JOURNAL OF
Optical Communications
and Networking

Low-complexity end-to-end deep learning framework for 100G-PON

Yongxin Xu, Xiaokai Guan, Wenqing Jiang, Xudong Wang, Weisheng Hu, and Lilin Yi*

State Key Laboratory of Advanced Optical Communication Systems and Networks, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
**lilinyi@sjtu.edu.cn*

Received 11 June 2024; revised 4 September 2024; accepted 30 September 2024; published 17 October 2024

End-to-end learning allows communication systems to achieve optimal performance compared with conventional blockwise structure design. By modeling the channel with neural networks and training the transmitter and receiver on this differentiable channel, the whole system can be jointly optimized. However, in existing schemes, channel modeling methods, such as the generative adversarial network and long short-term memory network, have complex architectures and cannot track channel changes, leading to less effective end-to-end learning. Meanwhile, the complexity of neural networks deployed at the transmitter and receiver is too high for practical applications. In this work, we propose an efficient and low-complexity end-to-end deep learning framework and experimentally validate it on a 100G passive optical network. It uses a noise adaptation network to model channel response and noise distribution and employs offline pretraining and online tracking training to improve the efficiency and accuracy of channel modeling. For the transmitter, it consists of a pattern-dependent look-up table (PDLUT) based on a neural network (NN-PDLUT) with a single convolutional layer. Further, the receiver is also an NN with a single convolutional layer; thus, the end-to-end signal processing is extremely simple. The experimental results show that end-to-end learning improves the receiver sensitivity by 0.85 and 1.59 dB compared with receiver-only equalization based on Volterra nonlinear equalization (VNLE) and joint equalization based on a PDLUT and a feed-forward equalizer, respectively. Moreover, the number of multiply–accumulate operations consumed by the transmitter and receiver in the end-to-end learning scheme is reduced by 75.7% compared with VNLE-based receiver-only equalization.

https://doi.org/10.1364/JOCN.532742

1. INTRODUCTION

Over the past decade, the passive optical network (PON) has been evolving continuously in terms of data rate per wavelength, architecture, and capacity to support new business and applications in 5G and the coming 6G era. Now that 50G-PON has been standardized by the International Telecommunication Union (ITU-T) and the Institute of Electrical and Electronics Engineers (IEEE) [1,2], the ITU-T has started the study of the next PON standard of very high-speed PON (VHSP) [3], aiming for a line rate beyond 50 Gb/s. Thus far, there is no extensive agreement on whether VHSP will use direct or coherence detection and whether the downstream rate is 100 or 200 Gb/s, for the reason of the cost per bit per meter. Intensity modulation and direct detection (IM/DD) is a cost-effective implementation method for PON and has been accompanying the physical layer of PON up to 50 Gb/s. However, as the data rate further increases, IM/DD faces serious bandwidth limitation, modulator nonlinearity, fiber chromatic dispersion, and power budget issues. A low-complexity and high-performance system impairment compensation scheme is urgently needed [4–6].

In recent years, deep learning technology has made progress in the fields of natural language, image, and video processing. It also earns a lot of attention in research on optical communication, including compensating for signal impairments [7,8], optimizing algorithms [9], modeling channels [10,11], and monitoring system performance [12]; further, deep neural networks or deep learning algorithms demonstrate prominent advantages over traditional schemes for these tasks. To improve the overall system performance, end-to-end deep learning of optical communication system is proposed [13]. Unlike the blockwise design style of traditional optical communication systems, end-to-end deep learning replaces separate modules (e.g., encoding/decoding, modulation/demodulation, pulse shaping, equalization, etc.) at the transmitter and receiver with a pair of neural networks, which are then jointly trained over a differentiable channel. Through end-to-end learning, the suboptimal problem of independent module design can be avoided, and the system performance can achieve the global optimum as much as possible.

Generally, in an end-to-end communication system, two symmetric neural networks are used as the transmitter and receiver, which are regarded as an autoencoder (AE) structure. Based on how the AE is trained, end-to-end learning can be categorized into three types of schemes. The first one is to portray the channel with a formulaic model, such as using formulaic models to represent devices like analog-to-digital converters (ADCs) and digital-to-analog converters (DACs), modulators, and photoelectric detectors [13] or using the split-step Fourier method (SSFM) [14] or regular perturbation model [15,16] to represent fiber channels. Then, the AE can complete gradient backpropagation through the channel represented by the formula during training. The modeling in this type of scheme has the advantage of high interpretability, but, in actual experiments, various effects interact with each other, making it difficult to characterize the channel accurately. This leads to some discrepancy between the channel characterized by the formula and the real physical channel, affecting end-to-end learning performance.

The second one uses a data-driven approach to overcome this problem [17,18]. Researchers train a neural network to model the real channel with channel input and output data via, for example, using a long short-term memory (LSTM) network [19], a generative adversarial network (GAN) [20–22], a deep fully connected neural network (DNN), [23] or an interpretable DNN that incorporates a mathematical model of an optical transmission system [24]. The differentiable channel is then fixed, and the AE is trained on it. A data-driven modeling approach can achieve more accurate channel models. However, channel changes cannot be tracked in this scheme, when the experimental channel undergoes slow changes (power or bias point drift, etc.), the channel modeled with historical information may be inaccurate, affecting the end-to-end optimization results. Moreover, retraining LSTM- or GAN-based channel models consumes a lot of time because LSTM cannot be processed in parallel, and a GAN contains a pair of models that need to be trained interactively.

The last scheme is completely different from the previous two schemes. It eliminates the need for channel modeling and instead performs end-to-end learning directly on the real channel. Similar to the derivative-free optimization in optimization theory, this scheme achieves joint optimization of the transmitter and receiver by using a policy gradient algorithm [25] to estimate the gradient of backpropagation or the cubature Kalman filter algorithm [26] to update trainable weights. But they usually come at the expense of training efficiency and require longer training times due to the instability of the gradient estimate in the experiments.

In these end-to-end learning algorithms, only the system performance is focused on, while the complexity of the neural networks at the transmitter and receiver is usually ignored, which leads to their inability to be practically applied. Considering the training efficiency and complexity of end-to-end learning, a new end-to-end learning framework is proposed. For the differentiable channel, we start from fitting a probability distribution function. According to the principle of maximum likelihood estimation, we design a noise adaptation network to model the channel response and the impact of system noise on transmitted signals. The structure of the noise adaptation network adopts the feed-forward multiscale deep neural network (MscaleDNN) [27,28], which possesses the capability to effectively learn the features at different frequencies. To adapt to different channel conditions and the slow changes of the channel state, a new training strategy combining offline pretraining and memory-buffer assisted online tracking training is designed. For different link configurations, we can directly load the corresponding offline pretrained parameters into the channel network and then utilize online training to periodically collect new channel data in the memory buffer, randomly sampling data from it to train the channel network. For the transmitter, we design a neural network-based pattern-dependent look-up table (NN-PDLUT) and then combine it with a convolutional layer to precompensate for the nonlinear and linear impairments in the system. A pattern-dependent look-up table (PDLUT) [29–31] is commonly used by adding a correction term to symbols to be transmitted, offering simple implementation and efficient nonlinear compensation. For the receiver, we only use a convolutional layer to compensate for the residual linear impairments. As a result, the entire end-to-end learning framework is highly adaptable and has low complexity.

In order to validate the proposed end-to-end learning framework, comparative experiments are conducted on a 100 Gbps O-band IM/DD PON system with a 10 GHz Mach–Zehnder modulator (MZM). The compared schemes include the proposed end-to-end learning, receiver-only equalization based on a feed-forward equalizer (FFE), and Volterra nonlinear equalization (VNLE), as well as joint transmitter and receiver equalization based on a PDLUT and an FFE. Experimental results indicate that the end-to-end learning framework has a fast convergence rate, and the noise adaptation network can accurately model the experimental channel, enabling the transmitter and receiver to provide clean compensation for linear and nonlinear impairments in the system. Compared with receiver-only equalization based on VNLE and joint equalization based on a PDLUT and an FFE, end-to-end learning improves the receiver sensitivity by 0.85 and 1.59 dB, respectively. Moreover, the number of multiply–accumulate (MAC) operations consumed by the transmitter and receiver in the end-to-end learning scheme is reduced by 75.7% compared with VNLE-based receiver-only equalization. The proposed end-to-end learning scheme demonstrates the advantages of high performance and low complexity.

The main contributions of this paper can be summarized as follows:

(1) A noise adaptation network architecture is proposed to simultaneously model the channel frequency response and noise effects. It has a simple structure and is easy to train.
(2) Offline pretraining with the memory-buffer-assisted online tracking training method is designed to improve the channel model's adaptability to channel state changes and accelerate end-to-end learning efficiency.

(3) A low-complexity transceiver signal processing scheme is proposed, and significant performance gain is achieved through end-to-end learning.

The rest of this paper is organized as follows. First, the basic components and principles of the proposed end-to-end learning framework are introduced in Section 2. Then, the experimental setup and experimental results are presented in Sections 3 and 4, respectively. Finally, a conclusion is given in Section 5.

2. END-TO-END DEEP LEARNING PRINCIPLES

The proposed end-to-end deep learning framework is shown in Fig. 1. At the transmitter, random bit sequence $\boldsymbol{T}_{\mathbf{bits}}$ is first modulated into symbols $\boldsymbol{T}_{\mathbf{symbols}}$ and then preprocessed by the TxNN to generate transmitted signal samples $\boldsymbol{T}_{\mathbf{samples}}$. After transmission over the optical fiber link, the received signal samples $\boldsymbol{R}_{\mathbf{samples}}$ is processed by the RxNN. Finally, the obtained equalized symbols $\boldsymbol{E}_{\mathbf{symbols}}$ are demodulated to a bit sequence $\boldsymbol{R}_{\mathbf{bits}}$.

The components that need to be learned in the framework include the TxNN, the RxNN, and the noise adaptation channel. The TxNN and the RxNN are neural networks deployed at the transmitter and receiver, respectively. They are responsible for signal processing tasks. The noise adaptation channel is also a neural network for modeling optical fiber links and acts as a differentiable channel used for gradient backpropagation for joint training of the TxNN and the RxNN. The goal of end-to-end learning is to minimize the joint losses $\text{Loss}_1 + \text{Loss}_2$.

Next, the principles and training methods of each component in the end-to-end deep learning framework are introduced in detail.

A. Noise Adaptation Channel

As a platform for end-to-end learning, the more accurately a differentiable channel inscribes the physical channel, the better the end-to-end learning will perform. Previous channel modeling approaches mainly used a GAN and its variants or an LSTM network and then train the networks by collecting signal data offline. However, both of them have poor training efficiency. A GAN includes two sets of networks, a generator and a discriminator, which need to be trained alternately. LSTM has a recurrent structure; as a result, parallel computation is not supported in LSTM. Besides, when the channel state changes slowly, the offline trained channel network loses match with the physical channel, degrading the end-to-end training performance.

Considering that the physical channel output is a random probability distribution, we can fit this arbitrary distribution

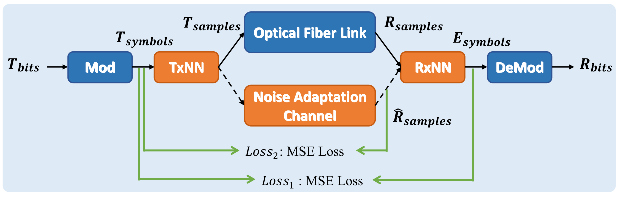


Fig. 1. End-to-end deep learning framework.

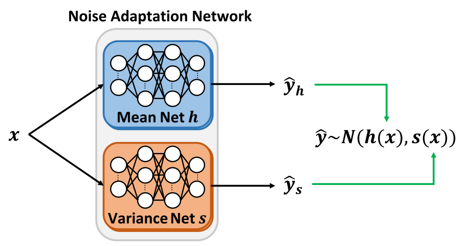


Fig. 2. Noise adaptation network.

using a Gaussian mixture model ([32], Chapter 3). For general short-reach IM/DD optical fiber communication systems, it is usually sufficient to use a Gaussian distribution. Thus, we design a noise adaptation network $C_{h,s}$ to simulate the channel response and noise distribution. Its structure is shown in Fig. 2, including a mean network h and a variance network s (for complex probability distribution scenarios, we can expand multiple mean networks and variance networks to form a Gaussian mixture model). $\boldsymbol{x} \in \mathbb{R}^{m\times \text{sps}}$ is the input of the network, obtained by sliding the window on $\boldsymbol{T}_{\mathbf{samples}} \in \mathbb{R}^{L_{\text{sym}}\times \text{sps}}$. L_{sym} is the length of $\boldsymbol{T}_{\mathbf{symbols}}$, m is the size of the sliding window, and sps is the number of samples per symbol. The output of the noise adaptation network is $\hat{\boldsymbol{y}} \in \mathbb{R}^{\text{sps}}$, and $\hat{\boldsymbol{y}}$ follows an sps-dimensional Gaussian distribution with mean value $\boldsymbol{\mu} \in \mathbb{R}^{\text{sps}}$ and covariance matrix $\Sigma \in \mathbb{R}^{\text{sps}\times\text{sps}}$. The probability density function (PDF) of $\hat{\boldsymbol{y}}$ is as follows:

$$f(\hat{\boldsymbol{y}}) = \frac{1}{(2\pi)^{\frac{\text{sps}}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}\left[(\hat{\boldsymbol{y}}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\hat{\boldsymbol{y}}-\boldsymbol{\mu})\right]\right\}, \tag{1}$$

where $\boldsymbol{\mu} = h(\boldsymbol{x})$, $\boldsymbol{\Sigma} = \text{diag}(s(\boldsymbol{x}))$, and $\text{diag}(\cdot)$ denotes the diagonalization of a vector. $h(\boldsymbol{x})$ and $s(\boldsymbol{x})$ are the outputs of the mean network and variance network, respectively. When the noise adaptation network serves as the simulated channel, the channel output is randomly sampled from the distribution shown in Eq. (1).

Both the mean and variance networks use the MscaleDNN as their network structure. The MscaleDNN is a kind of network architecture designed based on the frequency principle [28] and possesses the capability to effectively learn the response at different frequencies. The structure of the MscaleDNN is shown in Fig. 3, as a sum of K subnetworks, in which each scale input goes through a subnetwork, and $\alpha_1, \alpha_2, \ldots, \alpha_K$ are scale coefficients, usually set as $\alpha_i = i$ or $\alpha_i = 2^{i-1}$. The concrete subnetwork structure is a feed-forward structure, containing an input 1D convolutional layer, two linear layers with activation function LeakyReLU, and an output 1D convolutional layer.

The noise adaptation network is trained based on the maximum log-likelihood function method. $\boldsymbol{T}_{\mathbf{samples}}$ and $\boldsymbol{R}_{\mathbf{samples}}$ are used to produce the training data set, input data $\boldsymbol{x}_n$ is $\left[\boldsymbol{T}_{\mathbf{samples}}(n-\frac{m-1}{2},:), \ldots, \boldsymbol{T}_{\mathbf{samples}}(n,:), \ldots, \boldsymbol{T}_{\mathbf{samples}}(n+\frac{m-1}{2},:)\right]^T \in \mathbb{R}^{\times\text{sps}}$, and corresponding label $\boldsymbol{y}_n$ is $\boldsymbol{R}_{\mathbf{samples}}(n,:) \in \mathbb{R}^{\text{sps}}$. Let $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \ldots, (\boldsymbol{x}_B, \boldsymbol{y}_B)\}$ denote the B groups of data and labels sampled from the training data set. Then, the maximized log-likelihood function can be expressed as follows:

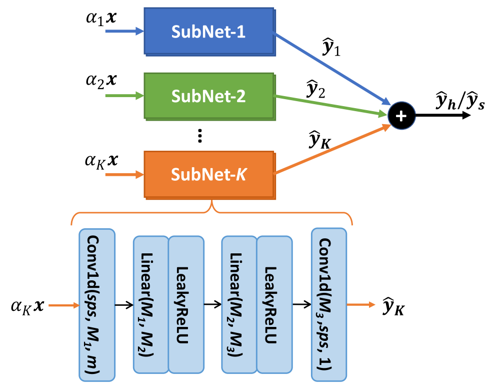


Fig. 3. Structure of the MscaleDNN and its subnetwork.

$$\begin{aligned}&\arg\max_{h,s}\sum_{i=1}^{B}\ln\left[\frac{1}{(2\pi)^{\frac{\text{sps}}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}}\exp\left\{-\frac{1}{2}\left[(\boldsymbol{y}_i-\boldsymbol{\mu}_i)^T\right.\right.\right.\\&\left.\left.\left.\times\ \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i-\boldsymbol{\mu}_i)\right]\right\}\right]\\&=\arg\max_{h,s}\sum_{i=1}^{B}\left[-\frac{1}{2}\left[(\boldsymbol{y}_i-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{y}_i-\boldsymbol{\mu}_i)\right]\right.\\&\left.-\ \frac{\text{sps}}{2}\ln|\boldsymbol{\Sigma}_i|-\frac{\text{sps}}{2}\ln 2\pi\right].\end{aligned} \tag{2}$$

B. Memory-Buffer-Assisted Online Channel Training

After constructing the channel network, we need to collect input and output data from the physical channel to train the network. However, in a practical environment, it is inevitable to encounter problems such as modulator bias point drift, power fluctuations, etc. This leads to the fact that the channel model based on historical data does not represent the current state of the channel. Differences between the channel model and the physical channel can lead to poor end-to-end learning results.

To overcome this, a memory-buffer-assisted online channel training method is introduced, paired with an offline pre-training method, to shorten the channel modeling time, track channel changes, and maintain a stable end-to-end learning process. As shown in Fig. 4, the noise adaptation channel is first pretrained using historical channel data. Then, during the end-to-end learning process, the channel network is switched to tracking training mode. Transmitted signal samples $\boldsymbol{T}_{\text{samples}}$ and received signal samples $\boldsymbol{R}_{\text{samples}}$ from the physical channel are periodically saved in pairs to the memory buffer to provide the most up-to-date training data reflecting the current channel state. It is worth emphasizing that $\boldsymbol{T}_{\text{samples}}$ used for the training channel follow a uniform distribution of $[-1, 1]$, regardless of offline pretraining and online tracking training.

By combining offline pretraining and online tracking training, the convergence of the loss function is accelerated when training the channel network, allowing the channel network to finish modeling the real physical channel faster. It further helps to increase the training speed of the TxNN and the RxNN and improve the efficiency of end-to-end optimization because the TxNN, the RxNN, and the channel network are trained alternatively.

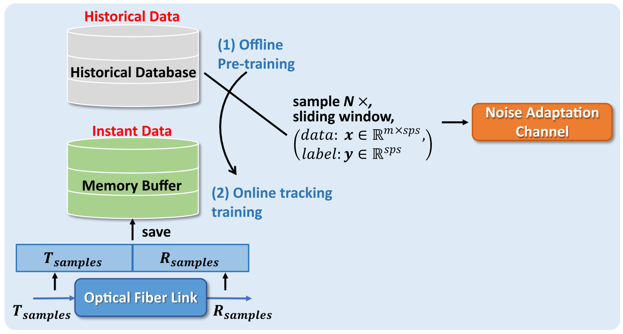


Fig. 4. Memory-buffer-assisted online channel training.

C. Transmitter and Receiver

Generally, in an end-to-end communication system, two neural networks are arranged at the transmitter and receiver to constitute an AE structure. However, the excessive complexity of these networks is often overlooked. To realize a low-complexity end-to-end learning framework, we build the TxNN and RxNN with a simple PDLUT and a single linear layer. The PDLUT can mitigate pattern-dependent distortion and nonlinear impairments by adding a correction term to symbols to be transmitted. In the classical training approach, the PDLUT is made with received symbols after compensating for intersymbol interference (ISI), then placed at the transmitter. Although it is convenient to train independently in this way, the results may not be optimal. By contrast, end-to-end learning offers the possibility to achieve joint optimization and flexible pre- and postprocessing.

The designed structure of the TxNN T_θ is shown in Fig. 5. It contains two parts: one part is the NN-PDLUT, and the other is a single convolutional layer. Further, θ denotes the weights and bias of the TxNN. For the NN-PDLUT, a sliding window with a fixed memory length of L is used to separate the transmitted symbols $\boldsymbol{T}_{\text{symbols}}$ into a pattern-dependent symbol sequence $\left[\boldsymbol{T}_{\text{symbols}}\left(n-\frac{L-1}{2}\right),\ldots,\boldsymbol{T}_{\text{symbols}}(n),\ldots,\boldsymbol{T}_{\text{symbols}}\left(n+\frac{L-1}{2}\right)\right]$, the modulation order of $\boldsymbol{T}_{\text{symbols}}$ is M, so there are M^L pattern sequence types. Then, the LUT index i is determined by the pattern sequence and converted into a one-hot vector as the input of the NN-PDLUT. The NN-PDLUT uses a residual network architecture composed of four linear layers; the two middle hidden layers use ELU as an activation function. The

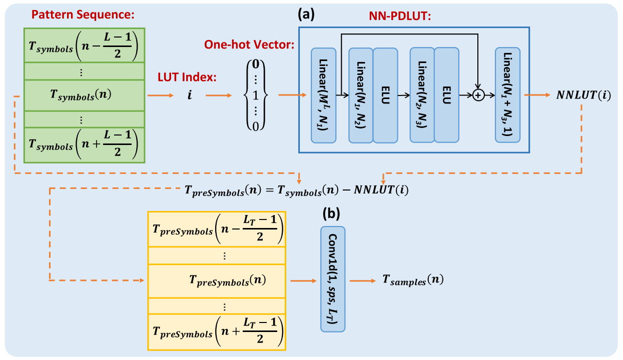


Fig. 5. Structure of the TxNN, including two parts: (a) NN-PDLUT and (b) single convolutional layer.

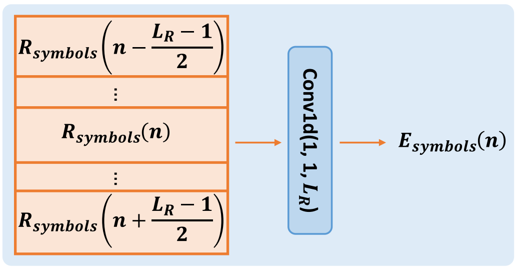

$R_{symbols}\left(n-\frac{L_R-1}{2}\right)$
⋮
$R_{symbols}(n)$
⋮
$R_{symbols}\left(n+\frac{L_R-1}{2}\right)$
Conv1d(1, 1, L_R)
$E_{symbols}(n)$

Fig. 6. Structure of the RxNN.

NN-PDLUT's output, NNLUT(i), is the LUT value corresponding to the LUT index i. The preprocessed symbols $\boldsymbol{T}_{\text{preSymbols}}$ are implemented as follows:

$$\boldsymbol{T}_{\text{preSymbols}}(n)=\boldsymbol{T}_{\text{symbols}}(n)-\text{NNLUT}(i). \tag{3}$$

For the second part in the TxNN, it uses a sliding window with the length of L_T to separate preprocessed symbols into the input sequence $\left[\boldsymbol{T}_{\text{preSymbols}}\left(n-\frac{L_T-1}{2}\right),\ldots,\boldsymbol{T}_{\text{preSymbols}}(n),\ldots,\boldsymbol{T}_{\text{preSymbols}}\left(n+\frac{L_T-1}{2}\right)\right]$, and the corresponding output of the single convolutional layer is $\boldsymbol{T}_{\text{samples}}(n)\in\mathbb{R}^{\text{sps}}$.

At the receiver, the RxNN R_w is responsible for recovering received signals into original transmitted symbols. Its structure is shown in Fig. 6, with only one convolutional layer, where w denotes the weights and bias of the TxNN. The received signal samples $\boldsymbol{R}_{\text{samples}}$ are first resampled to sps = 1 to obtain received symbols $\boldsymbol{R}_{\text{symbols}}$. Similarly, a sliding window with the length of L_R is used to separate the received symbols $\boldsymbol{R}_{\text{symbols}}$ into the input sequence $\left[\boldsymbol{R}_{\text{symbols}}\left(n-\frac{L_R-1}{2}\right),\ldots,\boldsymbol{R}_{\text{symbols}}(n),\ldots,\boldsymbol{R}_{\text{symbols}}\left(n+\frac{L_R-1}{2}\right)\right]$, and the corresponding equalized output is $\boldsymbol{E}_{\text{symbols}}(n)\in\mathbb{R}$.

D. End-to-End Training Process

After constructing the channel network, transmitter network, and receiver network, they are trained to minimize the specified channel modeling loss function and end-to-end loss function via a gradient-based algorithm. In this work, the channel modeling loss function is shown in Eq. (2), where end-to-end loss function Loss_e consists of two parts, as shown in Fig. 1. Loss_e can be expressed as follows:

$$\text{Loss}_e=\text{Loss}_1+\text{Loss}_2, \tag{4}$$

where Loss_1 is the MSE loss between transmitted symbols $\boldsymbol{T}_{\text{symbols}}$ and equalized output $\boldsymbol{E}_{\text{symbols}}$. It is defined as

$$\text{Loss}_1=\frac{1}{L_{\text{sym}}}\sum_{n=1}^{L_{\text{sym}}}\|\boldsymbol{T}_{\text{symbols}}(n)-\boldsymbol{E}_{\text{symbols}}(n)\|^2. \tag{5}$$

Loss_2 is the MSE loss between $\boldsymbol{T}_{\text{symbols}}$ and $\hat{\boldsymbol{R}}_{\text{samples}}$. It is defined as

$$\text{Loss}_2=\frac{1}{L_{\text{sym}}}\sum_{n=1}^{L_{\text{sym}}}\|\boldsymbol{T}_{\text{symbols}}(n)-\hat{\boldsymbol{R}}_{\text{symbols}}(n)\|^2, \tag{6}$$

Algorithm 1. Low-Complexity End-to-End Learning

```
1:  **Initialization**:
2:  Randomly initialize the noise adaptation channel $C_{b,s}$,
    the TxNN $T_\theta$, the RxNN $R_w$, and the memory buffer.
3:  **Pretraining**:
4:  Pretraining $C_{b,s}$ with historical $T_{\text{samples}}$ and $R_{\text{samples}}$
5:  **Main loop**:
6:  for $i = 1$: MainEpoch
7:    **Inner loop-1**:
8:    for $j = 1$: ChEpoch
9:      Sample $T_{\text{samples}}$ from a uniform distribution of $[-1, 1]$
        and transmit;
10:     Save $T_{\text{samples}}$ and received signal $R_{\text{samples}}$ to the
        memory buffer;
11:     Sample $N$ sets of $T_{\text{samples}}$ and $R_{\text{samples}}$ from the
        memory buffer to construct the training data set;
12:     Update $C_{b,s}$ according to Eq. (2).
13:   end for
14:   **Inner loop-2**:
15:   for $k = 1$: TREpoch
16:       Keep $C_{b,s}$ fixed;
17:       Send random bit sequence, construct the
          training data set with $T_{\text{symbols}}$, $R_{\text{samples}}$ and $E_{\text{symbols}}$.
18:       Update $T_\theta$ and $R_w$ by minimizing the end-to-end loss
          function defined as Eq. (4).
19:   end for
20: end for
```

$$\hat{\boldsymbol{R}}_{\text{symbols}}(n)=\frac{1}{\text{sps}}\sum_{i=1}^{\text{sps}}\hat{\boldsymbol{R}}_{\text{samples}}(n,i), \tag{7}$$

where L_{sym} is the size of $\boldsymbol{T}_{\text{symbols}}$.

Loss_2 is a constraint loss, similar to the constraints imposed on the target solution in the optimization area. Loss_2 forces the TxNN to take on more signal-processing tasks in end-to-end learning, thus ensuring that the RxNN can use a simple FFE structure, which is more in line with the DSP complexity requirements of PON.

Finally, the detailed training process of proposed end-to-end learning is summarized in Algorithm 1. It consists of initialization, pretraining, and the main loop.

In the main loop, the noise adaptation channel is first trained in inner loop-1; inner loop-1 is executed ChEpoch times. Then, the TxNN and the RxNN are trained in inner loop-2 TREpoch times. The two inner loops are repeated until the loss functions converge. Finally, to compensate for the difference between the modeled channel and the real physical channel for better end-to-end learning performance, the RxNN is fine-tuned with the transmitted and received signals from the experimental system.

3. EXPERIMENTAL SETUP

To validate the effectiveness of the proposed end-to-end learning framework, we conduct a comparison test on the downstream transmission of a 100G IMDD-PON system. These schemes include receiver-only equalization, PDLUT-based joint transceiver equalization, and end-to-end learning.

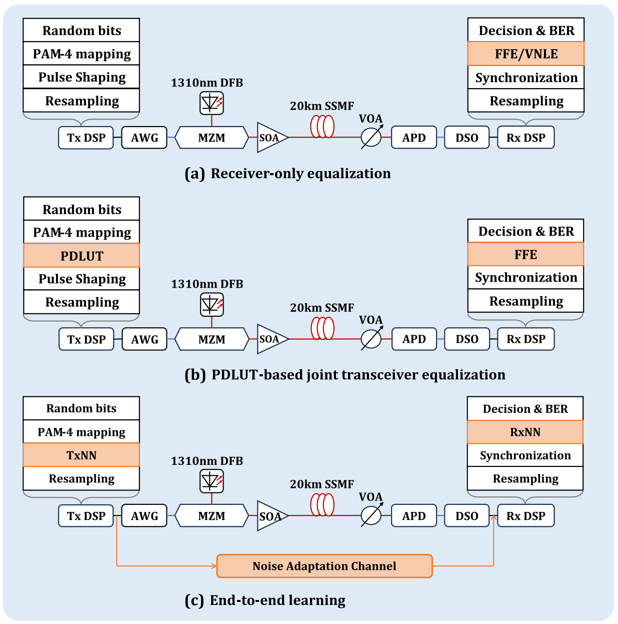


Fig. 7. Experimental schematics of 100G IMDD-PON for the three schemes: (a) receiver-only equalization, (b) PDLUT-based joint transceiver equalization, and (c) end-to-end learning.

The corresponding experimental schematics are shown in Fig. 7.

In Fig. 7(a), at the transmitter, random bits are first mapped to PAM-4 symbols with a symbol rate of 100 Gb/s; then, a root raised cosine (RRC) filter with the roll-off factor of 0.1 is used for pulse shaping. After that, the digital signal is resampled to 120 GSa/s before being loaded into the Keysight M8194A arbitrary waveform generator (AWG) with the sampling rate of 120 GSa/s. The AWG's output signal is amplified by a 23 dB electrical amplifier (EA) and modulated by a 10 GHz MZM biased at its quadrature point; further, a 1310 nm distributed feedback (DFB) laser is used for the light source. Optical launch power is controlled by a semiconductor optical amplifier (SOA) to 12 dBm, and the signal is transmitted over 20 km standard single mode fiber (SSMF). At the receiver, the received optical power (ROP) is adjusted by a variable optical attenuator (VOA); then, the signal is detected by a 30 GHz avalanche photodiode (APD) and extracted by a Tektronix digital storage oscilloscope (DSO) with a 33 GHz bandwidth and 100 GSa/s sampling rate. Finally, the offline DSP is executed, including resampling (to 1 sample per symbol), synchronization, FFE or VNLE equalization, symbol decision, and BER statistic.

The FFE can be expressed as Eq. (8), where $r(n)$ is the nth received signal, $o(n)$ is the corresponding output, and $w(l)$ is the lth tap value, and n_{taps} is the number of taps:

$$o(n) = \sum_{l=-\frac{n_{\text{taps}}-1}{2}}^{\frac{n_{\text{taps}}-1}{2}} w(l) r(n-l). \tag{8}$$

Third-order VNLE can be expressed as Eq. (9), where k_1, k_2, and k_3 are the first, second, and third-order memory length, w_i for $i = \{1, 2, 3\}$ is the ith order Volterra kernel:

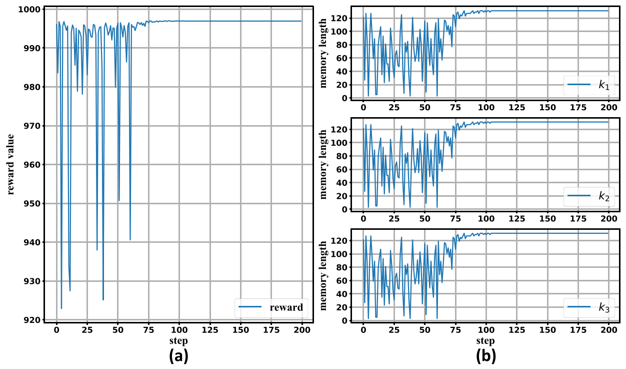


Fig. 8. Convergence results for (a) the reward value and (b) the first-, second-, and third-order memory length during the optimization for third-order VNLE.

$$\begin{aligned} o(n) = & \sum_{l_1=-\frac{k_1-1}{2}}^{\frac{k_1-1}{2}} w_1(l_1) r(n-l_1) \\ & + \sum_{l_1=-\frac{k_2-1}{2}}^{\frac{k_2-1}{2}} \sum_{l_2=l_1}^{\frac{k_2-1}{2}} w_2(l_1, l_2) r(n-l_1) r(n-l_2) \\ & + \sum_{l_1=-\frac{k_3-1}{2}}^{\frac{k_3-1}{2}} \sum_{l_2=l_1}^{\frac{k_3-1}{2}} \sum_{l_3=l_2}^{\frac{k_3-1}{2}} w_3(l_1, l_2, l_3) r(n-l_1) \\ & \times r(n-l_2) r(n-l_3) \end{aligned} \tag{9}$$

The FFE and VNLE are trained by the least mean square (LMS) algorithm. For VNLE, we use an automatic optimization algorithm [9] to obtain its first-order memory length k_1, second-order memory length k_2, and third-order memory length k_3. In the optimization process, the reward value (i.e., the optimization objective) is defined as $1000*(1-\text{BER})$. With the MAC operation limit of 1000 under ROP $= -14.5$ dBm, the convergence results for the reward value and memory lengths with the number of optimization steps are as shown in Figs. 8(a) and 8(b). Based on the convergence results, k_1, k_2, and k_3 are set to 131, 27, and 5. Further, n_{taps} of the FFE is set to 131.

In Fig. 7(b), the PDLUT is trained at the receiver and then placed at the transmitter. The PDLUT consists of table index i and table value LUT(i). To make the PDLUT, first, a sequence of PAM-4 symbols $t(n)$ is processed by pulse shaping and resampling. After transmission over the system link, the received signal undergoes feed-forward equalization to eliminate ISI, the equalized symbols are denoted as $y_{\text{FFE}}(n)$. Then, a sliding window with a memory length of L is used to separate $t(n)$ into the pattern-dependent symbol sequence $\left[T\left(n-\frac{L-1}{2}\right), \ldots, T(n), \ldots, T\left(n+\frac{L-1}{2}\right)\right]$; for PAM-4 symbols, there are $4^5 = 1024$ pattern sequence types when $L = 5$, which means table index $i = 0 \sim 1023$. Table value LUT(i) is obtained through Eqs. (10)–(13). $e(n)$ is the difference between $y_{\text{FFE}}(n)$ and $t(n)$, and $N(i)$ is the number of repeated patterns at index i.

Table 1. Network Parameters of the TxNN, the RxNN, and the Noise Adaptation Channel

	Parameter	Meaning	Value
TxNN	M	Modulation order	4
	L	Sliding window length of the NN-PDLUT	5
	N_1	Number of neurons in the linear layer	64
	N_2	Number of neurons in the linear layer	32
	N_3	Number of neurons in the linear layer	32
	L_T	Sliding window length	55
RxNN	L_R	Sliding window length	131
Noise adaptation channel	m	Size of the convolving kernel in the convolution layer	129
	sps	Samples per symbol	2
	K	Number of sub-networks in the MscaleDNN	5
	$\alpha_i, i = 1, \ldots, K$	Scale coefficients	2^{i-1}
	M_1	Number of channels produced by the convolution layer	128
	M_2	Number of neurons in the linear layer	64
	M_3	Number of neurons in the linear layer	32

Table 2. Training, Validation, and Test Parameters for End-to-End Learning

		Parameter	Value
Training parameter	Main loop	MainEpoch	50
	Inner loop-1	ChEpoch	2
		Number of bits sent	8192
		Learning rate	1e−4
		Batchsize	256
		Number of samples from the memory buffer per ChEpoch	5 groups
	Inner loop-2	TREpoch	150
		Number of bits sent	4096
		Learning rate	5e−4
Validation parameter		Number of symbols used to fine-tune the RxNN	10,000
		Number of symbols used to calculate the valid BER	22,768
Test parameter		Number of symbols used to fine-tune the RxNN	10,000
		Number of symbols used to calculate the test BER	88,304

$$e(n) = y_{\mathrm{FFE}}(n) - t(n), \tag{10}$$

$$\mathrm{LUT}(i) = \mathrm{LUT}(i) + e(n), \tag{11}$$

$$N(i) = N(i) + 1, \tag{12}$$

$$\mathrm{LUT}(i) = \mathrm{LUT}(i)/N(i). \tag{13}$$

After completing the PDLUT, the transmitted symbols are precompensated by the PDLUT and then postcompensated by the retrained FFE.

Figure 7(c) demonstrates the end-to-end learning scheme; for the TxNN, the RxNN, and the noise adaptation channel, their network structures and training methods are described in Section 2. The parameters of the TxNN, the RxNN, and the noise adaptation channel are set according to Table 1, and the training, validation, and test parameters are shown in Table 2. The validation data set is built by randomly generating 32,768 PAM4 symbols at the end of each main epoch, where the first 10,000 symbols are used to fine-tune the RxNN to obtain the best performance (the compared classical equalizers are also fully trained using 10,000 symbols), and the remaining 22,768 symbols are used to compute the BER on the validation data set (i.e., valid BER).

Finally, all schemes use three sets of PAM4 symbols with a length of 32,768 for BER testing. Similarly, the first 10,000 symbols are used to fine-tune the RxNN; the remaining symbols are used to calculate the BER on the test data set for performance evaluation.

4. EXPERIMENTAL RESULTS

Figure 9 demonstrates the training process of the end-to-end learning scheme under ROP = −14.5 dBm. The variation of the channel modeling loss function in the channel pretraining stage is shown in Fig. 9(a), named the channel pretraining loss. In the online tracking training stage, the variation of the channel modeling loss function with the main epoch is shown in Fig. 9(b), named the channel online-training loss. Both the channel pretraining loss and the channel online-training loss correspond to the negative form of Eq. (2). By comparing Figs. 9(a) and 9(b), it can be seen that the loss value is below −1 when finishing the pretraining stage, but the value is higher than −1 at the beginning of online tracking training. This suggests that the channel model trained offline is overfitted and

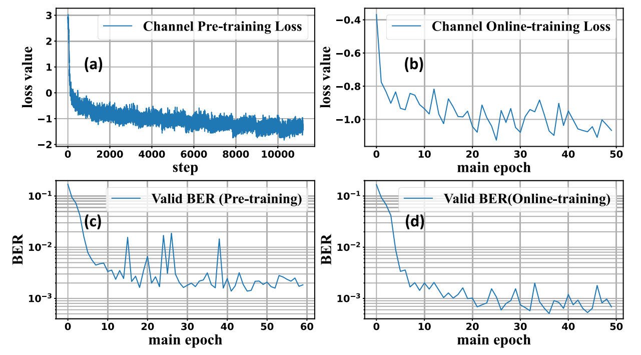


Fig. 9. (a) Variation of the channel modeling loss with the training step during the channel pretraining stage. (b) Variation of the channel modeling loss with the main epoch during the channel online-training stage. (c) End-to-end valid BER with the main epoch using only the pretrained channel network. (d) End-to-end valid BER with the main epoch using the online-trained channel network.

therefore needs to be tuned in conjunction with online tracking training. At the end of each main epoch, we validate the TxNN and the RxNN with 32,768 symbols, of which 10,000 symbols are used to fine-tune the RxNN, and the remaining 22,768 symbols are used to calculate the validation BER, named the valid BER. The variation of the valid BER with the main epoch using only a pretrained channel network is shown in Fig. 9(c), and the valid BER using the online-trained channel network is shown in Fig. 9(d). By comparison, it can be seen that the memory-buffer-assisted online channel training method can reduce the difference between the channel model and real physical channel, achieve better BER performance, and avoid retraining the channel model from scratch, saving training time. Finally, after finishing end-to-end training, the TxNN is kept fixed for different ROPs, and no retraining is required.

In Fig. 10, the power spectral density for the output signal of the noise adaptation channel is compared with that of the real physical channel. The noise adaptation channel models the entire frequency response of the transmitter device, fiber channel, and receiver device. It can be seen that the noise adaptation channel fits the real physical channel very well in the frequency range from 0 to 30 GHz, while the frequencies above 30 GHz lead to large differences in the power spectral density. This is because there are few useful signal components and mostly noise components, the power spectral density near 30 GHz is at least 20 dB lower than the low-frequency part. Besides, the flat power spectral density is due to the precompensation of the TxNN.

The BER comparison results of three schemes under different ROPs are shown in Fig. 11. RxFFE and RxVNLE are receiver-only equalization schemes, TxPDLUT_RxFFE is a PDLUT-based joint transceiver equalization scheme, and E2E means the end-to-end learning scheme (Code 1, Ref. [33]). In the end-to-end learning scheme, it is worth emphasizing that the experimental results corresponding to the other ROPs are all retrained directly from the pretrained models (noise adaptation channel and TxNN) obtained under ROP = −14.5 dBm. It can demonstrate the adaptability of the proposed framework and save a lot of training time. These schemes are represented

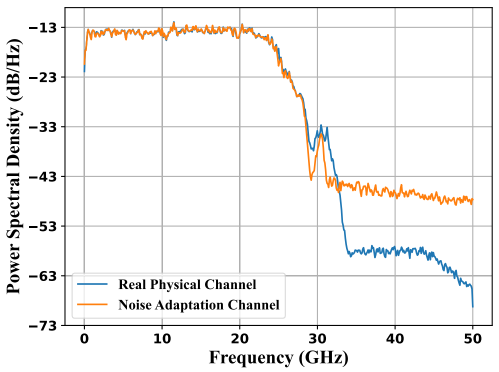


Fig. 10. Power spectral density for output signals of the noise adaptation channel and the real physical channel.

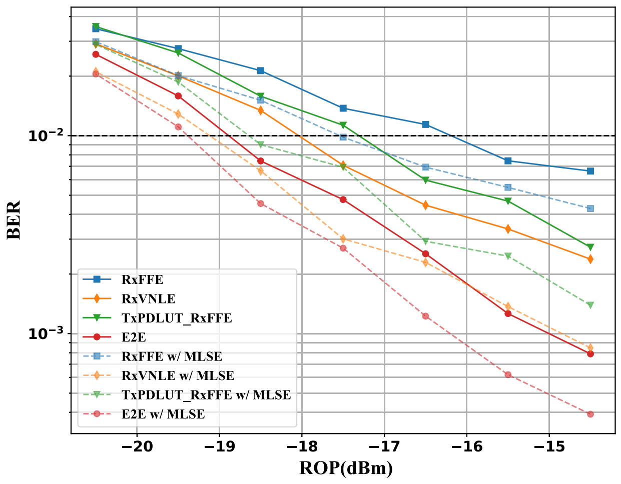


Fig. 11. BER comparison results of three schemes under different ROPs.

by four solid lines with different markers. Considering that maximum-likelihood sequence estimation (MLSE) equalization is the theoretically optimal equalizer in the presence of additive Gaussian white noise (AWGN), we add a whitening filter and MLSE to each scheme to demonstrate the optimal BER performance. The length of the estimated channel in MLSE is set to 4, and the BER results are represented by corresponding dotted lines.

As can be seen in Fig. 11, E2E outperforms RxVNLE, TxPDLUT_RxFFE, and RxFFE from the perspective of BER performance. Due to the joint optimization of the transmitter and receiver by end-to-end learning, suboptimal performance originating from isolated module designs can be avoided. According to the BER threshold of 1e-2, E2E allows 30.9 dB power budget, and it can increase the receiver sensitivity by 0.85 and 1.59 dB compared with RxVNLE and TxPDLUT_RxFFE. When these schemes are further processed by the MLSE equalizer, their BER performances are all improved. It can be found that the BER performance of E2E is similar to that of RxVNLE with MLSE. This shows that end-to-end learning is able to better compensate for linear and nonlinear impairments in the system.

Under ROP = −18.5 dBm, the statistical distributions of received PAM-4 signals obtained from the three schemes are

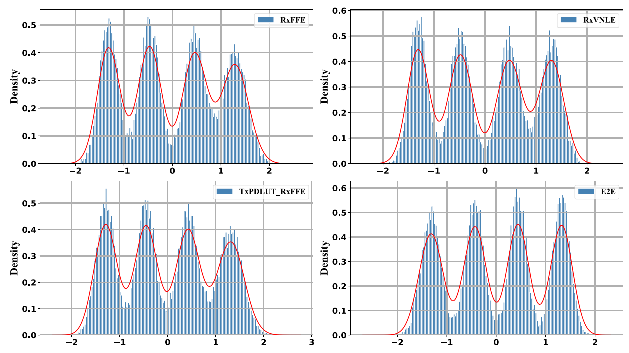
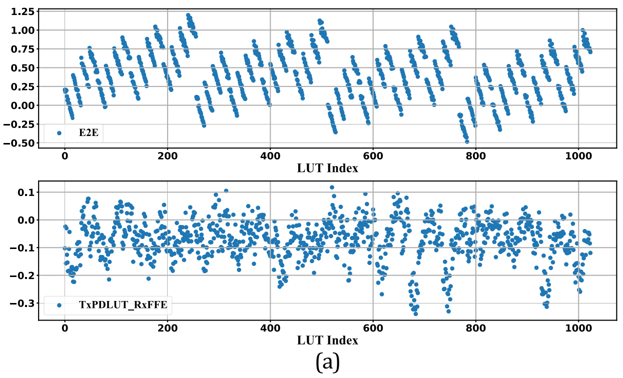


Fig. 12. Statistical distribution of received signals processed by different schemes under ROP = −18.5 dBm.



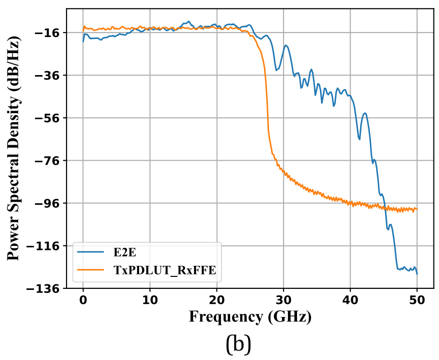


Fig. 13. (a) Comparison of look-up table values. (b) Comparison of transmitted signals in the end-to-end learning scheme and the PDLUT-based joint transceiver equalization scheme.

as shown in Fig. 12. It can be seen that the signal from the end-to-end learning scheme has clearer decision boundaries among different symbols, while the other schemes do not distinguish clearly between adjacent symbols, leading to poor BER performance (BER > 1e−2).

In the end-to-end learning scheme, the TxNN contains an NN-PDLUT. Therefore, we compare the training results of the NN-PDLUT with the PDLUT in the PDLUT-based joint transceiver equalization scheme. Figure 13(a) illustrates the lookup table indexes and values learned by the two schemes. Since the memory length L of the PDLUT is 5, for PAM-4 modulation, there are 1024 indexes. It can be noticed that the table values obtained by the two schemes are very different. This is because a traditional PDLUT is obtained by averaging the deviations of the input and output of a particular pattern sequence, whereas the NN-PDLUT in the TxNN learns the rule that handles these deviations better (no longer do averaging like a traditional PDLUT) in an end-to-end learning process. Moreover, the learned features of the NN-PDLUT receive the influence of a 1D convolutional layer in the TxNN during the joint training. Figure 13(b) presents the power spectral density of the signals generated at the transmitter in the two schemes. In the end-to-end learning scheme, the TxNN contains a convolutional layer to compensate for linear impairments. It can be seen that the high-frequency component of the signal is lifted, and the low-frequency component is depressed for compensating for the system bandwidth limitation. The spectrum of the precompensated signal becomes flat after transmission through the experimental system, as shown in Fig. 10.

In the PON system, in addition to the power budget, the complexity is also an important consideration. In hardware design, the digital signal processor (mainly used to implement multiplication, addition, and MAC operations) and the LUT are two are two major computing resources. Therefore, we compare the MAC and LUT consumption of the three schemes. For an FFE with n_{taps} taps, it needs to consume n_{taps} MAC operations. For a third-order VNLE with memory length $[k_1, k_2, k_3]$, the number of MAC operations it requires can be calculated using Eq. (14):

$$\text{MAC}_{\text{VNLE}} = N_1 + 2N_2 + 3N_3, \tag{14}$$

where $N_1 = k_1$, $N_2 = k_2(k_2 + 1)/2$, and $N_3 = k_3(k_3 + 1)(k_3 + 2)/6$.

Figure 14 illustrates the number of computing resources required for the three schemes. In the RxVNLE scheme, k_1, k_2, and k_3 are set to 131, 27, and 5; thus, it needs to consume a total of 991 MAC operations. In the TxPDLUT_RxFFE scheme, the size of the PDLUT is 1024 and the number of FFE taps is 131, so it needs to consume 131 MAC operations and 1024 LUTs. In the E2E scheme, the size of the NN-PDLUT is 1024 and the size of the convolutional layer is 110 in the TxNN; the size of the convolutional layer is 131 in the RxNN, so it needs to consume 131 MAC operations and 241 LUTs. Compared with RxVNLE, MAC consumption in E2E is reduced by 75.7%. Although a large amount of LUT resources

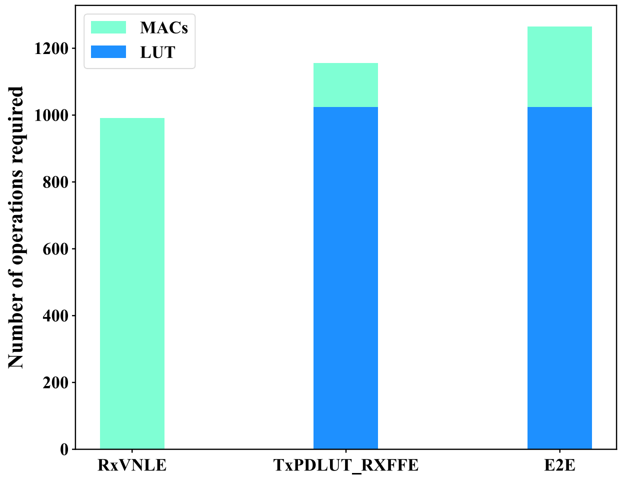


Fig. 14. Comparison of computing resource requirements for the three schemes.

are used, digital signal processor resources are even more valuable in the hardware design. Therefore, reducing the MAC operations in a scheme is helpful and beneficial to hardware implementation.

5. CONCLUSION

In this paper, a high-performance and low-complexity end-to-end learning framework is proposed and experimentally validated in a 100 Gbps IMDD-PON system. Within the framework, a noise adaptation network is designed to model the channel response and system noise distribution. Offline pretraining and online tracking training methods are proposed to adapt to different channel conditions and slow changes in channel states, thus enabling the noise adaptation network to accurately simulate the real channel. To achieve a low-complexity framework, only look-up tables and linear processing are used. At the transmitter, we use an NN-PDLUT and a single convolutional layer to compensate for nonlinear and linear impairments in the system. At the receiver, a single convolutional layer is used to compensate for residual linear impairments. The experimental results show that the end-to-end learning framework improves the receiver sensitivity by 0.85 and 1.59 dB compared with receiver-only equalization based on VNLE and joint equalization based on a PDLUT and an FFE, respectively. Moreover, the number of MAC operations consumed by the transmitter and receiver in end-to-end learning is reduced by 75.7% compared with VNLE-based receiver-only equalization. This indicates that the proposed end-to-end learning framework has significant advantages in performance and complexity. Through end-to-end learning, we can explore the optimal performance of a system and assist the design of algorithms for the transmitter and receiver. Further, deploying end-to-end learning frameworks in practical systems is a challenge. As the end-to-end learning framework requires training a channel network, it needs to obtain data from the experimental system. Therefore, we not only need to consider adding a computing module such as a graphics processing unit in the OLT but also need to design a data-exchange protocol between the OLT and ONU as well as determine the length of the time-domain pilot.

Funding. National Natural Science Foundation of China (62025503).

REFERENCES

1. R. Bonk, D. Geng, D. Khotimsky, *et al.*, "50G-PON: the first ITU-T higher-speed PON system," IEEE Commun. Mag. **60**(3), 48–54 (2022).
2. "IEEE standard for Ethernet amendment 9: physical layer specifications and management parameters for 25 Gb/s and 50 Gb/s passive optical networks," IEEE Std 802.3ca-2020 (2020).
3. "PON transmission technologies above 50 Gb/s per wavelength," ITU-T Recommendation G.suppl.VHSP (2023).
4. R. Bonk, E. Harstead, R. Borkowski, *et al.*, "Perspectives on and the road towards 100 Gb/s TDM PON with intensity-modulation and direct-detection," J. Opt. Commun. Netw. **15**, 518–526 (2023).
5. P. Torres-Ferrera, H. Wang, V. Ferrero, *et al.*, "100 Gbps/λ PON downstream O- and C-band alternatives using direct-detection and linear-impairment equalization [Invited]," J. Opt. Commun. Netw. **13**, A111–A123 (2021).
6. P. Torres-Ferrera, F. Effenberger, M. S. Faruk, *et al.*, "Overview of high-speed TDM-PON beyond 50 Gbps per wavelength using digital signal processing [Invited Tutorial]," J. Opt. Commun. Netw. **14**, 982–996 (2022).
7. L. Yi, T. Liao, L. Huang, *et al.*, "Machine learning for 100 Gb/s/λ passive optical network," J. Lightwave Technol. **37**, 1621–1630 (2019).
8. Q. Fan, G. Zhou, T. Gui, *et al.*, "Advancing theoretical understanding and practical performance of signal processing for nonlinear optical communications through machine learning," Nat. Commun. **11**, 3694 (2020).
9. Y. Xu, L. Huang, W. Jiang, *et al.*, "Automatic optimization of Volterra equalizer with deep reinforcement learning for intensity-modulated direct-detection optical communications," J. Lightwave Technol. **40**, 5395–5406 (2022).
10. H. Yang, Z. Niu, S. Xiao, *et al.*, "Fast and accurate optical fiber channel modeling using generative adversarial network," J. Lightwave Technol. **39**, 1322–1333 (2021).
11. D. Wang, X. Jiang, Y. Song, *et al.*, "Applications of physics-informed neural network for optical fiber communications," IEEE Commun. Mag. **60**(9), 32–37 (2022).
12. J. Xiang, Y. Cheng, S. Chen, *et al.*, "Knowledge distillation technique enabled hardware efficient OSNR monitoring from directly detected PDM-QAM signals," J. Opt. Commun. Netw. **14**, 916–923 (2022).
13. B. Karanov, M. Chagnon, F. Thouin, *et al.*, "End-to-end deep learning of optical fiber communications," J. Lightwave Technol. **36**, 4843–4855 (2018).
14. S. Gaiarin, F. Da Ros, R. T. Jones, *et al.*, "End-to-end optimization of coherent optical communications over the split-step Fourier method guided by the nonlinear Fourier transform theory," J. Lightwave Technol. **39**, 418–428 (2021).
15. V. Neskorniuk, A. Carnio, V. Bajaj, *et al.*, "End-to-end deep learning of long-haul coherent optical fiber communications via regular perturbation model," in *European Conference on Optical Communication (ECOC)* (2021).
16. V. Neskorniuk, A. Carnio, D. Marsella, *et al.*, "Memory-aware end-to-end learning of channel distortions in optical coherent communications," Opt. Express **31**, 1–20 (2023).
17. M. Srinivasan, J. Song, A. Grabowski, *et al.*, "End-to-end learning for VCSEL-based optical interconnects: state-of-the-art, challenges, and opportunities," J. Lightwave Technol. **41**, 3261–3277 (2023).
18. Z. Niu, H. Yang, H. Zhao, *et al.*, "End-to-end deep learning for long-haul fiber transmission using differentiable surrogate channel," J. Lightwave Technol. **40**, 2807–2822 (2022).
19. M. Li, D. Wang, Q. Cui, *et al.*, "End-to-end learning for optical fiber communication with data-driven channel model," in *Opto-Electronics and Communications Conference (OECC)* (2020).
20. B. Karanov, V. Oliari, M. Chagnon, *et al.*, "End-to-end learning in optical fiber communications: experimental demonstration and future trends," in *European Conference on Optical Communication (ECOC)* (2020).
21. B. Karanov, M. Chagnon, V. Aref, *et al.*, "Concept and experimental demonstration of optical IM/DD end-to-end system optimization using a generative model," in *Optical Fiber Communication Conference (OFC)* (2020), paper Th2A.48.
22. M. Li and S. Wang, "End-to-end learning for chromatic dispersion compensation in optical fiber communication," IEEE Commun. Lett. **26**, 1829–1832 (2022).
23. S. Xing, Z. Li, C. Huang, *et al.*, "End-to-end deep learning for a flexible coherent PON with user-specific constellation optimization," J. Opt. Commun. Netw. **16**, 59–70 (2024).
24. Q. Liu, M. Fu, X. Liu, *et al.*, "End-to-end joint digital and optical signal processing enabled by interpretable deep learning for coherent optical communication systems," in *Optical Fiber Communication Conference (OFC)* (2024), paper Th4B.1.
25. F. A. Aoudia and J. Hoydis, "Model-free training of end-to-end communication systems," IEEE J. Sel. Areas Commun. **37**, 2503–2516 (2019).
26. O. Jovanovic, F. Da Ros, D. Zibar, *et al.*, "Geometric constellation shaping for fiber-optic channels via end-to-end learning," J. Lightwave Technol. **41**, 3726–3736 (2023).

27. Z. Liu, W. Cai, and Z. Xu, "Multi-scale deep neural network (MscaleDNN) for solving Poisson-Boltzmann equation in complex domains," Commun. Comput. Phys. **28**, 1970–2001 (2020).
28. Z. Xu and H. Zhou, "Deep frequency principle towards understanding why deeper learning is faster," in *AAAI Conference on Artificial Intelligence* (2021), Vol. **35**, pp. 10541–10550.
29. Y. Tu, M. Xiang, W. Cheng, *et al.*, "Simplified look-up table (LUT) based digital pre-distortion for 100 GBaud/λ PAM-4 transmission," J. Lightwave Technol. **42**, 158–165 (2024).
30. J. Ke, Z. Wang, T. Yang, *et al.*, "Low-power multi-step PDLUT implementation for transmitter nonlinearity compensation," in *Opto-Electronics and Communications Conference (OECC)* (2023).
31. Y. Cai, H. Chien, M. Xiang, *et al.*, "Combined symbol-pattern-dependent adaptive equalization and sequence detection for coherent optical fiber communications," J. Lightwave Technol. **40**, 1320–1329 (2022).
32. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, 2016).
33. Y. Xu, X. Guan, W. Jiang, *et al.*, "Test code," figshare (2024), https://doi.org/10.6084/m9.figshare.26014828.