrum, triplets, or signal constellations. This is worth comparing to understand the performance and complexity trade-off for different alternatives.

- Physics-based activation functions could be explored to improve the performance and efficiency of NN-based equalizers. Physics-inspired functions can possibly better represent the underlying nonlinearities of the optical channel. This could lead to a more accurate signal equalization and make the NN more interpretable.

- Hardware implementation is another aspect that requires deeper investigation. There are still a limited number of papers on real hardware implementation to ensure practical deployment. Optimizing hardware implementations, including quantization and parallelization, remains a critical challenge for NN-based equalizers.

- Power consumption and power efficiency of the reduced-complexity NN-based equalizers should be quantified and compared as another performance metric. Power consumption is a critical consideration for effective, and sustainable equalizers, especially as networks scale and data rates increase. Efficient power management ensures hardware longevity and reduces operational costs.

Future research can further optimize NN-based equalizers in terms of complexity, performance and flexibility for scalable, low-power, and high-speed optical communication systems. This could make the NN-based equalizers more viable for deployment in next-generation networks.

References

[1] A. Bakhshali, H. Najafi, B. B. Hamgini, and Z. Zhang, "Neural network architectures for optical channel nonlinear compensation in digital subcarrier multiplexing systems," Optics Express, vol. 31, no. 16, pp. 26418–26434, 2023.

[2] O. Gerstel, M. Jinno, A. Lord, and S. B. Yoo, "Elastic optical networking: A new dawn for the optical layer?," IEEE communications Magazine, vol. 50, no. 2, pp. s12–s20, 2012.

[3] A. Ellis, N. M. Suibhne, D. Saad, and D. Payne, "Communication networks beyond the capacity crunch," 2016.

[4] T. Xu, N. A. Shevchenko, Y. Zhang, C. Jin, J. Zhao, and T. Liu, "Information rates in kerr nonlinearity limited optical fiber communication systems," Optics Express, vol. 29, no. 11, pp. 17428–17439, 2021.

[5] P. Bayvel, R. Maher, T. Xu, G. Liga, N. A. Shevchenko, D. Lavery, A. Alvarado, and R. I. Killey, "Maximizing the optical network capacity," Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2062, p. 20140440, 2016.

[6] T. Miya, Y. Terunuma, T. Hosaka, and T. Miyashita, "Ultimate low-loss single-mode fibre at 1.55 μm," Electronics Letters, vol. 15, no. 4, pp. 106–108, 1979.

[7] R. J. Mears, L. Reekie, I. Jauncey, and D. N. Payne, "Low-noise erbium-doped fibre amplifier operating at 1.54 μm," Electronics letters, vol. 23, no. 19, pp. 1026–1028, 1987.

[8] E. Agrell, M. Karlsson, A. Chraplyvy, D. J. Richardson, P. M. Krummrich, P. Winzer, K. Roberts, J. K. Fischer, S. J. Savory, B. J. Eggleton, et al., "Roadmap of optical communications," Journal of optics, vol. 18, no. 6, p. 063002, 2016.

[9] K. Kikuchi, "Fundamentals of coherent optical fiber communications," Journal of lightwave technology, vol. 34, no. 1, pp. 157–179, 2015.

[10] P. Poggiolini and F. Poletti, "Opportunities and challenges for long-distance transmission in hollow-core fibres," Journal of Lightwave Technology, vol. 40, no. 6, pp. 1605–1616, 2022.

[11] D. J. Richardson, J. M. Fini, and L. E. Nelson, "Space-division multiplexing in optical fibres," Nature photonics, vol. 7, no. 5, pp. 354–362, 2013.

[12] J. C. Cartledge, F. P. Guiomar, F. R. Kschischang, G. Liga, and M. P. Yankov, "Digital signal processing for fiber nonlinearities," Optics express, vol. 25, no. 3, pp. 1916–1936, 2017.

[13] S. K. O. Soman, "A tutorial on fiber kerr nonlinearity effect and its compensation in optical communication systems," Journal of Optics, vol. 23, no. 12, p. 123502, 2021.

[14] G. P. Agrawal, "Nonlinear fiber optics," in Nonlinear Science at the Dawn of the 21st Century, pp. 195–211, Springer, 2000.

[15] Y. Kodama, "Optical solitons in a monomode fiber," Journal of Statistical Physics, vol. 39, no. 5, pp. 597–614, 1985.

[16] P. J. Freire, A. Napoli, D. A. Ron, B. Spinnler, M. Anderson, W. Schairer, T. Bex, N. Costa, S. K. Turitsyn, and J. E. Prilepsky, "Reducing computational complexity of neural networks in optical channel equalization: From concepts to implementation," Journal of Lightwave Technology, 2023.

[17] E. Ip and J. M. Kahn, "Compensation of dispersion and nonlinear impairments using digital backpropagation," Journal of Lightwave Technology, vol. 26, no. 20, pp. 3416–3425, 2008.

[18] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," Mechanical systems and signal processing, vol. 151, p. 107398, 2021.

[19] H. Dahrouj, R. Alghamdi, H. Alwazani, S. Bahanshal, A. A. Ahmad, A. Faisal, R. Shalabi, R. Alhadrami, A. Subasi, M. T. Al-Nory, et al., "An overview of machine learning-based techniques for solving optimization problems in communications and signal processing," IEEE Access, vol. 9, pp. 74908–74938, 2021.

[20] M. A. Jarajreh, E. Giacoumidis, I. Aldaya, S. T. Le, A. Tsokanos, Z. Ghassemlooy, and N. J. Doran, "Artificial neural network nonlinear equalizer for coherent optical ofdm," IEEE Photonics Technology Letters, vol. 27, no. 4, pp. 387–390, 2014.

[21] S.-i. Amari and A. Cichocki, "Adaptive blind signal processing-neural network approaches," Proceedings of the IEEE, vol. 86, no. 10, pp. 2026–2048, 1998.

[22] M. Ibnkahla, "Applications of neural networks to digital communications–a survey," Signal processing, vol. 80, no. 7, pp. 1185–1215, 2000.

[23] D. Zibar, M. Piels, R. Jones, and C. G. Schäeffer, "Machine learning techniques in optical communication," Journal of Lightwave Technology, vol. 34, no. 6, pp. 1442–1452, 2015.

[24] M. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," IEE Proceedings-Computers and Digital Techniques, vol. 150, no. 6, pp. 403–411, 2003.

[25] P. J. Freire, S. Srivallapanondh, M. Anderson, B. Spinnler, T. Bex, T. A. Eriksson, A. Napoli, W. Schairer, N. Costa, M. Blott, et al., "Implementing neural network-based equalizers in a coherent optical transmission system using field-programmable gate arrays," Journal of Lightwave Technology, vol. 41, no. 12, pp. 3797–3815, 2023.

[26] A. Amari, O. A. Dobre, R. Venkatesan, O. S. Kumar, P. Ciblat, and Y. Jaouën, "A survey on fiber nonlinearity compensation for 400 gb/s and beyond optical communication systems," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 3097–3113, 2017.

[27] A. Napoli, Z. Maalej, V. A. Sleiffer, M. Kuschnerov, D. Rafique, E. Timmers, B. Spinnler, T. Rahman, L. D. Coelho, and N. Hanik, "Reduced complexity digital back-propagation methods for optical communication systems," Journal of lightwave technology, vol. 32, no. 7, pp. 1351–1362, 2014.

[28] D. Rafique, M. Mussolin, M. Forzati, J. Mårtensson, M. N. Chugtai, and A. D. Ellis, "Modified split-step fourier method for compensation of nonlinear fibre impairments," in 2011 13th International Conference on Transparent Optical Networks, pp. 1–4, IEEE, 2011.

[29] L. Li, Z. Tao, L. Dou, W. Yan, S. Oda, T. Tanimura, T. Hoshida, and J. C. Rasmussen, "Implementation efficient nonlinear equalizer based on correlated digital backpropagation," in Optical Fiber Communication Conference, p. OWW3, Optica Publishing Group, 2011.

[30] C. Fougstedt, C. Häger, L. Svensson, H. D. Pfister, and P. Larsson-Edefors, "Asic implementation of time-domain digital backpropagation with deep-learned chromatic dispersion filters," in 2018 European Conference on Optical Communication (ECOC), pp. 1–3, IEEE, 2018.

[31] L. Liu, L. Li, Y. Huang, K. Cui, Q. Xiong, F. N. Hauske, C. Xie, and Y. Cai, "Intrachannel nonlinearity compensation by inverse volterra series transfer function," Journal of Lightwave Technology, vol. 30, no. 3, pp. 310–316, 2011.

[32] M. Schetzen, "Theory of pth-order inverses of nonlinear systems," IEEE Transactions on Circuits and Systems, vol. 23, no. 5, pp. 285–291, 1976.

[33] S. B. Amado, F. P. Guiomar, N. J. Muga, J. D. Reis, S. M. Rossi, A. Chiuchiarelli, J. R. Oliveira, A. L. Teixeira, and A. N. Pinto, "Experimental demonstration of the parallel split-step method in ultra-long-haul 400g transmission," in 2015 European Conference on Optical Communication (ECOC), pp. 1–3, IEEE, 2015.

[34] A. Amari, P. Ciblat, and Y. Jaouën, "Fifth-order volterra series based nonlinear equalizer for long-haul high data rate optical fiber communications," in 2014 48th Asilomar Conference on Signals, Systems and Computers, pp. 1367–1371, IEEE, 2014.

[35] A. Amari, O. A. Dobre, and R. Venkatesan, "Fifth-order volterra-based equalizer for fiber nonlinearity compensation in nyquist wdm superchannel system," in 2017 19th International Conference on Transparent Optical Networks (ICTON), pp. 1–4, IEEE, 2017.

[36] S. Jansen, D. v. d. Borne, B. Spinnler, S. Calabro, H. Suche, P. Krummrich, W. Sohler, G.-D. Khoe, and H. D. Waardt, "Optical phase conjugation for ultra long-haul phase-shift-keyed transmission," Journal of Lightwave Technology, vol. 24, no. 1, p. 54, 2006.

[37] X. Liu, A. Chraplyvy, P. Winzer, R. Tkach, and S. Chandrasekhar, "Phase-conjugated twin waves for communication beyond the kerr nonlinearity limit," Nature Photonics, vol. 7, no. 7, pp. 560–568, 2013.

[38] O. S. Kumar, O. Dobre, R. Venkatesan, S. Wilson, O. Omomukuyo, A. Amari, and D. Chang, "A spectrally efficient linear polarization coding scheme for fiber nonlinearity compensation in co-ofdm systems," in Next-Generation Optical Communication: Components, Sub-Systems, and Systems VI, vol. 10130, pp. 152–161, SPIE, 2017.

[39] Y. Yu, W. Wang, P. D. Townsend, and J. Zhao, "Modified phase-conjugate twin wave schemes for spectral efficiency enhancement," in 2015 European Conference on Optical Communication (ECOC), pp. 1–3, IEEE, 2015.

[40] Z. Tao, L. Dou, W. Yan, L. Li, T. Hoshida, and J. C. Rasmussen, "Multiplier-free intrachannel nonlinearity compensating algorithm operating at symbol rate," Journal of Lightwave Technology, vol. 29, no. 17, pp. 2570–2576, 2011.

[41] O. S. Kumar, A. Amari, O. A. Dobre, and R. Venkatesan, "Enhanced regular perturbation-based nonlinearity compensation technique for optical transmission systems," IEEE Photonics Journal, vol. 11, no. 4, pp. 1–12, 2019.

[42] M. Malekiha and D. Plant, "Adaptive optimization of quantized perturbation coefficients for fiber nonlinearity compensation," IEEE Photonics Journal, vol. 8, no. 3, pp. 1–7, 2016.

[43] A. Hasegawa and Y. Kodama, Solitons in Optical Communications. Oxford University Press, 1995.

[44] S. K. Turitsyn, J. E. Prilepsky, S. T. Le, S. Wahls, L. L. Frumin, M. Kamalian, and S. A. Derevyanko, "Nonlinear fourier transform for optical data processing and transmission: advances and perspectives," Optica, vol. 4, no. 3, pp. 307–322, 2017.

[45] FONTE Consortium, "Review and optimization results for the nis nft-based systems," Tech. Rep. Deliverable D1.1, Aston University, 2019. Available at: https://fonte.astonphotonics.uk/.

[46] A. Shabat and V. Zakharov, "Exact theory of two-dimensional self-focusing and one-dimensional self-modulation of waves in nonlinear media," Sov. Phys. JETP, vol. 34, no. 1, p. 62, 1972.

[47] M. J. Ablowitz and H. Segur, Solitons and the inverse scattering transform. SIAM, 1981.

[48] J. E. Prilepsky, S. A. Derevyanko, K. J. Blow, I. Gabitov, and S. K. Turitsyn, "Nonlinear inverse synthesis and eigenvalue division multiplexing in optical fiber channels," Physical review letters, vol. 113, no. 1, p. 013901, 2014.

[49] M. I. Yousefi and F. R. Kschischang, "Information transmission using the nonlinear fourier transform, part iii: Spectrum modulation," IEEE Transactions on Information Theory, vol. 60, no. 7, pp. 4346–4369, 2014.

[50] M. I. Yousefi and F. R. Kschischang, "Information transmission using the nonlinear fourier transform, part i: Mathematical tools," IEEE Transactions on Information Theory, vol. 60, no. 7, pp. 4312–4328, 2014.

[51] S. Wahls and H. V. Poor, "Fast inverse nonlinear fourier transform for generating multi-solitons in optical fiber," in 2015 IEEE International Symposium on Information Theory (ISIT), pp. 1676–1680, IEEE, 2015.

[52] M. Li, S. Yu, J. Yang, Z. Chen, Y. Han, and W. Gu, "Nonparameter nonlinear phase noise mitigation by using m-ary support vector machine for coherent optical systems," IEEE Photonics Journal, vol. 5, no. 6, pp. 7800312–7800312, 2013.

[53] E. Giacoumidis, Y. Lin, M. Blott, and L. P. Barry, "Real-time machine learning based fiber-induced nonlinearity compensation in energy-efficient coherent optical networks," APL Photonics, vol. 5, no. 4, p. 041301, 2020.

[54] O. Sidelnikov, A. Redyuk, and S. Sygletos, "Equalization performance and complexity analysis of dynamic deep neural networks in long haul transmission systems," Optics express, vol. 26, no. 25, pp. 32765–32776, 2018.

[55] P. Freire, S. Srivallapanondh, B. Spinnler, A. Napoli, N. Costa, J. E. Prilepsky, and S. K. Turitsyn, "Computational complexity optimization of neural network-based equalizers in digital signal processing: A comprehensive approach," Journal of Lightwave Technology, vol. 42, no. 12, pp. 4177–4201, 2024.

[56] P. Freire, E. Manuylovich, J. E. Prilepsky, and S. K. Turitsyn, "Artificial neural networks for photonic applications—from algorithms to implementation: tutorial," Advances in Optics and Photonics, vol. 15, no. 3, pp. 739–834, 2023.

[57] T. O'shea and J. Hoydis, "An introduction to deep learning for the physical layer," IEEE Transactions on Cognitive Communications and Networking, vol. 3, no. 4, pp. 563–575, 2017.

[58] V. Neskorniuk, A. Carnio, V. Bajaj, D. Marsella, S. K. Turitsyn, J. E. Prilepsky, and V. Aref, "End-to-end deep learning of long-haul coherent optical fiber communications via regular perturbation model," in 2021 European Conference on Optical Communication (ECOC), pp. 1–4, IEEE, 2021.

[59] S. Deligiannidis, K. R. Bottrill, K. Sozos, C. Mesaritakis, P. Petropoulos, and A. Bogris, "Multichannel nonlinear equalization in coherent WDM systems based on bi-directional recurrent neural networks," Journal of Lightwave Technology, vol. 42, no. 2, pp. 541–549, 2024.

[60] S. Gaiarin, X. Pang, O. Ozolins, R. T. Jones, E. P. Da Silva, R. Schatz, U. Westergren, S. Popov, G. Jacobsen, and D. Zibar, "High speed pam-8 optical interconnects with digital equalization based on neural network," in 2016 Asia Communications and Photonics Conference (ACP), pp. 1–3, IEEE, 2016.

[61] M. Schaedler, C. Bluemm, M. Kuschnerov, F. Pittalà, S. Calabrò, and S. Pachnicke, "Deep neural network equalization for optical short reach communication," Applied Sciences, vol. 9, no. 21, p. 4675, 2019.

[62] P. J. Freire, Y. Osadchuk, B. Spinnler, A. Napoli, W. Schairer, N. Costa, J. E. Prilepsky, and S. K. Turitsyn, "Performance versus complexity study of neural network equalizers in coherent optical systems," Journal of Lightwave Technology, vol. 39, no. 19, pp. 6085–6096, 2021.

[63] T. A. Eriksson, H. Bülow, and A. Leven, "Applying neural networks in optical communication systems: Possible pitfalls," IEEE Photonics Technology Letters, vol. 29, no. 23, pp. 2091–2094, 2017.

[64] A. Shahkarami, M. Yousefi, and Y. Jaouen, "Complexity reduction over bi-rnn-based nonlinearity mitigation in dual-pol fiber-optic communications via a crnn-based approach," Optical Fiber Technology, vol. 74, p. 103072, 2022.

[65] P. Li, L. Yi, L. Xue, and W. Hu, "56 gbps im/dd pon based on 10g-class optical devices with 29 db loss budget enabled by machine learning," in 2018 Optical Fiber Communications Conference and Exposition (OFC), pp. 1–3, IEEE, 2018.

[66] C.-Y. Chuang, L.-C. Liu, C.-C. Wei, J.-J. Liu, L. Henrickson, W.-J. Huang, C.-L. Wang, Y.-K. Chen, and J. Chen, "Convolutional neural network based nonlinear classifier for 112-gbps high speed optical link," in Optical Fiber Communication Conference, pp. W2A–43, Optical Society of America, 2018.

[67] P. Li, L. Yi, L. Xue, and W. Hu, "100gbps im/dd transmission over 25km ssmf using 20g-class dml and pin enabled by machine learning," in Optical Fiber Communication Conference, pp. W2A–46, Optica Publishing Group, 2018.

[68] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157–166, 1994.

[69] M. Schaedler, F. Pittalà, S. Calabrò, G. Böcherer, C. Bluemm, and S. Pachnicke, "Recurrent neural network soft demapping for mitigation of fiber nonlinearities and isi," in Optical Fiber Communication Conference, pp. M5F–4, Optica Publishing Group, 2021.

[70] K. Sozos, S. Deligiannidis, G. Sarantoglou, C. Mesaritakis, and A. Bogris, "Recurrent neural networks and recurrent optical spectrum slicers as equalizers in high symbol rate optical transmission systems," Journal of Lightwave Technology, vol. 41, no. 15, pp. 5037–5050, 2023.

[71] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," Neural computation, vol. 12, no. 10, pp. 2451–2471, 2000.

[73] 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, vol. 38, no. 21, pp. 5991–5999, 2020.

[74] C. Häger and H. D. Pfister, "Physics-based deep learning for fiber-optic communication systems," IEEE Journal on Selected Areas in Communications, vol. 39, no. 1, pp. 280–294, 2020.

[75] X. Chi, C. Bai, F. Yang, Q. Qi, R. Zhang, H. Xu, L. Yang, W. Bi, T. Chen, and S. Bai, "Joint intra and inter-channel nonlinear compensation scheme based on improved learned digital back propagation for wdm systems," Optics Express, vol. 32, no. 4, pp. 5095–5116, 2024.

[76] N. Castro, S. Boscolo, A. D. Ellis, and S. Sygletos, "Learned volterra models for nonlinearity equalization in wavelength-division multiplexed systems," Optics Express, vol. 33, no. 8, pp. 16717–16737, 2025.

[77] S. Zhang, F. Yaman, K. Nakamura, T. Inoue, V. Kamalov, L. Jovanovski, V. Vusirikala, E. Mateo, Y. Inada, and T. Wang, "Field and lab experimental demonstration of nonlinear impairment compensation using neural networks," Nature communications, vol. 10, no. 1, p. 3033, 2019.

[78] A. Redyuk, E. Averyanov, O. Sidelnikov, M. Fedoruk, and S. Turitsyn, "Compensation of nonlinear impairments using inverse perturbation theory with reduced complexity," Journal of Lightwave Technology, vol. 38, no. 6, pp. 1250–1257, 2020.

[79] M. M. Melek and D. Yevick, "Nonlinearity mitigation with a perturbation based neural network receiver," Optical and Quantum Electronics, vol. 52, pp. 1–10, 2020.

[80] P. J. Freire, J. E. Prilepsky, Y. Osadchuk, S. K. Turitsyn, and V. Aref, "Deep neural network-aided soft-demapping in coherent optical systems: Regression versus classification," IEEE Transactions on Communications, vol. 70, no. 12, pp. 7973–7988, 2022.

[81] B. Sang, J. Zhang, C. Wang, M. Kong, Y. Tan, L. Zhao, W. Zhou, D. Shang, Y. Zhu, H. Yi, et al., "Multi-symbol output long short-term memory neural network equalizer for 200+ gbps im/dd system," in 2021 European Conference on Optical Communication (ECOC), pp. 1–4, IEEE, 2021.

[82] S. Srivallapanondh, P. Freire, B. Spinnler, N. Costa, W. Schairer, A. Napoli, S. K. Turitsyn, and J. E. Prilepsky, "Experimental validation of xpm mitigation using a generalizable multi-task learning neural network," Optics Letters, vol. 49, no. 24, pp. 6900–6903, 2024.

[83] Y. Liu, V. Sanchez, P. J. Freire, J. E. Prilepsky, M. J. Koshkouei, and M. D. Higgins, "Attention-aided partial bidirectional rnn-based nonlinear equalizer in coherent optical systems," Optics Express, vol. 30, no. 18, pp. 32908–32923, 2022.

[84] X. Huang et al., "Complex principal component analysis-based complex-valued fully connected nn equalizer for optical fibre communications," Optics Express, vol. 31, no. 25, pp. 42310–42326, 2023.

[85] J. Xiang et al., "Low-complexity conditional generative adversarial network (c-gan) based nonlinear equalizer for coherent data-center interconnections," Journal of Lightwave Technology, vol. 41, no. 18, pp. 5966–5972, 2023.

[86] N. Gautam, S. V. Pendem, B. Lall, and A. Choudhary, "Transformer-based nonlinear equalization for dp-16qam coherent optical communication systems," IEEE Communications Letters, vol. 28, no. 3, pp. 577–581, 2024.

[87] Z. Jiang, X. Liu, and L. Zhang, "Wide and deep learning-aided nonlinear equalizer for coherent optical communication systems," Photonics, vol. 11, no. 2, p. 141, 2024.

[88] O. Sidelnikov et al., "Advanced convolutional neural networks for nonlinearity mitigation in long-haul wdm transmission systems," Journal of Lightwave Technology, vol. 39, no. 8, pp. 2397–2406, 2021.

[89] P. J. Freire, Y. Osadchuk, B. Spinnler, W. Schairer, A. Napoli, N. Costa, J. E. Prilepsky, and S. K. Turitsyn, "Experimental study of deep neural network equalizers performance in optical links," 2021 Optical Fiber Communications Conference and Exhibition (OFC), pp. 1–3, 2021.

[90] M. Qiu, Q. Zhuge, M. Chagnon, Y. Gao, X. Xu, M. Morsy-Osman, and D. V. Plant, "Digital subcarrier multiplexing for fiber nonlinearity mitigation in coherent optical communication systems," Optics Express, vol. 22, no. 15, pp. 18770–18777, 2014.

[91] W. S. Saif, S. K. O. Soman, and O. A. Dobre, "Deep learning-assisted nonlinearity compensation in subcarrier-multiplexing coherent optical systems," Journal of Lightwave Technology, 2024.

[92] F. Zhang, Q. Zhuge, M. Qiu, W. Wang, M. Chagnon, and D. V. Plant, "Xpm model-based digital backpropagation for subcarrier-multiplexing systems," Journal of Lightwave Technology, vol. 33, no. 24, pp. 5140–5150, 2015.

[93] H. Zhang, Z. Yu, L. Shu, Z. Wan, Y. Zhao, and K. Xu, "Fiber nonlinearity equalizer using mlp-ann for coherent optical ofdm," in 2019 18th International Conference on Optical Communications and Networks (ICOCN), pp. 1–3, 2019.

[94] I. Aldaya et al., "Compensation of nonlinear distortion in coherent optical ofdm systems using a mimo deep neural network-based equalizer," Optics Letters, vol. 45, no. 20, pp. 5820–5823, 2020.

[95] H. Mrabet, E. Giacoumidis, I. Dayoub, and A. Belghith, "A survey of applied machine learning techniques for optical orthogonal frequency division multiplexing based networks," Transactions on Emerging Telecommunications Technologies, vol. 33, no. 4, p. e4400, 2022.

[96] N. M. Nawi, W. H. Atomi, and M. Z. Rehman, "The effect of data pre-processing on optimized training of artificial neural networks," Procedia Technology, vol. 11, pp. 32–39, 2013.

[97] Z. Xu, C. Sun, T. Ji, H. Ji, and W. Shieh, "Transfer learning aided neural networks for nonlinear equalization in short-reach direct detection systems," in 2020 Optical Fiber Communications Conference and Exhibition (OFC), pp. 1–3, IEEE, 2020.

[98] P. J. Freire, D. Abode, J. E. Prilepsky, N. Costa, B. Spinnler, A. Napoli, and S. K. Turitsyn, "Transfer learning for neural networks-based equalizers in coherent optical systems," Journal of Lightwave Technology, vol. 39, no. 21, pp. 6733–6745, 2021.

[99] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 23–30, IEEE, 2017.

[100] P. J. Freire, B. Spinnler, D. Abode, J. E. Prilepsky, A. Ali, N. Costa, W. Schairer, A. Napoli, A. D. Ellis, and S. K. Turitsyn, "Domain adaptation: The key enabler of neural network equalizers in coherent optical systems," in Optical Fiber Communication Conference, pp. Th2A–35, Optica Publishing Group, 2022.

[101] Q. Zhou, F. Zhang, and C. Yang, "Adann: Adaptive neural network-based equalizer via online semi-supervised learning," Journal of Lightwave Technology, vol. 38, no. 16, pp. 4315–4324, 2020.

[102] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in International conference on machine learning, pp. 1126–1135, PMLR, 2017.

[103] K. Alomar, H. I. Aysel, and X. Cai, "Data augmentation in classification and segmentation: A survey and new strategies," Journal of Imaging, vol. 9, no. 2, p. 46, 2023.

[104] V. Neskorniuk, P. J. Freire, A. Napoli, B. Spinnler, W. Schairer, J. E. Prilepsky, N. Costa, and S. K. Turitsyn, "Simplifying the supervised learning of kerr nonlinearity compensation algorithms by data augmentation," in 2020 European Conference on Optical Communications (ECOC), pp. 1–4, 2020.

[105] S. Velliangiri, S. Alagumuthukrishnan, et al., "A review of dimensionality reduction techniques for efficient computation," Procedia Computer Science, vol. 165, pp. 104–111, 2019.

[106] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," Computers & Geosciences, vol. 19, no. 3, pp. 303–342, 1993.

[107] B. Liu, C. Bluemm, S. Calabrò, B. Li, and U. Schlichtmann, "Area-efficient neural network cd equalizer for 4× 200Gb/s PAM4 CWDM4 systems," 2023 Optical Fiber Communications Conference and Exhibition (OFC), pp. 1–3, 2023.

[108] S. Srivallapanondh, P. J. Freire, A. Alam, N. Costa, B. Spinnler, A. Napoli, E. Sedov, S. K. Turitsyn, and J. E. Prilepsky, "Multi-task learning to enhance generalizability of neural network equalizers in coherent optical systems," 2023 European Conference on Optical Communication (ECOC), pp. 1–4, 2023.

[109] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," arXiv preprint arXiv:1710.01878, 2017.

[110] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?," Proceedings of machine learning and systems, vol. 2, pp. 129–146, 2020.

[111] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," Neurocomputing, vol. 461, pp. 370–403, 2021.

[112] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," arXiv preprint arXiv:2103.13630, 2021.

[113] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[114] L.-N. Wang, W. Liu, X. Liu, G. Zhong, P. P. Roy, J. Dong, and K. Huang, "Compressing deep networks by neuron agglomerative clustering," Sensors, vol. 20, no. 21, p. 6033, 2020.

[115] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin, "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions," in International Conference on Machine Learning, pp. 5363–5372, PMLR, 2018.

[116] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," SIAM review, vol. 51, no. 3, pp. 455–500, 2009.

[117] C. Hager, H. D. Pfister, R. M. Butler, G. Liga, and A. Alvarado, "Revisiting multi-step nonlinearity compensation with machine learning," in 45th European Conference on Optical Communication (ECOC 2019), pp. 1–4, 2019.

[118] N. Gautam, V. Kaushik, A. Choudhary, and B. Lall, "Optidistillnet: Learning nonlinear pulse propagation using the student-teacher model," Optics Express, vol. 30, no. 23, pp. 42430–42439, 2022.

[119] I. Taras and D. M. Stuart, "Quantization error as a metric for dynamic precision scaling in neural net training," arXiv preprint arXiv:1801.08621, 2018.

[120] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 11, pp. 7436–7456, 2021.

[121] M. E. Nojehdeh, L. Aksoy, and M. Altun, "Efficient hardware implementation of artificial neural networks using approximate multiply-accumulate blocks," in 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 96–101, IEEE, 2020.

[122] X. Huang, D. Zhang, X. Hu, C. Ye, and K. Zhang, "Recurrent neural network based equalizer with embedded parallelization for 100gbps/λ pon," in 2021 Optical Fiber Communications Conference and Exhibition (OFC), pp. 1–3, IEEE, 2021.

[123] D. Nichols, S. Singh, S.-H. Lin, and A. Bhatele, "A survey and empirical evaluation of parallel deep learning frameworks," arXiv preprint arXiv:2111.04949, 2021.

[124] B. Cannas, A. Fanni, L. See, and G. Sias, "Data preprocessing for river flow forecasting using neural networks: wavelet transforms and data partitioning," Physics and Chemistry of the Earth, Parts A/B/C, vol. 31, no. 18, pp. 1164–1171, 2006.

[125] A. Ghis, K. Smiri, and A. Jemai, "Mixed software/hardware based neural network learning acceleration.," in ICSOFT, pp. 417–425, 2021.

[126] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, "Hardware approximate techniques for deep neural network accelerators: A survey," ACM Computing Surveys, vol. 55, no. 4, pp. 1–36, 2022.

[127] L. Della Toffola, M. Pradel, and T. R. Gross, "Performance problems you can fix: A dynamic analysis of memoization opportunities," ACM SIGPLAN Notices, vol. 50, no. 10, pp. 607–622, 2015.

[128] Y. Huang, Y. Cheng, A. Bapna, O. Firat, D. Chen, M. Chen, H. Lee, J. Ngiam, Q. V. Le, Y. Wu, and z. Chen, "Gpipe: Efficient training of giant neural networks using pipeline parallelism," in Advances in Neural Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[129] M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, "Deep learning with dynamic computation graphs," in International Conference on Learning Representations, 2016.

[130] K. Basterretxea, J. M. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," IEE Proceedings-Circuits, Devices and Systems, vol. 151, no. 1, pp. 18–24, 2004.

[131] P. J. Freire, A. Napoli, B. Spinnler, N. Costa, S. K. Turitsyn, and J. E. Prilepsky, "Neural networks-based equalizers for coherent optical transmission: Caveats and pitfalls," IEEE Journal of Selected Topics in Quantum Electronics, vol. 28, no. 4, pp. 1–23, 2022.

[132] E. Jacobsen and P. Kootsookos, "Fast, accurate frequency estimators [dsp tips & tricks]," IEEE Signal Processing Magazine, vol. 24, no. 3, pp. 123–125, 2007.

[133] B. Spinnler, "Equalizer design and complexity for digital coherent receivers," IEEE Journal of Selected Topics in Quantum Electronics, vol. 16, no. 5, pp. 1180–1192, 2010.

[134] S. Mirzaei, A. Hosangadi, and R. Kastner, "Fpga implementation of high speed fir filters using add and shift method," in 2006 International Conference on Computer Design, pp. 308–313, IEEE, 2006.

[135] C. Baskin, N. Liss, E. Schwartz, E. Zheltonozhskii, R. Giryes, A. M. Bronstein, and A. Mendelson, "Uniq: Uniform noise injection for non-uniform quantization of neural networks," ACM Transactions on Computer Systems (TOCS), vol. 37, no. 1-4, pp. 1–15, 2021.

[136] B. Hawks, J. Duarte, N. J. Fraser, A. Pappalardo, N. Tran, and Y. Umuroglu, "Ps and qs: Quantization-aware pruning for efficient low latency neural network inference," arXiv preprint arXiv:2102.11289, 2021.

[137] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," arXiv preprint arXiv:1909.13144, 2019.

[138] T. Koike-Akino, Y. Wang, K. Kojima, K. Parsons, and T. Yoshida, "Zero-multiplier sparse dnn equalization for fiber-optic qam systems with probabilistic amplitude shaping," in 2021 European Conference on Optical Communications (ECOC), pp. 1–4, IEEE, 2021.

[139] M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li, "Deepshift: Towards multiplication-less neural networks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2359–2368, 2021.

[140] H. You, X. Chen, Y. Zhang, C. Li, S. Li, Z. Liu, Z. Wang, and Y. Lin, "Shiftaddnet: A hardware-inspired deep network," arXiv preprint arXiv:2010.12785, 2020.

[141] P. Gentili, F. Piazza, and A. Uncini, "Efficient genetic algorithm design for power-of-two fir filters," in 1995 International conference on acoustics, speech, and signal processing, vol. 2, pp. 1268–1271, IEEE, 1995.

[142] J. B. Evans, "Efficient fir filter architectures suitable for fpga implementation," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 41, no. 7, pp. 490–493, 1994.

[143] W. R. Lee, V. Rehbock, K. L. Teo, and L. Caccetta, "Frequency-response masking based fir filter design with power-of-two coefficients and suboptimum pwr," Journal of Circuits, Systems, and Computers, vol. 12, no. 05, pp. 591–599, 2003.

[144] P. Kurup and T. Abbasi, Logic synthesis using Synopsys®. Springer Science & Business Media, 2012.

[145] H. Li and W. Ye, "Efficient implementation of fpga based on vivado high level synthesis," in 2016 2nd IEEE International Conference on Computer and Communications (ICCC), pp. 2810–2813, 2016.

[146] X. Staff, "Gate count capacity metrics for fpgas," Xilinx Corp., San Jose, CA, Application Note XAPP, vol. 59, 1997.

[147] S. Srivallapanondh, P. Freire, G. Parisi, M. Devigili, N. Costa, B. Spinnler, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn, "Weight-clustered neural networks for low-complexity nonlinear equalization in digital subcarrier multiplexing systems," in Optical Fiber Communication Conference, Optica Publishing Group, 2025.

[148] S. Srivallapanondh, P. Freire, G. Parisi, M. Devigili, N. Costa, B. Spinnler, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn, "Low complexity neural network equalizer for nonlinearity mitigation in digital subcarrier multiplexing systems," Optics Express, vol. 33, no. 2, pp. 2558–2575, 2025.

[149] S. Srivallapanondh, P. J. Freire, A. Napoli, S. K. Turitsyn, and J. E. Prilepsky, "Hardware realization of nonlinear activation functions for nn-based optical equalizers," in CLEO: Science and Innovations, pp. SF1F–4, Optica Publishing Group, 2023.

[150] H. Sun, M. Torbatian, M. Karimi, R. Maher, S. Thomson, M. Tehrani, Y. Gao, A. Kumpera, G. Soliman, A. Kakkar, et al., "800g dsp asic design using probabilistic shaping and digital sub-carrier multiplexing," Journal of lightwave technology, vol. 38, no. 17, pp. 4744–4756, 2020.

[151] P. Poggiolini, A. Nespola, Y. Jiang, G. Bosco, A. Carena, L. Bertignono, S. M. Bilal, S. Abrate, and F. Forghieri, "Analytical and experimental results on system maximum reach increase through symbol rate optimization," Journal of Lightwave Technology, vol. 34, no. 8, pp. 1872–1885, 2016.

[152] D. Welch, A. Napoli, J. Bäck, S. Buggaveeti, C. Castro, A. Chase, X. Chen, V. Dominic, T. Duthel, T. A. Eriksson, et al., "Digital subcarrier multiplexing: Enabling software-configurable optical networks," Journal of lightwave technology, vol. 41, no. 4, pp. 1175–1191, 2023.

[153] D. Welch, A. Napoli, J. Bäck, W. Sande, J. Pedro, F. Masoud, C. Fludger, T. Duthel, H. Sun, S. J. Hand, et al., "Point-to-multipoint optical networks using coherent digital subcarriers," Journal of Lightwave Technology, vol. 39, no. 16, pp. 5232–5247, 2021.

[154] Y. Zhang, M. O'Sullivan, and R. Hui, "Digital subcarrier multiplexing for flexible spectral allocation in optical transport network," Optics Express, vol. 19, no. 22, pp. 21880–21889, 2011.

[155] M. Pelikan, "Bayesian optimization algorithm," in Hierarchical Bayesian optimization algorithm: toward a new generation of evolutionary algorithms, pp. 31–48, Springer, 2005.

[156] T. M. O. Team, "Clustering algorithm in tensorflow model optimization." https://github.com/tensorflow/model-optimization/blob/v0.7.2/tensorflow_model_optimization/python/core/clustering/keras/clustering_algorithm.py#L24-L194, 2023. Accessed: 2024-08-21.

[157] D. Marcuse, C. Manyuk, and P. K. A. Wai, "Application of the manakov-pmd equation to studies of signal propagation in optical fibers with randomly varying birefringence," Journal of Lightwave Technology, vol. 15, no. 9, pp. 1735–1746, 1997.

[158] S. Musetti, P. Serena, and A. Bononi, "On the accuracy of split-step fourier simulations for wideband nonlinear optical communications," Journal of Lightwave Technology, vol. 36, no. 23, pp. 5669–5677, 2018.

[159] S. J. Savory, "Digital coherent optical receivers: Algorithms and subsystems," IEEE Journal of selected topics in quantum electronics, vol. 16, no. 5, pp. 1164–1179, 2010.

[160] X. Zhou, "An improved feed-forward carrier recovery algorithm for coherent receivers with *m*-qam modulation format," IEEE Photonics Technology Letters, vol. 22, no. 14, pp. 1051–1053, 2010.

[161] M. Qiu, Q. Zhuge, M. Chagnon, F. Zhang, and D. V. Plant, "Laser phase noise effects and joint carrier phase recovery in coherent optical transmissions with digital subcarrier multiplexing," IEEE Photonics Journal, vol. 9, no. 1, pp. 1–13, 2017.

[162] G. P. Agrawal, Fiber-Optic Communication Systems. Wiley, 5th ed., 2021.

[163] X. Li, X. Chen, G. Goldfarb, E. Mateo, I. Kim, F. Yaman, and G. Li, "Electronic post-compensation of wdm transmission impairments using coherent detection and digital signal processing," Optics express, vol. 16, no. 2, pp. 880–888, 2008.

[164] A. Napoli, D. Rafique, B. Spinnler, M. Kuschnerov, M. Noelle, and M. Bohn, "Performance dependence of single-carrier digital back-propagation on fiber types and data rates," Optical Fiber Communication Conference, pp. W2A–49, 2014.

[165] O. Çetin, F. Temurtaş, and Ş. Gülgönül, "An application of multilayer neural network on hepatitis disease diagnosis using approximations of sigmoid activation function," Dicle Tıp Dergisi, vol. 42, no. 2, pp. 150–157, 2015.

[166] Z. Li, Y. Zhang, B. Sui, Z. Xing, and Q. Wang, "Fpga implementation for the sigmoid with piecewise linear fitting method based on curvature analysis," Electronics, vol. 11, no. 9, p. 1365, 2022.

[167] N. G. Timmons and A. Rice, "Approximating activation functions," arXiv preprint arXiv:2001.06370, 2020.

[168] F. Temurtas, A. Gulbag, and N. Yumusak, "A study on neural networks using taylor series expansion of sigmoid activation function," in International Conference on Computational Science and Its Applications, pp. 389–397, Springer, 2004.

[169] M. R. Spiegel, Mathematical Handbook of Formulas and Tables. New York: McGraw-Hill, 1968.

[170] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," IEE Proceedings-Circuits, Devices and Systems, vol. 144, no. 6, pp. 313–317, 1997.

[171] T. Yang, Y. Wei, Z. Tu, H. Zeng, M. A. Kinsy, N. Zheng, and P. Ren, "Design space exploration of neural network activation function circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 10, pp. 1974–1978, 2018.

[172] A. H. Namin, K. Leboeuf, H. Wu, and M. Ahmadi, "Artificial neural networks activation function hdl coder," in 2009 IEEE international conference on electro/information technology, pp. 389–392, IEEE, 2009.

[173] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient hardware implementation of the hyperbolic tangent sigmoid function," in 2009 IEEE International Symposium on Circuits and Systems, pp. 2117–2120, IEEE, 2009.

[174] L. Wang, X. Dong, Y. Wang, L. Liu, W. An, and Y. Guo, "Learnable lookup table for neural network quantization," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12423–12433, 2022.

[175] A. Gulli and S. Pal, Deep learning with Keras. Packt Publishing Ltd, 2017.

[176] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," arXiv preprint arXiv:1903.05662, 2019.

[177] S. Deligiannidis, C. Mesaritakis, and A. Bogris, "Performance and complexity analysis of bi-directional recurrent neural network models versus volterra nonlinear equalizers in digital coherent systems," Journal of Lightwave Technology, vol. 39, no. 18, pp. 5791–5798, 2021.

[178] A. X. M. Chang and E. Culurciello, "Hardware accelerators for recurrent neural networks on fpga," in 2017 IEEE International symposium on circuits and systems (ISCAS), pp. 1–4, IEEE, 2017.

[179] R. Robey and Y. Zamora, Parallel and high performance computing. Simon and Schuster, 2021.

[180] S. Srivallapanondh, P. J. Freire, B. Spinnler, N. Costa, A. Napoli, S. K. Turitsyn, and J. E. Prilepsky, "Knowledge distillation applied to optical channel equalization: Solving the parallelization problem of recurrent connection," in Optical Fiber Communication Conference, pp. Th1F–7, Optica Publishing Group, 2023.

[181] S. Srivallapanondh, P. J. Freire, B. Spinnler, N. Costa, A. Napoli, S. K. Turitsyn, and J. E. Prilepsky, "Parallelization of recurrent neural network-based equalizer for coherent optical systems via knowledge distillation," Journal of Lightwave Technology, vol. 42, no. 7, pp. 2275–2284, 2024.

[182] G. Hinton, O. Vinyals, J. Dean, et al., "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, vol. 2, no. 7, 2015.

[183] Q. Xu, Z. Chen, M. Ragab, C. Wang, M. Wu, and X. Li, "Contrastive adversarial knowledge distillation for deep model compression in time-series regression tasks," Neurocomputing, vol. 485, pp. 242–251, 2022.

[184] B. Sang, W. Zhou, Y. Tan, M. Kong, C. Wang, M. Wang, L. Zhao, J. Zhang, and J. Yu, "Low complexity neural network equalization based on multi-symbol output technique for 200+ gbps im/dd short reach optical system," Journal of Lightwave Technology, vol. 40, no. 9, pp. 2890–2900, 2022.

[185] S. Lee, D. Jha, A. Agrawal, A. Choudhary, and W.-k. Liao, "Parallel deep convolutional neural network training by exploiting the overlapping of computation and communication," in 2017 IEEE 24th International Conference on High Performance Computing (HiPC), pp. 183–192, IEEE, 2017.

[186] R. A. Hamad, M. Kimura, L. Yang, W. L. Woo, and B. Wei, "Dilated causal convolution with multi-head self attention for sensor human activity recognition," Neural Computing and Applications, vol. 33, no. 20, pp. 13705–13722, 2021.

[187] P. Krishnapriya and A. Iyer, "Power and area efficient implementation for parallel fir filters using ffas and da," International Journal of Advanced Research in Electrical. Electronics and Instrumentation Engineering, vol. 2, no. 1, 2013.

[188] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, 1999.

[189] R. Woods, J. McAllister, and G. Lightbody, FPGA-based implementation of Signal Processing Systems. John Wiley & Sons, 2017.

[190] C. Li, X. Liao, and J. Yu, "Complex-valued recurrent neural network with iir neuron model: training and applications," Circuits, Systems and Signal Processing, vol. 21, no. 5, pp. 461–471, 2002.

[191] S. Zhang, C. Liu, H. Jiang, S. Wei, L. Dai, and Y. Hu, "Feedforward sequential memory networks: A new structure to learn long-term dependency," arXiv preprint arXiv:1512.08301, 2015.

[192] K. K. Parhi, "Chapter 10: Pipelined and parallel recursive and adaptive filters." http://people.ece.umn.edu/users/parhi/SLIDES/chap10.pdf, 1999. Accessed: 2022–08-03.

[193] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," Neural networks, vol. 18, no. 5-6, pp. 602–610, 2005.

[194] B. Karanov, M. Chagnon, V. Aref, F. Ferreira, D. Lavery, P. Bayvel, and L. Schmalen, "Experimental investigation of deep learning for digital signal processing in short reach optical fiber communications," in 2020 IEEE Workshop on Signal Processing Systems (SiPS), pp. 1–6, IEEE, 2020.

[195] Z. Xu, S. Dong, J. H. Manton, and W. Shieh, "Low-complexity multi-task learning aided neural networks for equalization in short-reach optical interconnects," Journal of Lightwave Technology, vol. 40, no. 1, pp. 45–54, 2021.

[196] M. Takamoto, Y. Morishita, and H. Imaoka, "An efficient method of training small models for regression problems with knowledge distillation," in 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 67–72, IEEE, 2020.

[197] J. Xiang, S. Colburn, A. Majumdar, and E. Shlizerman, "Knowledge distillation circumvents nonlinearity for optical convolutional neural networks," Applied Optics, vol. 61, no. 9, pp. 2173–2183, 2022.

[198] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," International Journal of Computer Vision, vol. 129, no. 6, pp. 1789–1819, 2021.

[199] H. Ma, S. Yang, R. Wu, X. Hao, H. Long, and G. He, "Knowledge distillation-based performance transferring for lstm-rnn model acceleration," Signal, Image and Video Processing, pp. 1–8, 2022.

[200] M. R. U. Saputra, P. P. De Gusmao, Y. Almalioglu, A. Markham, and N. Trigoni, "Distilling knowledge from a deep pose regressor network," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 263–272, 2019.

[201] A. K. Menon, A. S. Rawat, S. J. Reddi, S. Kim, and S. Kumar, "Why distillation helps: a statistical perspective," arXiv preprint arXiv:2005.10419, 2020.

[202] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," Advances in neural information processing systems, vol. 30, 2017.

[203] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," ACM Transactions on Modeling and Computer Simulation (TOMACS), vol. 8, no. 1, pp. 3–30, 1998.

[204] M. Kuschnerov, M. Chouayakh, K. Piyawanno, B. Spinnler, E. De Man, P. Kainzmaier, M. S. Alfiad, A. Napoli, and B. Lankl, "Data-aided versus blind single-carrier coherent receivers," IEEE Photonics Journal, vol. 2, no. 3, pp. 387–403, 2010.

[205] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in Proceedings of the twenty-first international conference on Machine learning, p. 78, 2004.

[206] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," arXiv preprint arXiv:1609.03499, 2016.

[207] X. Zhen, R. Chakraborty, N. Vogt, B. B. Bendlin, and V. Singh, "Dilated convolutional neural networks for sequential manifold-valued data," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10621–10631, 2019.

[208] S. Gong, Z. Wang, T. Sun, Y. Zhang, C. D. Smith, L. Xu, and J. Liu, "Dilated fcn: Listening longer to hear better," in 2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), pp. 254–258, IEEE, 2019.

[209] F. Yu, "Multi-scale context aggregation by dilated convolutions," arXiv preprint arXiv:1511.07122, 2015.

[210] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al., "Rectifier nonlinearities improve neural network acoustic models," Proc. icml, vol. 30, no. 1, p. 3, 2013.

[211] D. E. Rumelhart, G. E. Hinton, R. J. Williams, et al., "Learning internal representations by error propagation," 1985.

[212] B. J. Kim, H. Choi, H. Jang, D. G. Lee, W. Jeong, and S. W. Kim, "Guidelines for the regularization of gammas in batch normalization for deep residual networks," arXiv preprint arXiv:2205.07260, 2022.

[213] Google, "Google colab." https://colab.research.google.com/. Accessed: July 31, 2023.

[214] J. Zhang, T. Xu, T. Jin, W. Jiang, S. Hu, X. Huang, B. Xu, Z. Yu, X. Yi, and K. Qiu, "Meta-learning assisted source domain optimization for transfer learning based optical fiber nonlinear equalization," Journal of Lightwave Technology, vol. 41, no. 5, pp. 1269–1277, 2022.

[215] R. Caruana, Multitask learning. Springer, 1998.

[216] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," arXiv preprint arXiv:2009.09796, 2020.

[217] Y. Zhang and Q. Yang, "A survey on multi-task learning," IEEE transactions on knowledge and data engineering, vol. 34, no. 12, pp. 5586–5609, 2021.

[218] K.-H. Thung and C.-Y. Wee, "A brief review on multi-task learning," Multimedia Tools and Applications, vol. 77, no. 22, pp. 29705–29725, 2018.

[219] Y. Zhang and Q. Yang, "An overview of multi-task learning," National Science Review, vol. 5, no. 1, pp. 30–43, 2018.

[220] E. Sedov, "Hpcom," Apr. 2023.

[221] X. Luo, C. Bai, X. Chi, H. Xu, Y. Fan, L. Yang, P. Qin, Z. Wang, and X. Lv, "Nonlinear impairment compensation using transfer learning-assisted convolutional bidirectional long short-term memory neural network for coherent optical communication systems," Photonics, vol. 9, no. 12, p. 919, 2022.

[222] X. Xiao, Z. Zhou, B. Dong, D. Ma, L. Zhou, and J. Sun, "Meta-dsp: A meta-learning approach for data-driven nonlinear compensation in high-speed optical fiber systems," arXiv preprint arXiv:2311.10416, 2023.

[223] R. Upadhyay, P. C. Chhipa, R. Phlypo, R. Saini, and M. Liwicki, "Multi-task meta learning: learn how to adapt to unseen tasks," in 2023 International Joint Conference on Neural Networks (IJCNN), pp. 1–10, IEEE, 2023.

Appendix A

A.1 PWL equations

The equations of the PWL approximations of sigmoid and tanh can be found in Table A.1 for 3, 5, 7, and 9 segments.

No. of segments	Functions			
	tanh		sigmoid	
	Equation	Condition	Equation	Condition
3	1	$x > 1.1$	1	$x > 2.2$
	$0.90909x$	$-1.1 < x \leq 1.1$	$0.22727x + 0.5$	$-2.2 < x \leq 2.2$
	-1	$x \leq -1.1$	0	$x \leq -2.2$
5	1	$x > 1.7$	1	$x > 2.6$
	$0.41666x + 0.29166$	$0.5 < x \leq 1.7$	$0.17223x + 0.55219$	$0.8 < x \leq 2.6$
	x	$-0.5 < x \leq 0.5$	$0.23747x + 0.5$	$-0.8 < x \leq 0.8$
	$0.41666x - 0.29166$	$-1.7 < x \leq -0.5$	$0.17223x + 0.44781$	$-2.6 < x \leq -0.8$
	-1	$x \leq -1.7$	0	$x \leq -2.6$
7	1	$x > 1.8$	1	$x > 3$
	$0.285x + 0.48699$	$1.1 < x \leq 1.8$	$0.12363x + 0.62909$	$1.4 < x \leq 3$
	$0.57214x + 0.17114$	$0.4 < x \leq 1.1$	$0.18701x + 0.54036$	$0.8 < x \leq 1.4$
	x	$-0.4 < x \leq 0.4$	$0.23747x + 0.5$	$-0.8 < x \leq 0.8$
	$0.57214x - 0.17114$	$-1.1 < x \leq -0.4$	$0.18701x + 0.45964$	$-1.4 < x \leq -0.8$
	$0.285x - 0.48699$	$-1.8 < x \leq -1.1$	$0.12363x + 0.37091$	$-3 < x \leq -1.4$
	-1	$x \leq -1.8$	0	$x \leq -3$
9	1	$x > 2.2$	1	$x > 3.4$
	$0.14331x + 0.68417$	$1.4 < x \leq 2.2$	$0.08514x + 0.71051$	$2 < x \leq 3.4$
	$0.3381x + 0.412$	$0.9 < x \leq 1.4$	$0.12644x + 0.62791$	$1.5 < x \leq 2$
	$0.269382x + 0.09185$	$0.3 < x \leq 0.9$	$0.182242x + 0.09185$	$0.8 < x \leq 1.5$
	x	$-0.3 < x \leq 0.3$	$0.23747x + 0.5$	$-0.8 < x \leq 0.8$
	$0.269382x - 0.09185$	$-0.9 < x \leq -0.3$	$0.08514x + 0.45585$	$-1.5 < x \leq -0.8$
	$0.3381x - 0.412$	$-1.4 < x \leq -0.9$	$0.12644x + 0.37209$	$-2 < x \leq -1.5$
	$0.14331x - 0.68417$	$-2.2 < x \leq -1.4$	$0.182242x + 0.28949$	$-3.4 < x \leq -2$
	-1	$x \leq -2.2$	0	$x \leq -3.4$

Table A.1 PWL approximation equations of sigmoid and tanh for 3, 5, 7 and 9 segments.