Low-Complexity Efficient Neural Network Optical Channel Equalizers: Training, Inference, and Hardware Synthesis

Pedro J. Freire(1), Sasipim Srivallapanondh(1), Bernhard Spinnler(2), Antonio Napoli(2), Nelson Costa(3), Jaroslaw E. Prilepsky(1), Sergei K. Turitsyn(1)

(1) Aston University, United Kingdom freiredp@aston.ac.uk  (2) Infinera, Munich, Germany  (3) Infinera Unipessoal Lda., Carnaxide, Portugal

Abstract The paper discusses strategies for achieving low-complexity neural network-based equalizers for optical communications, addressing challenges in training, inference, and hardware synthesis.

Introduction
Machine learning methods have proven to be efficient in several applications in optical communications, particularly in channel equalization[1]. Deep neural network (NN) structures have demonstrated the potential to significantly enhance transmission quality in various scenarios[2], with computational complexity comparable or better than that of classical approaches[3],[4]. However, additional work is still required to reach product-level applications. One of the most pressing challenges is to demonstrate how such NN-based equalizers can be effectively implemented, taking into account both the training and inference phases. In this contribution, we provide an overview of the main strategies enabling low computational complexity NN-based equalizers, from training to hardware synthesis, while still maintaining attractive Q-factor gains. The metrics used to evaluate the effectiveness of these complexity reduction strategies are also presented. Lastly, a practical study of some complexity reduction approaches is carried out to demonstrate the performance and complexity trade-offs of different methods.

Overview on Complexity Reduction In Neural Network Equalizers
To achieve optimal computational complexity in the pursuit of efficient NN equalizers for resource-constrained hardware, it is necessary to thoroughly investigate three crucial phases: training, inference, and hardware synthesis phases. Fig. 1 illustrates the most common techniques applied to reduce complexity in each of these phases.

First, during training, reducing complexity is crucial for the efficient and practical deployment of hardware with limited resources. Techniques such as transfer learning or approaches to improve generalization, such as data augmentation, domain randomization, and semi-supervised learning, can be applied. These approaches indirectly reduce the need for large amounts of original training data. Effective generalization reduces the need for complex models with a high number of parameters, resulting in faster and more efficient training. Transfer learning adapts the knowledge acquired in the source tasks to the target tasks, considerably reducing the training time and resources required. This method was investigated in equalization tasks in both directly[5] and coherently[6] detected systems. Data augmentation allows datasets to be more diverse and representative by artificially generating additional data points from existing data[7], resulting in a less-overfitting model. The model requires fewer parameters and, consequently, converges faster, as shown in[8] for the NN equalizers. Domain randomization improves the robustness of models[9] by training them with randomized simulated data. This approach reduces the dependence on real-world data, thus improving the training efficiency[10]. Semi-supervised learning leverages both labeled and unlabeled data in training, thus enhancing 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[11] for channel equalization.

Next, during inference, the NN must accurately equalize the input signal using the minimum computational resources while meeting the required performance metrics. This result can be achieved using network pruning, sparse representation, knowledge distillation (KD), and tensor decomposition. Network pruning reduces the complexity of NNs by removing parameters, neurons, or layers that do not significantly affect equalization performance, as described in[2],[12],[13]. The sparse representation compresses the NN weights and biases by keeping only a small number of non-
zero coefficients. Quantization, binarization, and weight sharing or weight clustering (W.C.) are examples of this category. This approach has the potential to reduce complexity and memory usage while still maintaining model performance. 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 and can accelerate the inference of the NN equalizer. Tensor decomposition decomposes high-dimensional data into a lower-dimensional space. In the authors showed that the sparse decomposition of the tensor in convolutional filters can successfully reduce model complexity and memory usage during inference.

Finally, 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. In this case, the multiplier and adder can be approximated to reduce the hardware resource requirements. The approximation replaces full-precision multipliers and adders with low-precision alternatives, such as binary or ternary multipliers. Another possibility is to use parallelization, which involves partitioning the NN into multiple sub-networks to be processed in parallel for faster and more efficient execution. In addition, memoization and skipping can also be used to reduce computational costs. Memoization stores intermediate results in memory to avoid recalculation, while skipping selectively skips certain computations based on their relevance to the final output.

**Complexity Metrics (Training and Inference)**

After implementing the mentioned strategies, it is essential to evaluate their effectiveness. This will give a comprehensive understanding of the model's complexity during the training and real-time inference phases on the target hardware platform. Fig. 2 shows the metrics used to measure the complexity of both phases.

Classic metrics to assess the complexity in the training phase include the number of trainable parameters and training time required to achieve the desired performance. However, these metrics are poor benchmarks since two NNs with the same number of trainable parameters can have very distinct training complexity and the training time depends on the hardware resources used and the size of the training dataset. To address these issues, two additional metrics are proposed. The first one is the product of the number of epochs and the number of batches (NENB), which reflects the model's computational demand. 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. The second one assesses flexibility/generalizability 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.

Different metrics can be used to evaluate the computational complexity, from the software level to the hardware level, in the inference phase. Starting at the software level, Big-O can be used to describe the growth rate of an algorithm's complexity as the input size increases. Then, the number of real multiplications can be evaluated, ignoring additions. To study the fixed-point arithmetic, the number of bit-operations must be assessed to evaluate the impact of changing the bit-width precision on the complexity. After that, to assess the effect of the quantization strategies, the number of additions and bit shifts counts as the number of total equivalent additions to represent the multiplication operation. Additionally, the complexity of the real hardware implementation can be assessed by calculating the number of logic gates (e.g., the number of flip-flops, registers, and logic blocks).

Finally, power consumption should also be assessed as it may be a bottleneck during the implementation phase.
**A Practical Study on the Benefits of Complexity Reduction in Neural Network Equalizer**

This section presents a practical study of the impact of the complexity reduction techniques on the optical performance of a numerically simulated single 64 QAM 30 GBd dual-polarization channel transmitted along 20×50km of SSMF. The NN structure is a biLSTM+CNN with 100 hidden units and an input window of 221 symbols used to recover 171 output symbols, as explained in [2]. In this case, the launch power was set to 2 dBm, which leads to the highest Q-factor after nonlinear equalization.

![Fig. 2: Main metrics to evaluate complexity in the training and inference stages for NN-equalizers.](image)

**Fig. 2**: Main metrics to evaluate complexity in the training and inference stages for NN-equalizers.

In this study, the performance of NN is evaluated when quantization-aware training (QAT) is implemented to determine whether training can mitigate the error introduced by the low bit precision of the NN weights. Fig. 3 summarizes the Q-factor as a function of bitwidth, considering quantization with different schemes: Uniform, Power-of-Two (PoT), Additive Power-of-Two (APoT) with 2 terms, and W.C. The original model without quantization, the standard 1 Step-per-span (StPS) digital backpropagation (DBP), and Chromatic Dispersion Compensation (CDC) are used as a benchmark. As expected, as the complexity is reduced (lower precision/bitwidth), the NN tends to perform worse in all the complexity reduction schemes considered. The analysis of Fig. 3 shows that the W.C. approach outperforms the other quantization techniques due to its ability to learn the optimal alphabet for each part of the NN structure instead of using a static quantized alphabet like uniform, PoT, and APoT methods. Consequently, the performance of the equalizers is less affected. Notably, with a 6-bit alphabet size of clusters, the W.C. technique achieves similar performance to the original model, while a 2-bit alphabet size performs similarly to the 1 StPS DBP. Regarding the trade-off between optical performance and computational complexity, the APoT approach with two terms may be a feasible option, as it only requires one adder for each multiplication, as detailed in [24].

We note that the use of QAT can result in an unstable training process which requires continuous training monitoring. Moreover, when considering smaller bit levels, we recommend implementing a gradual quantization approach in which the precision is gradually decreased during the training process while optimizing the learning rate and batch size.

**Conclusions**

In this study, we present key approaches for developing low-complexity neural network (NN) equalizers in the training, inference, and hardware synthesis stages. To assess the effectiveness of these techniques, we introduce the primary metrics for evaluating the complexity of the NN. Finally, we conduct a practical study to investigate the impact of complexity reduction methods on equalization performance. Our findings demonstrate that the implementation of complexity reduction strategies may lead to the NN model with reduced complexity while still maintaining a Q-factor level similar to the original model.

**Acknowledgments**: This work has been supported by the EU H2020 Marie Skodowska-Curie Action project MENTOR (No. 956713), EPSRC project TRANSNET, and the H2020 B5G-OPEN project (G.A. 10101663).

---

1 We assume that the weights only are quantized while the inputs are still considered to be float32.

2 The training phase of a quantized model may encounter obstacles associated with learning, such as the exploration versus exploitation trade-off, hyperparameter sensitivity, loss leading to NaN, or gradient problems.
References


