Computational complexity comparison of feedforward/radial basis function/recurrent neural network-based equalizer for a 50-Gb/s PAM4 direct-detection optical link

ZHHAOPENG Xu,1,1* CHUANBOWEN Sun,1 TONGHUI Ji,2 JONATHAN H. MANTON,1 AND WILLIAM SHIEH1

1Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, VIC 3010, Australia
2School of Computer Science and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
1zhaopeng@student.unimelb.edu.au

Abstract: The computational complexity and system bit-error-rate (BER) performance of four types of neural-network-based nonlinear equalizers are analyzed for a 50-Gb/s pulse amplitude modulation (PAM)-4 direct-detection (DD) optical link. The four types are feedforward neural networks (F-NN), radial basis function neural networks (RBF-NN), auto-regressive recurrent neural networks (AR-RNN) and layer-recurrent neural networks (L-RNN). Numerical results show that, for a fixed BER threshold, the AR-RNN-based equalizers have the lowest computational complexity. Amongst all the nonlinear NN-based equalizers with the same number of inputs and hidden neurons, F-NN-based equalizers have the lowest computational complexity while the AR-RNN-based equalizers exhibit the best BER performance. Compared with F-NN or RNN, RBF-NN tends to require more hidden neurons with the increase of the number of inputs, making it not suitable for long fiber transmission distance. We also demonstrate that only a few tens of multiplications per symbol are needed for NN-based equalizers to guarantee a good BER performance. This relatively low computational complexity signifies that various NN-based equalizers can be potentially implemented in real time. More broadly, this paper provides guidelines for selecting a suitable NN-based equalizer based on BER and computational complexity requirements.

© 2019 Optical Society of America under the terms of the OSA Open Access Publishing Agreement

1. Introduction

This paper investigates the computational complexity and system performance of different neural-network-based nonlinear equalizers to determine their potential for real-time implementation in a 50-Gb/s 20-km pulse amplitude modulation (PAM)-4 direct-detection (DD) system. Motivation for this study comes from the exponential growth of Internet Protocol (IP) traffic which has led to a corresponding increased demand for capacity in data centers. According to the Cisco Global Cloud Index forecast, global data center traffic will reach 20.6 Zettabytes (ZB) per year by the end of 2021, up from 6.8 ZB per year in 2016 [1]. Inspired by this dramatic demand for capacity, data center interconnection has become a hot topic in both academia and industry [2,3]. For such short-reach applications, DD systems are preferred over coherent-detection systems due to its simplicity and low cost [4,5]. However, unlike in coherent-detection systems where all the linear optical impairments such as chromatic dispersion (CD) can be compensated for by digital signal processing (DSP), the square-law detection in DD converts all the linear channel effects into nonlinear ones, making it hard to fully compensate for the resulting signal distortion [6]. The nonlinear impairments in short-reach direct-detection systems can severely limit the achievable capacity and reach, hence it is crucial to compensate for the nonlinear effects in DD systems.

#383039 https://doi.org/10.1364/OE.27.036953
Journal © 2019
Received 12 Nov 2019; accepted 27 Nov 2019; published 4 Dec 2019
Many nonlinear compensation methods have been proposed for short-reach DD systems, such as the well-known Volterra-series-based nonlinear equalizer [7,8], and various neural network (NN)-based nonlinear equalizers [9–17]. Traditional equalization methods such as feedforward equalization (FFE) can also be applied for short-reach DD systems, however, their performance is quite limited compared with the nonlinear methods such as NNs [16]. In terms of Volterra-series-based nonlinear equalizers, the computational complexity is a big problem due to the large number of high-order Volterra terms, and the performance is also restricted by the polynomial model. With the recent advances in machine learning, many machine learning techniques have been employed and have become popular research tools in the field of optical fiber communication. For example, k-means clustering and Gaussian mixture models (GMM) have been used for constellation clustering [18,19]; support vector machines (SVM) have been used for nonlinear classification [20,21]; and different kinds of NNs have been used in areas such as modulation format identification (MFI) [22,23], optical performance monitoring (OPM) [24,25] and nonlinear equalization in optical fiber transmission systems [9–17]. The nonlinear activation function in each hidden layer allows NNs to approximate complex nonlinear functions quite well, making them suitable for dealing with the nonlinear distortions in short-reach DD systems. In [9–13], several different feedforward NNs (F-NN) are shown to be effective for various short-reach transmission schemes. A pruning method for reducing the computational complexity of NN-based nonlinear equalizers is proposed in [14]. In addition to traditional F-NN, other kinds of NNs which are popular in machine learning have also been considered for nonlinear equalization of short-reach optical transmission systems. For example, radial basis function NNs (RBF-NN) [15], recurrent NNs (RNN) [16], and convolutional NNs (CNN) [17] have all been employed and validated in different short-reach scenarios. However, the computational complexity of the nonlinear equalizer is a critical issue in optical fiber communications because the nonlinear compensation algorithm needs to be implemented in real-time at an extremely high symbol rate in the order of tens of Gigabaud. It is noted that to recover one symbol in real-time coherent transmission systems, the computational complexity of the receiver end DSP is of the order of tens of multiplications [26].

The computational complexity of a simple F-NN and a Volterra-series-based nonlinear equalizer for coherent transmission systems have been compared in [27]. It is shown that a F-NN based nonlinear equalizer involves lower computational complexity than a Volterra series-based equalizer for equivalent BER performance. We have analyzed the computational complexity of the F-NN-based equalizers in [28], and these equalizers tend to achieve better BER performance with wider receiver bandwidth. In this paper, we extend our previous work and consider not only F-NN, but also RBF-NN and RNN-based nonlinear equalizers for short-reach DD systems. Note that in addition to auto-regressive RNNs (AR-RNN) used in [16], this paper also considers layer-recurrent NNs (L-RNN) for the first time in short-reach applications. The computational complexity and bit-error-rate (BER) performance of these NN-based equalizers are analyzed and compared in a 50-Gb/s 20-km PAM4 DD system. Among all the NN-based equalizers, F-NN shows the lowest computational complexity, while AR-RNN tends to achieve the best system bit-error-rate (BER) performance. For a fixed BER, AR-RNN also exhibits the lowest computational complexity among all the NN-based equalizers. Compared with FNN, L-RNN or AR-RNN, RBF-NN requires more hidden neurons with the increase of the number of inputs, which hinders its application for longer fiber transmission distance. Assuming no other issues are brought into the system, the analysis for the 50-Gb/s 20-km PAM4 system here could also be applicable to 100-Gb/s 5-km systems or other similar systems. We find that the NN-based equalizers can all achieve BERs lower than the 7% hard-decision (FEC) threshold while using only a few tens of multiplications to recover each symbol, showing their potential for real-time implementation in DSP application specific integrated circuits (ASICs). In a nutshell, this paper gives a thorough analysis of the computational complexity of F-NN-, RBF-NN-, AR-RNN-, and
L-RNN-based nonlinear equalizers and provides guidelines for selecting a proper NN when both the system performance and the computational complexity are taken into account.

2. Computational complexity analysis of NN-based equalizers

2.1. F-NN based equalizer

The schematic of a 2-layer F-NN [29] is shown in Fig. 1. Note that we employed two important parameters in NN, namely weights and biases. Some references only introduce weights into NNs because biases can be treated as weights if one neuron is added in the previous layer. However, this is just a choice of notation which affects neither computational complexity analysis nor NN performance. As shown in Fig. 1, the \( i \)-th layer consists of \( n^i \) neurons, where \( i = 0, 1, 2 \) \( (n^2 = 1 \) in the Fig. 1). For the \( i \)-th layer, the weight matrix \( w^i \) is an \( n^i \times n^{i-1} \) matrix containing all the weights connected from the \((i-1)\)-th layer to the \( i \)-th layer, the bias matrix \( b^i \) is an \( n^i \times 1 \) matrix containing all the biases and \( f^i(\cdot) \) is the activation function. The selection of activation function \( f^i(\cdot) \) is important to achieve good equalization performance. The most commonly used activation functions are the sigmoid function, the hyperbolic tangent \( (\tanh) \) function, and the rectified linear unit \( (\text{ReLU}) \). These functions provide different types of nonlinearities and may fit for different problems.

For the PAM4 nonlinear equalization application, the reason why we only consider 2-layer NNs rather than deep NNs (NNs with more than 2 hidden layers) is that 1 hidden layer is enough to achieve efficient equalization and there is no need to explore deeper NNs. It has been proved that a 2-layer NN can uniformly approximate any continuous function to arbitrary accuracy, provided the NN has a sufficient large number of hidden units and a variety of activation functions [29]. Therefore, we will focus on the single-hidden-layer architecture due to its simplicity. Actually, the computational complexity analysis can be easily generalized to deep NNs, as well as other types of NNs such as RNN.

In terms of the number of output neurons, we find that the NN-based nonlinear equalizers in [9–15] all employed NNs with 4 outputs. This kind of NNs are treated as classifiers where one-hot code is usually used to represent the 4 levels of the PAM4 symbol. To further reduce the size of NNs used for short-reach PAM4 systems, one good choice is to simply employ 1 output, in other words, \( n^2 = 1 \) instead of \( n^2 = 4 \). NNs with only 1 output can be regarded as predictors where an estimated value of the transmitted symbol is given. We have tested that NNs with \( n^2 = 1 \) and \( n^2 = 4 \) achieve similar system performance and this is reasonable because we can imagine that it is easy for NN to simulate one hot encoding/decoding procedure for PAM4 signals by adjusting the NN size and parameters. What’s more, for AR-RNN which requires feedbacks from the output layer, NNs with 1 output are more flexible to operate than those with 4 outputs. Much more computational complexity can be reduced if several delays are connected from the output layer to the input layer instead of only one delay. In this paper, 1 output is used.
for all the NNs to reduce the computational complexity as much as possible without comprising system performance.

Employing received PAM4 symbols as inputs, the NN based equalization for 1 symbol recovery refers to one-time forward-propagation (FP) of F-NN. Assuming all the weights and biases are known through training, the one-time FP step for F-NN can be expressed as

$$y = f^2([w^2]f^1([w^1]X + b^1) + b^2),$$

(1)

where \(X\) is the input vector containing \(n[0]\) inputs and \(y\) is the output. For \(m\)-symbol recovery, we can simply stack \(m\) input vectors by column to form a \(n[0] \times m\) matrix. By vectorizing \(m\) input vectors into matrices, only one FP step is needed for symbol equalization and the number of multiplications needed to recover 1 symbol, denoted \(N_{\text{mul,F-NN}}\), remains the same. Considering the 2-layer F-NN with only 1 output, \(N_{\text{mul,F-NN}}\) is given by

$$N_{\text{mul,F-NN}} = (n[0] + 1)n[1].$$

(2)

When mean squared error (MSE) is adopted as the NN performance evaluation metric, only 1 multiplication is needed to calculate the cost function for the 1-output F-NN. In Eq. (2), we ignore this computational complexity because as a performance metric, the mean squared error (MSE) given by the cost function and the BER performance show similar trend. We also omit the computationally heavy training process (multiple back propagation (BP) steps), because it only needs to be performed once for a stationary transmission system. After the NN is trained, it can be used as a fixed equalizer without retraining.

### 2.2. RBF-NN based equalizer

The RBF-NN [30] and F-NN share the same architecture illustrated in Fig. 1. Unlike F-NN whose hidden neurons calculate the weighted sum of the input vectors, RBF-hidden neurons calculate the Euclidean distance between the weight vectors and the input vector. Therefore, it’s better to interpret the weight matrix as stacks of vectors, where each row represents one hidden neuron, rather than connection between the input layer and the hidden layer. To be more specific, the output of the hidden neurons \(H^{[1]} = [H_1^{[1]}, H_2^{[1]}, \cdots H_{n[1]}^{[1]}]^T\) can be calculated as

$$H_{i}^{[1]} = f^{[1]}(b_{i}^{[1]} \| (w_{i}^{[1]})^T - X \|), \quad i = 1, 2, \cdots n[1],$$

(3)

where \(w_{i}^{[1]}\) is the \(i\)-th row vector of the weight matrix \(w^{[1]}\) following the notation used for F-NN, and the commonly-used activation function for radial basis hidden neurons is \(f^{[1]}(x) = e^{-x^2}\). From Eq. (3) we can see that the hidden neurons with weight vectors quite different from the input vector have outputs near zero, while those with weight vectors close to the input vector have outputs near one. The biases \(b^{[1]} = [b_{1}^{[1]}, b_{2}^{[1]}, \cdots b_{n[1]}^{[1]}]^T\) allow the sensitivity of the hidden neurons to be adjusted. The output \(y\) can then be expressed as

$$y = f^2([w^2]H^{[1]} + b^2).$$

(4)

Compared with F-NN, RBF-NN normally requires more neurons to guarantee a good performance, and this is also verified in our simulation when we use RBF-NN based nonlinear equalizer to recover PAM4 signals. Furthermore, F-NN can contain many hidden layers (deep NN) while RBF-NN consist of only 2 layers. Same as F-NN, when RBF-NN has only one neuron in the output layer, the number of multiplications needed to recover one symbol \(N_{\text{mul,RBF-NN}}\) is given by

$$N_{\text{mul,RBF-NN}} = (n[0] + 2)n[1].$$

(5)

As shown in Eq. (5), the FP step for RBF-NN involves more computational complexity compared with F-NN when same number of input and hidden neurons are used. The complexity difference can be very small as we reduce the number of hidden neurons.
2.3. **AR-RNN based equalizer**

RNN is very useful in machine learning to deal with sequential data, where AR-RNN (Jordan network) [31] and L-RNN (Elman network) [32] are two types of RNN most widely used. The schematic of a 2-layer AR-RNN is shown in Fig. 2, in which we can observe it employs several delays of the output as feedback. The introduction of past predicted output values as additional inputs provides more information when predicting the current output value, so AR-RNN can usually perform better than F-NN or RBF-NN with additional feedback information, especially for sequential data set. Assuming \( k \) feedbacks are employed, the one-time FP step for AR-RNN can be expressed as

\[
y = f^{(2)} \left( w^{[2]} f^{(1)} \left( [w^{[1]}, w^d] \left[ \begin{array}{c} X \\ Y^d \end{array} \right] + b^{[1]} \right) + b^{[2]} \right),
\]

where \( w^d \) is an \( n^{[1]} \times k \) matrix containing all the weights connected from the hidden layer to the feedbacks and \( Y^d \) is an input vector containing \( k \) feedbacks. Compared with the FP step in F-NN, AR-RNN expands the weight matrix and input matrix because delays of the output is regarded as additional inputs. Therefore, the number of multiplications needed for AR-RNN to recover one symbol, denoted \( N_{mul,AR-RNN} \), is the same as that of F-NN with \( (n^{[0]} + k) \) inputs. \( N_{mul,AR-RNN} \) is given by

\[
N_{mul,AR-RNN} = (n^{[0]} + k + 1)n^{[1]}.
\]

![Fig. 2. Schematic of a 2-layer AR-RNN [31].](image)

The computational complexity of AR-RNN can be significantly affected by the number of feedbacks \( k \). When \( k \) is small, the computational complexity is comparable to F-NN or RBF-NN. When large \( k \) is used, AR-RNN can be computational costly.

2.4. **L-RNN based equalizer**

In addition to AR-RNN, L-RNN is also widely employed in many fields to process sequential data. The structure of 2-layer L-RNN is shown in Fig. 3, where layer delays rather than output feedbacks are used. In terms of simplicity and performance, L-RNN with a single delay is the most popular structure. With this architecture, L-RNN is able to learn some information from previous data when dealing with the current data. Assuming the hidden layer delays denoted by \( H^h \) (an \( n^{[1]} \times 1 \) vector), are connected to the hidden neurons with a \( n^{[1]} \times n^{[1]} \) weight matrix \( w^h \),
the one-time FP step for L-RNN can be shown as

$$y = f^{[2]}(w^{[2]}f^{[1]}([w^{[1]}, w^h] \begin{bmatrix} X \\ H^h \end{bmatrix} + b^{[1]})) + b^{[2]}.$$

(8)

Note here $n^{[1]}$ layer delays serve as additional inputs in L-RNN, while for AR-RNN, the number of additional inputs is $k$. Similar to the computational analysis for AR-RNN, the number of multiplications required to recover one symbol, denoted $N_{mul,L-RNN}$, can be expressed as

$$N_{mul,L-RNN} = (n^{[0]} + n^{[1]} + 1)n^{[1]}.$$

(9)

From Eqs. (7) and (9), we can see that when $n^{[1]}$ is greater than $k$, then L-RNN introduces higher computational complexity than AR-ANN and vice versa. Small $n^{[1]}$ is preferred in L-RNN to make a significant reduction in computational complexity.

3. System setup and results

3.1. System setup

A 50-Gb/s PAM4 direct detection optical link is simulated using VPItransmissionMaker 9.8 and the system setup is illustrated in Fig. 4. The electrical PAM4 signal after the digital-to-analog converter (DAC) is modulated by an electro-absorption modulated laser (EML) at 1550 nm with a 3-dB bandwidth of 12.5 GHz and a chirp factor of 0.1. The sample rate of DAC is 100 GSa/s. A root raised cosine (RRC) filter with the roll-off factor of 0.1 is used for pulse shaping. The PAM4 symbol is then transmitted via a 20-km fiber with the dispersion coefficient of 17 ps/nm/km. At the receiver side, a variable optical attenuator (VOA) is first used to adjust the received optical power (ROP) and then the optical signal is directly detected by a photodetector (PD) with the 3-dB bandwidth of 12.5 GHz. After an analog-to-digital converter (ADC) with the same sampling rate as the ADC, a matched RRC filter is used at Rx DSP, and then the filtered signal is downsampled for nonlinear equalization. After the NN-based equalizer, PAM4 decision is performed to recover the transmitted symbols.

The inputs of the NN-based nonlinear equalizers contain the current symbol, $(n^{[0]} - 1)/2$ past symbols, and $(n^{[0]} - 1)/2$ post symbols. For AR-RNN or L-RNN, $k$ output feedbacks or $n^{[1]}$ layer delays serve as additional inputs. The tanh function is used as the activation function of the input and the hidden layer, and a pure-linear function is employed for the output layer. 10000 PAM4 symbols are used to train different NNs and another 1 million PAM4 symbols are employed for nonlinear equalization and BER calculation. 80 epochs (number of iterations) are selected during the training process to achieve low MSE. More PAM4 symbols can be transmitted without
changing the current NN-based equalizer since the training process only needs to be performed once for a stationary channel. After the NN is trained, it can be used as a fixed equalizer for our PAM4 transmission system.

3.2. Results and discussions

The BER and $N_{\text{mul}}$ (number of multiplications needed to recover one symbol) contour map with various $n_0[0]$ and $n_1[1]$ is illustrated in Fig. 5, where Figs. 5(a) and 5(b) represent F-NN based equalizer, Figs. 5(c) and 5(d) represent RBF-NN based equalizer, Figs. 5(e) and 5(f) are for L-RNN based equalizer, Figs. 5(g) and 5(h) denote the results of AR-RNN based equalizer. The ROP is set as $-16$dBm. For AR-RNN, $k$ is optimized and set as 2.

As shown in Figs. 5(a), 5(e), and 5(g), BER decreases with the increase of $n_0[0]$ and $n_1[1]$ when F-NN, L-RNN, and AR-RNN are employed. However, BER floor exists when $n_0[0]$ and $n_1[1]$ is larger than a threshold. To be specific, no further BER improvement is observed when $n_0[0]$ is beyond 11 and $n_1[1]$ beyond 5. This is reasonable since 11 inputs have already contained enough information of the signal distortion and 5 hidden neurons are also sufficient for NN to perform nonlinear equalization. More inputs and hidden neurons may introduce unnecessary weights and biases, which contribute very little to the BER performance. The BER performance of RBF-NN which is shown in Fig. 5(c), however, shows different trend compared with other NNs. With the increase of $n_0[0]$, larger $n_1[1]$ is needed to guarantee a good performance. In other words, $n_1[1]$ should match $n_0[0]$ for the RBF architecture. Though $n_1[1]$ is relevant to $n_0[0]$, we found that instead of the huge number of hidden neurons used in [15], a small $n_1[1]$ can also work very well. When $n_0[0]$ is 11 and $n_1[1]$ beyond 7, RBF-NN achieves the best BER performance. We also observe that all the NN-based equalizers can achieve BER below the 7% hard-decision forward error correction (FEC) threshold, among which AR-RNN based equalizer tends to achieve the best performance.

Considering the same $n_0[0]$ and $n_1[1]$ used for all equalizers, the order of $N_{\text{mul}}$ is: L-RNN > AR-RNN > RBF-NN > F-NN, which is illustrated in Figs. 5(b), 5(d), 5(f) and 5(h). Note there is tradeoff between BER and computational complexity here. If the BER performance is not strongly required, we can simply use a F-NN based equalizer which introduces the least computational complexity, otherwise we would prefer to use AR-RNN which has the best system performance. Since more weights are related to the hidden layer, $n_1[1]$ contributes more to the computational complexity than $n_0[0]$ for all the NNs. The complexity of L-RNN is much more sensitive to $n_1[1]$ compared with other NNs because of its layer delay feedbacks. Therefore, in terms of computational complexity, we should choose $n_1[1]$ as small as possible, as long as the BER performance satisfies our needs. Considering the FEC threshold, all the NN-based equalizers only require few tens of multiplications, which shows the potential to be implemented in real-time ASICs. We can make a trade-off between the BER performance and the computational complexity by referring to the $N_{\text{mul}}$/BER contour maps and select a proper NN architecture as an efficient nonlinear equalizer.

Fig. 4. System setup of a 50-Gb/s PAM4 direct detection optical link.
Fig. 5. (a)/(b) BER/$N_{\text{mul}}$ contour map for F-NN based equalizer; (c), (d) RBF-NN based equalizer; (e), (f) L-RNN based equalizer; (g), (h) AR-RNN based equalizer.
The minimum computational complexity required for 4 types of NNs under different BER is also investigated and shown in Fig. 6(a). Here we try to find the lowest \( N_{\text{mul}} \) for all types of NN-based equalizers that could satisfy a fixed BER requirement when the ROP is \(-16\) dBm. For each fixed BER threshold, we conducted simulation and exhaustive search all possible \( n^0 \) and \( n^1 \) for each type of NN-based equalizers. Then the NNs with performance lower than the BER threshold are selected, of which we choose one with the least number of multiplications. When the BER requirement is low, only a few (about 20) multiplications are sufficient for all 4 types of NNs. When the BER requirement becomes higher, AR-RNN starts to show the lowest complexity and the best performance among all the NNs. Since AR-RNN shows much better performance than F-NN with the same size, \( n^0 \) and \( n^1 \) could be reduced to save computational complexity for a fixed BER threshold. However, for L-RNN, the limited BER performance improvement may not be neutralized by reducing \( n^0 \) or \( n^1 \), so we can observe an intersection of the F-NN and L-RNN in Fig. 6(a). To be more specific, both 20% soft-decision and 7% hard-decision FEC threshold are considered, and the minimum \( N_{\text{mul}} \) requirement for them is shown in Fig. 6(b). For BER performance under the 7% hard-decision FEC threshold (3.8 \times 10^{-3}), an AR-RNN with only 7 inputs and 3 hidden neurons can work well, which brings in the least computational complexity among all 4 types of NNs. RBF-NN requires the largest \( N_{\text{mul}} \) in this case, because \( n^1 \) needed for RBF-NN increases with \( n^0 \) to guarantee a good BER performance. For the 20% soft-decision FEC threshold (2.4 \times 10^{-2}), the \( N_{\text{mul}} \) needed for all 4 types of NN are around 20. No significant complexity difference can be observed because a much higher BER threshold is used here.

![Fig. 6.](image-url)

Figure 7(a) shows the BER performance versus ROP using different NN-based equalizers and Fig. 7(b) illustrates the \( N_{\text{mul}} \) needed for the 4 types of NNs with 2 different architectures. Here \((n^0, n^1)\) is used to represent the parameters of the 2-layer NNs. The BER performance of different NNs coincides with our previous analysis, which follows approximately AR-RNN > L-RNN > RBF-NN > F-NN. NNs with \( (11,7) \) (\( N_{\text{mul}} > 80 \)) are used to demonstrate good BER performance while NNs with \( (5,4) \) (\( N_{\text{mul}} < 40 \)) show acceptable performance as well as low computational complexity. We can find that all the NNs can achieve significant BER improvement compared with DD even with a very simple architecture. Considering the FEC threshold, receiver sensitivity of \(-13\) dBm can be achieved using F-NN with \( (5,4) \). 3.5 dB receiver sensitivity improvement can be observed if AR-RNN with \( (5,4) \) is employed. The \( N_{\text{mul}} \) needed for all the NNs with \( (5,4) \) is less than 40. If we consider a more complex NN architecture, e.g., \( (11,7) \), the receiver sensitivity can be further pushed to about \(-17.5\) dBm with the help of AR-RNN, where the \( N_{\text{mul}} \) needed is only 98.

The BER performance versus fiber length with different NN-based equalizers is shown in Fig. 8 when the ROP is \(-14\) dBm. Here, \( n^0 \) of 21 and \( n^1 \) of 9 are employed, since longer transmission
Fig. 7. (a) BER versus ROP using different NN-based equalizers, and (b) $N_{\text{mul}}$ of different NNs with (5,4) or (11,7).

distance introduces more severe signal distortion, and thus a larger NN architecture is needed. Compared with DD, NNs can support much longer fiber transmission. The maximum fiber length can be extended to 19, 22, 23, 27 km with the help of RBF-NN, F-NN, L-RNN, AR-RNN, respectively. We also notice that as the fiber length increases, there exists an intersection between RBF-NN and F-NN at a fiber length of around 17 km. When fiber length is beyond 18 km and $n_0$ is 21, 9 hidden neurons are not sufficient to match $n_0$ which leads to a poor BER performance. When the fiber length is 16 km, due to the weaker nonlinearity induced, the number of actual useful inputs can be much smaller than 21. Therefore 9 hidden neurons could match the number of actual useful inputs, and a good BER performance can then be achieved. RBF-NN is not suitable for long transmission distance, because longer fiber transmission needs more inputs to provide useful channel information, thus much more hidden neurons are required to match the inputs which brings in huge computational complexity.

Fig. 8. BER versus fiber length using different NN-based equalizers.

4. Conclusion

The computational complexity and BER performance of F-NN-, RBF-NN-, L-RNN-, and AR-RNN-based nonlinear equalizers are investigated in a 50-Gb/s PAM4 direct detection optical link. Among all the NN-based equalizers with the same number of inputs and hidden neurons, we find that AR-RNN can achieve the best system BER performance while F-NN shows the lowest computational complexity. Only a few tens of multiplications are needed for all the NN-based equalizers to achieve BER performances below the FEC threshold. This indicates the potential for practical NN-based equalizers to be implemented on DSP ASICs. Guidance is given on selecting an appropriate NN architecture for nonlinear equalization.
Funding

Australian Research Council (DP150101864, DP190103724).

Disclosures

The authors declare no conflicts of interest.

References


