Experimental Investigation of Deep Learning for Digital Signal Processing in Short Reach Optical Fiber Communications

(Invited Paper)

Boris Karanov*‡, Mathieu Chagnon‡, Vahid Aref‡, Filipe Ferreira*, Domaniç Lavery*, Polina Bayvel*, and Laurent Schmalen†

*Dept. Electronic & Electrical Engineering, University College London, WC1E 7JE London, U.K.
†Communications Engineering Lab, Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
‡Nokia Bell Labs, 70435 Stuttgart, Germany

***Abstract*—We investigate methods for experimental performance enhancement of auto-encoders based on a recurrent neural network (RNN) for communication over dispersive nonlinear channels. In particular, our focus is on the recently proposed sliding window bidirectional RNN (SBRNN) optical fiber auto-encoder. We show that adjusting the processing window in the sequence estimation algorithm at the receiver improves the reach of simple systems trained on a channel model and applied "as is" to the transmission link. Moreover, the collected experimental data was used to optimize the receiver neural network parameters, allowing to transmit 42 Gb/s with bit-error rate (BER) below the 6.7% hard-decision forward error correction threshold at distances up to 70 km as well as 84 Gb/s at 20 km. The investigation of digital signal processing (DSP) optimized on experimental data is extended to pulse amplitude modulation with receivers performing sliding window sequence estimation using a feed-forward or a recurrent neural network as well as classical nonlinear Volterra equalization. Our results show that, for fixed algorithm memory, the DSP based on deep learning achieves an improved BER performance, allowing to increase the reach of the system.**

***Index Terms*—Optical communications, digital signal processing, deep learning, neural networks, modulation, detection.**

I. Introduction

The application of machine learning in the design, networking and monitoring of communication systems has attracted great research interest in the recent years [1]–[4]. From a physical layer perspective, deep learning techniques [5], enabled by artificial neural networks (ANN), are considered a viable alternative to classical digital signal processing (DSP) [1], [4]. The combination of ANNs and deep learning can be used to optimize DSP functions at the transmitter or receiver. For example, deep learning was applied to the digital back-propagation block for non-linearity compensation in long-haul coherent optical fiber systems, reducing the computational demands [6]. Furthermore, an ANN was used as a receiver equalizer for low-cost fiber-optic communication links as found in passive access networks (PONs) [7]. In molecular communications, a novel detection scheme based on neural networks in combination with an efficient sliding window sequence estimation algorithm was proposed in [8]. The aforementioned applications of deep learning are examples where the techniques are within a specific DSP module. The signal processing for such communication systems is engineered and optimized module-by-module, each part being responsible for an individual task such as coding, modulation or equalization.

A different approach to deep learning-based DSP, which utilizes the full potential of the function approximation capabilities of artificial neural networks, is to interpret the complete communication chain from the transmitter input to the receiver output as a single deep ANN [9], [10] and optimize its parameters in an end-to-end process. Such fully learnable communication transceivers (also known as *auto-encoders*) partly avoid the modular design of conventional systems and have the potential to achieve an end-to-end optimized performance for a specific metric. Auto-encoders are especially suitable for communication over channels where the optimum transceiver is unknown or its implementation is computationally prohibitive. Dispersive nonlinear channels are an example, which is often encountered in optical fiber communications. This prompted the introduction of the auto-encoder concept for such systems [12]–[14]. In particular, end-to-end deep learning is seen as a viable DSP solution to enhance the system performance at reduced complexity in low-cost intensity modulation/direct detection (IM/DD) optical fiber links where the combination of chromatic dispersion, introducing inter-symbol interference (ISI), and nonlinear signal detection by a photodiode imposes severe limitations [11], [19]. The first experimental demonstration of an optical fiber auto-encoder showed that a simple feed-forward neural network (FFNN) design for IM/DD can outperform pulse amplitude modulation (PAM) with conventional, simple linear equalizers [12]. Addressing the limitations of the FFNN system in handling the channel memory, a transceiver based on a bidirectional recurrent ANN and sliding window sequence estimation (SBRNN) was proposed in [15] and experimentally verified in [16], achieving a substantial performance improvement. Moreover, the SBRNN

The work was funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie project COIN (No. 676448) and the UK EPSRC programme grant TRANSNET (EP/R035342/1).

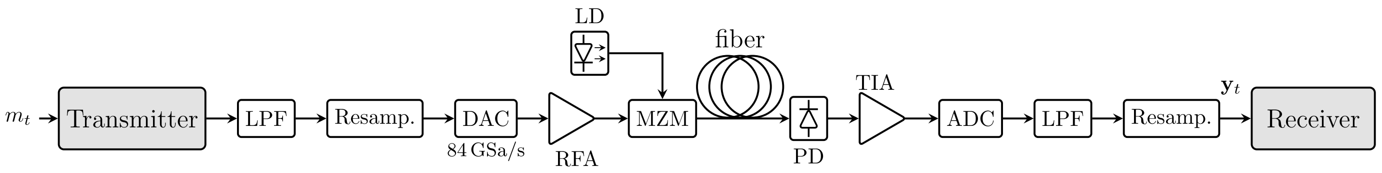

Fig. 1. Schematic of the experimental optical IM/DD transmission test-bed used for investigation of the digital signal processing schemes.

auto-encoder outperformed PAM transmission with state-of-the-art nonlinear receivers [15] or classical maximum likelihood sequence detection (MLSD) [17] without the associated computational complexity of such schemes.

In this paper, we extend the experimental study of the SBRNN auto-encoder by investigating methods to further enhance the performance it achieves when parameters are trained on a channel model and applied to the transmission link. A simple solution, which does not require the implementation of an additional training procedure, is to adjust the sliding window size in the estimation algorithm and thus capture more of the channel memory. Moreover, we show that the BER can be improved by initiating an optimization for the receiver parameters using the collection of experimental data. In this way, the performance penalty due to any mismatch between the simplified channel model and the actual transmission link can be reduced. We extend the investigation of DSP schemes using experimental data and deep learning by considering PAM systems with sliding window FFNN or SBRNN receivers and comparing them to a classical nonlinear Volterra equalizer. While the SBRNN auto-encoder had a superior performance, the SFFNN was shown as a viable receiver alternative for PAM systems with an improved BER over Volterra equalizers.

II. Experimental Setup and Data Collection

Due to its simplicity and cost-effectiveness, optical fiber transmission based on IM/DD is considered a front-runner technology for *fiber to the x* (FTTX) systems. The IM/DD links are also preferred for other short reach applications such as data center, metro and access networks [19]. In this section we describe the experimental IM/DD test-bed used to examine the performance of the DSP schemes and explain how the data-sets for parameter optimization and testing were formed.

A. Optical Fiber Transmission Link

Figure 1 shows a schematic of the test-bed. We consider an optically un-amplified transmission link. The output of the transmitter is filtered in the digital domain by a Brick-wall low-pass filter (LPF) which has a bandwidth of 32 GHz. Afterwards, the signal is appropriately re-sampled (module *Resamp.*) and applied to the digital-to-analog converter (DAC) which operates at 84 GSa/s. The obtained electrical waveform is pre-amplified by a driver amplifier and used to modulate the 1550 nm laser diode (LD) via a Mach-Zehnder modulator (MZM). For the auto-encoder system, the MZM is meticulously biased to match simulation, while for the PAM transmission, the bias is set at the quadrature point. The optical power input to the fiber is set to 1 dBm. The modulated optical signal is propagated over a fixed length of standard single mode fiber and the received waveform is direct-detected by a PIN photo-diode (PD) combined with a trans-impedance amplifier (TIA). The signal is real-time sampled by an analog-to-digital converter (ADC). After synchronization, proper scaling, offset and re-sampling, the digitized photo-current is stored for the subsequent signal processing at the receiver.

B. Data Collection for the Auto-encoder System

For the auto-encoder experiment we transmit $Z = 800$ different sequences of $T = 5152$ random input messages $m_{i,j} \in \mathcal{M}$ (with an alphabet size of $M = |\mathcal{M}| = 64$), with $i \in \{1, \ldots, Z\}$ and $j \in \{1, \ldots, T\}$. The data is generated using the Mersenne twister algorithm to avoid learning representations of a pseudo-random sequence during receiver optimization [20]. Each of the sequences is first encoded by the transmitter BRNN into series of blocks of n samples before being applied to the experimental test-bed, starting with the LPF. Note that we used an up-sampling factor of 4 in both transmitter and receiver. The value of n depends on the data rate. For 42 Gb/s or 84 Gb/s transmission, we select $n = 48$ or $n = 24$, respectively. Since the BRNN transmitter encodes a message carrying $\log_2 M = 6$ bits, the series of encoded blocks were down-sampled in experiment after the LPF such that 6 bits were carried by 12 or 6 DAC samples for 42 Gb/s or 84 Gb/s transmission, respectively. A full load of the DAC requires the concatenation of 8 encoded sequences and this is fed to the link for a single transmission iteration. To collect a sufficient amount of experimental data, we performed 100 DAC loads. For receiver optimization and testing we used the sequences of blocks $\mathbf{y}_{i,j}$, $i \in \{1, \ldots, Z\}$ and $j \in \{1, \ldots, T\}$, at the output of the test-bed together with their labels $m_{i,j}$. Sequences $i \in \{1, \ldots, 720\}$ formed the training sets

$$\mathbf{D}^{\text{AE}} := \begin{bmatrix} \mathbf{y}_{1,1\ldots T} & \mathbf{y}_{2,1\ldots T} & \mathbf{y}_{3,1\ldots T} & \mathbf{y}_{4,1\ldots T} \\ \mathbf{y}_{5,1\ldots T} & \mathbf{y}_{6,1\ldots T} & \mathbf{y}_{7,1\ldots T} & \mathbf{y}_{8,1\ldots T} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{y}_{717,1\ldots T} & \mathbf{y}_{718,1\ldots T} & \mathbf{y}_{719,1\ldots T} & \mathbf{y}_{720,1\ldots T} \end{bmatrix}$$

$$\mathbf{L}^{\text{AE}} := \begin{bmatrix} m_{1,1\ldots T} & m_{2,1\ldots T} & m_{3,1\ldots T} & m_{4,1\ldots T} \\ m_{5,1\ldots T} & m_{6,1\ldots T} & m_{7,1\ldots T} & m_{8,1\ldots T} \\ \vdots & \vdots & \vdots & \vdots \\ m_{717,1\ldots T} & m_{718,1\ldots T} & m_{719,1\ldots T} & m_{720,1\ldots T} \end{bmatrix},$$

where $\mathbf{D}^{\text{AE}} \in \mathbb{R}^{180 \times 4 \cdot T \cdot n}$ consists of the data elements $\mathbf{y}_{i,j}$ and $\mathbf{L}^{\text{AE}} \in \mathbb{R}^{180 \times 4 \cdot T}$ contains the label elements $m_{i,j}$. During the training procedure (see Sec. III-A3) we pick a window of V columns to form the mini-batch on each stochastic gradient descent (SGD) optimization step. To ensure a sufficiently large mini-batch of independent examples, when forming the data-sets, we split one full DAC load into two parts, treated independently by the learning algorithm, e.g. $\mathbf{y}_{1,1\ldots T}\mathbf{y}_{2,1\ldots T}\mathbf{y}_{3,1\ldots T}\mathbf{y}_{4,1\ldots T}$ and $\mathbf{y}_{5,1\ldots T}\mathbf{y}_{6,1\ldots T}\mathbf{y}_{7,1\ldots T}\mathbf{y}_{8,1\ldots T}$. Note that this results in only negligible performance degradation since $4T \gg V$. We verified that convergence of the loss is achieved within one epoch of the training data, which we then used as a stopping criterion. The remaining sequences with $i \in \{721, \ldots, Z\}$, indepent from the training set are used in the testing phase.

C. Data Collection for the PAM Systems

We consider the conventional two- and four-level PAM and investigate deep learning approaches for detection based on recurrent and feed-forward neural networks as well as nonlinear Volterra equalization. At the transmitter we use a Mersenne twister to generate PAM2 or PAM4 symbols which we accordingly scaled to $\mathcal{M} = \{0; \pi/4\}$ or $\mathcal{M} = \{0; \pi/12; \pi/6; \pi/4\}$, respectively, enabling operation in the linear region of the MZM. Note that Gray-coded bit-to-symbol mapping was explicitly assumed in the case of PAM4. The sequence of symbols is pulse-shaped at $n = 2$ Sa/sym by a 0.25 roll-off raised cosine filter. It is then applied to the experimental test-bed. We performed $Z = 100$ loads of the DAC, each of them equivalent to a sequence of $T = 249750$ PAM symbols $m_{i,j}$, $i \in \{1, \ldots, Z\}$ and $j \in \{1, \ldots, T\}$. The sequences of received symbols $\mathbf{y}_{i,j}$ after propagation through the test-bed link were utilised together with the labels $m_{i,j}$ for optimization of the DSP parameters and testing. Sequences $i \in \{1, \ldots, 90\}$ were used to form the training sets

$$\mathbf{D}^{\text{PAM}} := \begin{bmatrix} \mathbf{y}_{1,1} & \cdots & \mathbf{y}_{1,T/2} \\ \mathbf{y}_{1,T/2+1} & \cdots & \mathbf{y}_{1,T} \\ \vdots & \ddots & \vdots \\ \mathbf{y}_{90,1} & \cdots & \mathbf{y}_{90,T/2} \\ \mathbf{y}_{90,T/2+1} & \cdots & \mathbf{y}_{90,T} \end{bmatrix}$$

$$\mathbf{L}^{\text{PAM}} := \begin{bmatrix} m_{1,1} & \cdots & m_{1,T/2} \\ m_{1,T/2+1} & \cdots & m_{1,T} \\ \vdots & \ddots & \vdots \\ m_{90,1} & \cdots & m_{90,T/2} \\ m_{90,T/2+1} & \cdots & m_{90,T} \end{bmatrix}$$

where $\mathbf{D}^{\text{PAM}} \in \mathbb{R}^{180 \times T \cdot n/2}$ consists of the received elements $\mathbf{y}_{i,j}$ and $\mathbf{L}^{\text{PAM}} \in \mathbb{R}^{180 \times T/2}$ has the elements $m_{i,j}$. During the training of the PAM systems we pick column-wise with a window of W from the data to form a mini-batch. Similar to the auto-encoder training one full DAC load is split into two independently treated parts, e.g. $\mathbf{y}_{1,1} \ldots \mathbf{y}_{1,T/2}$ and $\mathbf{y}_{1,T/2+1} \ldots \mathbf{y}_{1,T}$, ensuring an identical mini-batch size. Again, one epoch over the data was performed. Testing is performed on the remaining sequences $i \in \{91, \ldots, 100\}$.

III. Digital Signal Processing Schemes and Performance

The IM/DD transmission is distorted by chromatic dispersion, which introduces ISI from both preceding and succeeding symbols, and square-law detection by a photo-diode [11]. The joint effects of these phenomena render the communication channel nonlinear with memory, necessitating processing to recover data sequences. However, due to the lack of optimal, computationally feasible algorithms, carefully chosen approximations are required. In this section we investigate classical DSP as well as deep learning-based solutions.

A. Sliding Window BRNN Auto-encoder

An efficient auto-encoder implements the complete chain of transmitter, channel and receiver as an end-to-end deep ANN, whose parameters are optimized in a single, joint learning process [9]. In particular, for optical IM/DD, a bidirectional recurrent neural network (BRNN) can be used to handle the memory of the nonlinear channel [15], [17]. As proposed in [8], the BRNN is combined with a powerful receiver algorithm for sequence estimation.

1) BRNN processing: The schematic of the BRNN building block, used at both transmitter and receiver, is visualized in Fig. 2-a). In the forward direction, the input $\mathbf{x}_t$ at time t is processed by the recurrent cell together with the previous output $\overrightarrow{\mathbf{h}}_{t-1}$ to produce the updated output $\overrightarrow{\mathbf{h}}_t = \alpha_{\text{Tx/Rx}}\left(\mathbf{W}\left(\mathbf{x}_t^T \quad \overrightarrow{\mathbf{h}}_{t-1}^T\right)^T + \mathbf{b}\right)$, where ^{T} denotes the matrix transpose, $\mathbf{W} \in \mathbb{R}^{n \times (M+n)}$ and $\mathbf{b} \in \mathbb{R}^n$ (transmitter) or $\mathbf{W} \in \mathbb{R}^{2M \times (n+2M)}$ and $\mathbf{b} \in \mathbb{R}^{2M}$ (receiver) are the weight matrix and bias vector, respectively, and $\alpha_{\text{Tx/Rx}}$ is the activation function. In the backward direction $\mathbf{x}_t$ is instead combined with $\overleftarrow{\mathbf{h}}_{t+1}$ to yield $\overleftarrow{\mathbf{h}}_t$. The output of the transmitter BRNN at time t is $\mathbf{h}_t^{\text{tx}} = \frac{1}{2}\left(\overrightarrow{\mathbf{h}}_t + \overleftarrow{\mathbf{h}}_t\right)$, while at the receiver $\mathbf{h}_t^{\text{rx}} = \left(\overrightarrow{\mathbf{h}}_t^T \quad \overleftarrow{\mathbf{h}}_t^T\right)^T$. To enable approximately linear MZM operation, the transmitter activation is the clipping function $\alpha_{\text{Tx}}(\mathbf{a}) = \alpha_{\text{ReLU}}(\mathbf{a}) - \alpha_{\text{ReLU}}\left(\mathbf{a} - \frac{\pi}{4}\right)$ [12]. At the receiver, we employ the ReLU activation $\alpha_{\text{Rx}}(\mathbf{a}) = \alpha_{\text{ReLU}}(\mathbf{a})$ [5]. After the ReLU nodes, a *softmax* layer is

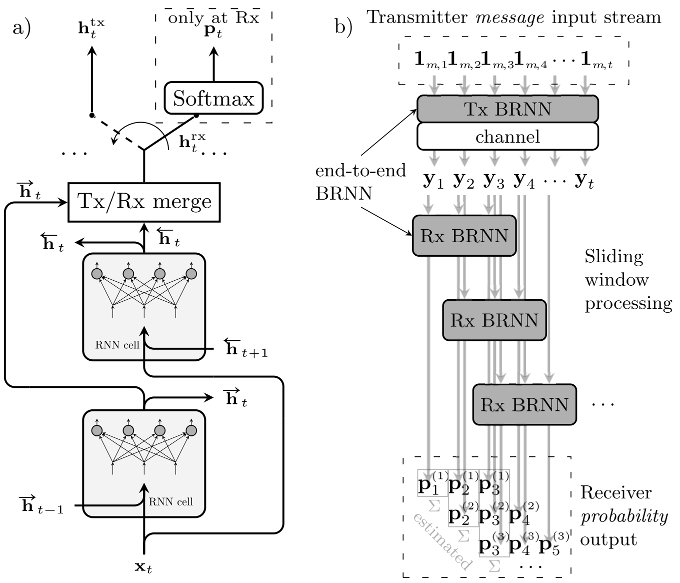
a)
$\mathbf{h}_t^{\text{tx}}$
only at Rx
$\mathbf{p}_t$
Softmax
$\mathbf{h}_t^{\text{rx}}$
Tx/Rx merge
$\overrightarrow{\mathbf{h}}_t$
$\overleftarrow{\mathbf{h}}_t$
RNN cell
$\overleftarrow{\mathbf{h}}_{t+1}$
$\overrightarrow{\mathbf{h}}_{t-1}$
$\mathbf{x}_t$
b)
Transmitter *message* input stream
$\mathbf{1}_{m,1}\mathbf{1}_{m,2}\mathbf{1}_{m,3}\mathbf{1}_{m,4}\cdots\mathbf{1}_{m,t}$
Tx BRNN
channel
end-to-end BRNN
$\mathbf{y}_1\ \mathbf{y}_2\ \mathbf{y}_3\ \mathbf{y}_4\ \cdots\ \mathbf{y}_t$
Rx BRNN
Sliding window processing
$\mathbf{p}_1^{(1)}\mathbf{p}_2^{(1)}\mathbf{p}_3^{(1)}$
$\mathbf{p}_2^{(2)}\mathbf{p}_3^{(2)}\mathbf{p}_4^{(2)}$
$\mathbf{p}_3^{(3)}\mathbf{p}_4^{(3)}\mathbf{p}_5^{(3)}$
estimated
Receiver *probability* output

Fig. 2. Schematic of a) bidirectional RNN processing and b) sliding window sequence estimation algorithm.

applied to obtain probability vectors at the receiver according to $\mathbf{p}_t = \text{softmax}(\mathbf{W}_{\text{softmax}}\mathbf{h}_t^{\text{rx}} + \mathbf{b}_{\text{softmax}})$ with $\mathbf{p}_t \in \mathbb{R}^M$, $\mathbf{W}_{\text{softmax}} \in \mathbb{R}^{M\times 4M}$ and $\mathbf{b}_{\text{softmax}} \in \mathbb{R}^M$.

In the auto-encoder setting, the BRNN transmitter takes as an input a sequence of independently drawn messages m, represented as a stream of one-hot vectors $(\ldots, \mathbf{1}_{m,t-1}, \mathbf{1}_{m,t}, \mathbf{1}_{m,t+1}, \ldots)$, with $\mathbf{1}_{m,t} \in \mathbb{R}^M$ (with elements "1" at position m and "0" elsewhere). It encodes them into a sequence of transmit blocks of n samples $(\ldots, \mathbf{h}_{t-1}^{\text{tx}}, \mathbf{h}_t^{\text{tx}}, \mathbf{h}_{t+1}^{\text{tx}}, \ldots)$. This sequence is sent through the communication channel. The receiver processes the noisy samples $(\ldots, \mathbf{y}_{t-1}, \mathbf{y}_t, \mathbf{y}_{t+1}, \ldots)$ to obtain $(\ldots, \mathbf{h}_{t-1}^{\text{rx}}, \mathbf{h}_t^{\text{rx}}, \mathbf{h}_{t+1}^{\text{rx}}, \ldots)$ and then probability vectors $(\ldots, \mathbf{p}_{t-1}, \mathbf{p}_t, \mathbf{p}_{t+1}, \ldots)$ via softmax.

2) Sliding window sequence estimation algorithm: In principle, one could apply the BRNN on the full sequence. However, to improve receiver latency, we employ a windowed processing, as proposed in [8]. Figure 2-b) shows a schematic of the end-to-end processing and estimation of data sequences. For a given sequence of T messages, the transmitter BRNN encodes the full stream of input one-hot vectors $\mathbf{1}_{m,1}, \ldots, \mathbf{1}_{m,T}$. The obtained waveform is then subject to the channel, yielding the sequence of received blocks of samples $\mathbf{y}_1, \ldots, \mathbf{y}_T$. At a time t, the receiver BRNN processes the window of W received blocks $\mathbf{y}_t, \ldots, \mathbf{y}_{t+W-1}$, transforming them into W probability vectors $\mathbf{p}_t^{(t)}, \ldots, \mathbf{p}_{t+W-1}^{(t)}$. Then it slides one time slot ahead to process the blocks $\mathbf{y}_{t+1}, \ldots, \mathbf{y}_{t+W}$. The final output probability vectors are given by

$$\mathbf{p}_i = \sum_{k=0}^{\max(W,i)-1} a_i^{(k)} \mathbf{p}_i^{(i-k)} \tag{1}$$

where we set $a_i^{(q)} = [\max(W,i)]^{-1}$. We refer the interested reader to [16, III-B] for a study on optimizing the weight coefficients $a_i^{(q)}$, yielding additional small gains compared to the uniform assumption followed in this work. Decision on the transmitted message can be made using the probability output of the estimation algorithm, i.e. $\tilde{m}_i = \text{argmax}(\mathbf{p}_i)$. We count a block (symbol) error when $m_i \neq \tilde{m}_i$. The block error rate (BLER) for the transmitted sequence is thus given by

$$\text{BLER} = \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\{m_i \neq \tilde{m}_i\}}, \tag{2}$$

where $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function, equal to 1 when the argument is satisfied and 0 otherwise. Using the method described in [17, Sec. III-C], we perform optimization of the bit-to-symbol mapping to obtain the BER. The reported BER is the average BER over all test sequences.

3) Training and performance: First, the end-to-end system learning is performed following [17, Sec. III-A]) on a numerical link model identical to [15, Sec. 2.1]. The optimized transceiver is applied "as is" to the experimental test-bed. Figure 3 shows the performance of this system for a fixed sliding window size of $W = 10$. After 50 km, a BER of $1.3 \cdot 10^{-3}$ is achieved, while at 60 km it is slightly above the

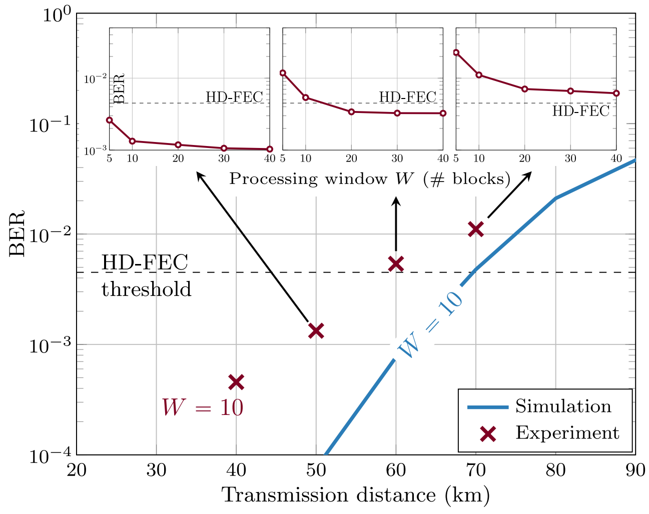

Fig. 3. BER versus distance for the SBRNN auto-encoder trained on a channel model. Inset figures show the BER vs. window size W at 50, 60 and 70 km.

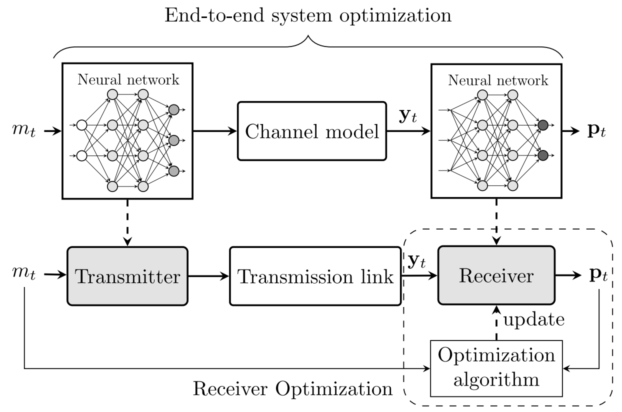

Fig. 4. Receiver optimization using the collected data from measurements.

$4.5 \cdot 10^{-3}$ HD-FEC threshold [18] (6.7% overhead). A simple method for performance improvement at longer distances is to increase the window in the sequence estimation algorithm, thus compensating for more of the ISI. The inset figures examine the BER at 50, 60 and 70 km as a function of W. We see that at 60 km, the system can still operate below the HD-FEC threshold with $W = 20$. In an excellent agreement with the simulation [15], the experiment shows diminishing gains from further increase of W since noise and non-linearities become more dominant when the ISI is compensated.

A different approach for improving the system performance is shown in Fig. 4. It uses the collected experimental data for receiver optimization, allowing to reduce the penalty stemming from the discrepancy between simulation model and the actual transmission link [10]. Training elements are the pairings of an input message m_t from $\mathbf{L}^{\text{AE}}$ and the corresponding received block of samples $\mathbf{y}_t$ from $\mathbf{D}^{\text{AE}}$ (see Sec. II-B). The receiver parameters $\boldsymbol{\theta}_{\text{Rx}}$ are adjusted via SGD, minimizing the average loss L over a mini-batch S of the training data, given by

$$L(\boldsymbol{\theta}_{\text{Rx}}) = \frac{1}{|S|}\sum_{(m_t,\mathbf{y}_t)\in S} \ell(m_t, f_{\text{rec},t}(\ldots, \mathbf{y}_t, \ldots)), \tag{3}$$

where $\ell(\mathbf{x},\mathbf{y}) = -\sum_i x_i \log(y_i)$ is the cross entropy and f_{rec}

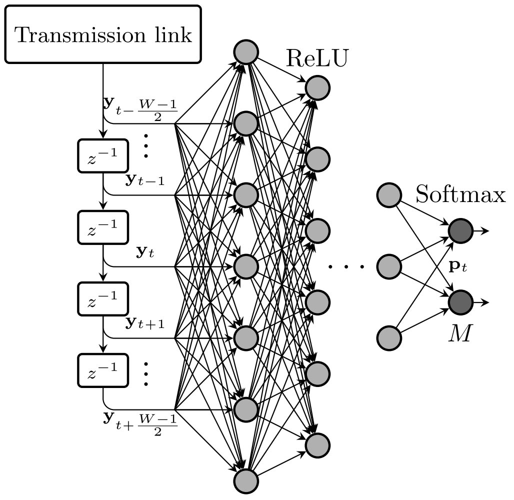

Fig. 5. Schematic of the sliding window FFNN receiver for PAM symbols.

is the receiver ANN function. More specifically, training is performed in the following way: At the beginning, the outputs $\overrightarrow{\mathbf{h}}_{t-1}$ and $\overleftarrow{\mathbf{h}}_{t+1}$ in the forward and backward passes of the BRNN are initialized to $\mathbf{0}$. At an optimization step s, the mini-batch of elements[1] $\mathbf{D}^{\text{AE}}[:,s:(s+V)]$ is processed by the receiver to obtain corresponding output probability vectors, where V is the training window which we fixed to 10. The loss between the input messages $\mathbf{L}^{\text{AE}}[:,s:(s+V)]$ and probability outputs is computed and averaged before completing a single iteration of the optimization algorithm. After optimization, the system performance is evaluated using the testing set and reported in Sec. III-E. Note that transmitter optimization from experimental data has been implemented for optical fiber systems in [21]. To avoid the additional complexity, we do not consider such an optimization in this paper.

B. Sliding Window BRNN PAM Receiver

This system combines the PAM transmitter with the SBRNN receiver. The receiver is implemented following an identical procedure to Sec. III-A. Differently, here we used the elements of $\mathbf{D}^{\text{PAM}}$ and $\mathbf{L}^{\text{PAM}}$ for training. At an iteration s, the mini-batch of elements $\mathbf{D}^{\text{PAM}}[:,s:(s+V)]$ is processed by the BRNN to obtain the corresponding output probability vectors. We compute the average cross entropy loss between the input messages $\mathbf{L}^{\text{PAM}}[:,s:(s+V)]$ and these outputs before completing an optimization step of the ANN parameters. We fixed $V=61$, such that the scheme is trained with a processing memory identical to the auto-encoder.

C. Sliding Window FFNN PAM Receiver

In this scheme, a PAM sequence is transmitted through the link and the received samples are fed to a multi-layer feed-forward ANN, as shown in Fig. 5 estimating the center symbol of the sequence. More specifically, the transmitted message m_t at time t is detected by processing the train of W received symbols of $n=2$ samples $(\mathbf{y}_{t-\frac{W-1}{2}},\ldots,\mathbf{y}_t,\ldots,\mathbf{y}_{t+\frac{W-1}{2}})$.

[1]We use the MATLAB notation $\mathbf{A}[:,i:j]$ to denote extracting the $j-i+1$ columns of column indices $i,i+1,\ldots,j$ from $\mathbf{A}$.

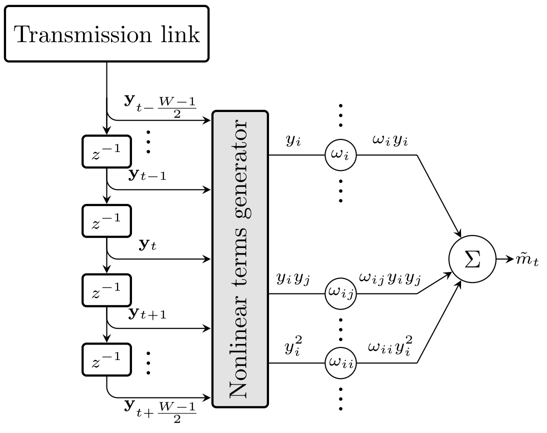

Fig. 6. Schematic of the nonlinear Volterra equalizer for PAM symbols.

In this receiver, the processing memory dictates the size of the layers, the first one having parameters $\mathbf{W}\in\mathbb{R}^{W\cdot n\times 4W\cdot n}$ and $\mathbf{b}\in\mathbb{R}^{4W\cdot n}$. Subsequently, 6 hidden layers are employed before m_t is estimated. The number of nodes on each of these is given by $\lfloor 4W\cdot n/(2^{i-1})\rfloor$, where i is the hidden layer index. The ReLU activation is applied on all except the final layer, where *softmax* computes a probability vector for the transmitted PAM2/4 symbol with $\mathbf{p}_t\in\mathbb{R}^{2/4}$. Afterwards, we slide the processing window by one position ahead to estimate the next symbol in the sequence. Note that a similar approach has been considered previously in [7], [15]. Training is performed in the following way: at an iteration s, the mini-batch of elements $\mathbf{D}^{\text{PAM}}[:,s:(s+W)]$ is processed by the FFNN to obtain the corresponding output probability vectors. We compute the average cross entropy between these outputs and the input messages $\mathbf{L}^{\text{PAM}}[:,s+\frac{W-1}{2}]$ and complete an optimization step of the parameters via SGD.

D. Nonlinear Volterra Equalizer for PAM Transmission

Volterra equalizers, shown schematically in Fig. 6, have been widely considered for IM/DD links based on PAM [22]. To compensate for linear and nonlinear ISI, we use a second order Volterra filter, whose output can be expressed as

$$\begin{aligned}\tilde{m}_t = \omega_{dc} &+ \sum_{q_1=-\frac{W-1}{2}\cdot n}^{\frac{W-1}{2}\cdot n} \omega_{q_1}\cdot y_{t+q_1} \\ &+ \sum_{q_2=-\frac{W_1-1}{2}\cdot n}^{\frac{W_1-1}{2}\cdot n}\ \sum_{q_3=k_2}^{\frac{W_1-1}{2}\cdot n} \omega_{q_2,q_3}\cdot y_{t+q_2}\cdot y_{t+q_3},\end{aligned}\tag{4}$$

where W and W_1 denote the symbol memory in the first and second order series, respectively, and ω_i and $\omega_{i,j}$ are the first and second order coefficients. Similar to the other PAM schemes, the algorithm is processing the neighborhood of W received symbols of $n=2$ samples $(\mathbf{y}_{t-\frac{W-1}{2}},\ldots,\mathbf{y}_t,\ldots,\mathbf{y}_{t+\frac{W-1}{2}})$, thus including the ISI from pre- and post-cursor samples in the detection of the current input. To capture identical amount of memory, we fix $W=61$, while we choose $W_1=21$ to keep the complexity low. The filter coefficients are optimized using the MATLAB `fitlm`

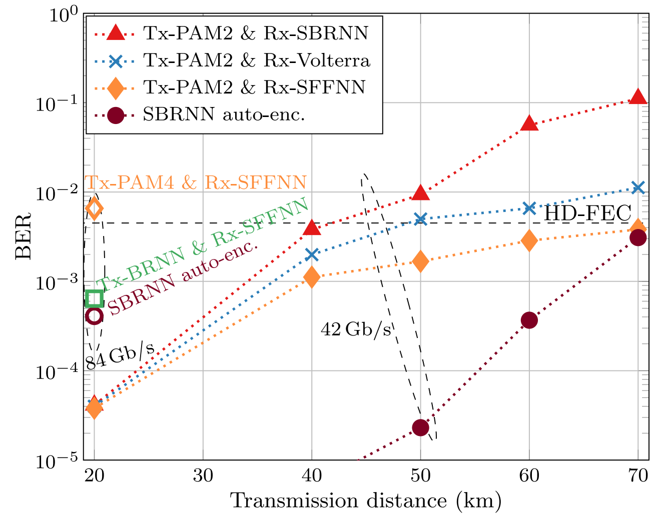

Fig. 7. BER as a function of distance for 42 Gb/s and 84 Gb/s systems employing DSP schemes trained on experimental data.

function for linear and polynomial regression. Training is performed on a randomly chosen pair of transmitted sequence $\mathbf{L}^{\text{PAM}}[i,:]$ and received samples $\mathbf{D}^{\text{PAM}}[i,:]$.

E. Performance of the DSP optimized on experimental data

To compare the performance of the DSP, we fix the number of bits simultaneously processed by each receiver algorithm. Thus, for the auto-encoder we set $W = 10$ (60 bits), while for the PAM2 systems $W = 61$ (61 bits). To allow for a better compensation in the case of PAM4, we kept $W = 61$. Figure 7 shows the BER of all experimental systems at different distances. Note that systems were trained separately for each distance and multi-distance learning methods (see [12, Sec. IV]) fall outside of this manuscript's scope. We see that for shorter lengths and 42 Gb/s, the SBRNN auto-encoder has a BER much lower than the PAM schemes. In terms of system reach, it allows transmission up to 70 km below HD-FEC, similar to the PAM2&SFFNN. Both schemes outperformed the Volterra equalizer. The results show the SFFNN as a viable receiver solution for the conventional PAM. For further investigation, we employ it with PAM4 in a 84 Gb/s system. Moreover, we combined the SFFNN receiver in an auto-encoder setting with the BRNN transmitter. The results show that the PAM4&SFFNN cannot transmit below the HD-FEC at 20 km, while the Tx-BRNN&Rx-SFFNN auto-encoder is significantly superior, achieving a BER close to the SBRNN. Nevertheless, it is worth mentioning that, unlike the SBRNN, the number of parameters in the SFFNN receiver increases rapidly with the processing memory. For a computational complexity comparison between the SBRNN and SFFNN systems we refer the interested reader to [15, Sec. 3].

IV. Conclusions

We investigated the experimental performance of deep learning in end-to-end as well as receiver-only application for DSP in short reach optical fiber links. Our results show that 42 Gb/s can be transmitted at up to 60 km below a common HD-FEC threshold by a simple SBRNN auto-encoder which learns transceiver parameters based on a model. The reach can be enhanced to 70 km by optimization of the receiver on experimental data. Interestingly, 42 Gb/s systems based on PAM2 and receiver-only DSP by an SFFNN covered similar distances, outperforming nonlinear Volterra equalization. In addition to the SBRNN, we examined an auto-encoder based on BRNN transmitter and SFFNN receiver. Both configurations allowed to transmit 84 Gb/s at 20 km. Our experiment highlights deep learning as a viable DSP solution for low-cost optical communications to increase reach or enhance the data rate at shorter distances.

References

[1] O. Simeone, “A very brief introduction to machine learning with applications to communication systems,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, no. 4, pp. 648-664, 2018.
[2] Danish Rafique and Luis Velasco, "Machine Learning for Network Automation: Overview, Architecture, and Applications," *J. Opt. Commun. Netw.*, vol. 10, no. 10, D126-D143, 2018.
[3] C. Jiang *et al.*, “Machine learning paradigms for next-generation wireless networks,” *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98-105, 2017.
[4] F. N. Khan *et al.*, “An optical communication's perspective on machine learning and its applications,” *J. Lightw. Technol.*, vol. 37, no. 2, pp. 493-516, 2019.
[5] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*. MIT Press, 2016.
[6] C. Häger and H. Pfister, “Nonlinear interference mitigation via deep neural networks,” in *Proc. of OFC*, 2018, paper W3A.4.
[7] D. van Veen and V. Houtsma, "Strategies for economical next-generation 50G and 100G passive optical networks," *J. Opt. Commun. Netw.*, vol. 12, no. 1, A95-A103, 2020.
[8] N. Farsad and A. Goldsmith, “Neural network detection of data sequences in communication systems,” *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5663-5678, 2018.
[9] T. O'Shea and J. Hoydis, “An introduction to deep learning for the physical layer,” *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.
[10] S. Dörner *et al.*, “Deep learning-based communication over the air,” *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 132-143, 2018.
[11] G. Agrawal, *Fiber-optic Communication Systems*, 4th ed., John Wiley & Sons, Inc., 2010.
[12] B. Karanov *et al.*, “End-to-end deep learning of optical fiber communications,” *J. Lightw. Technol.*, vol. 36, no. 20, pp. 4843-4855, 2018.
[13] S. Li *et al.*, “Achievable information rates for nonlinear fiber communication via end-to-end auto-encoder learning,” in *Proc. of ECOC*, 2018, pp. 1-3.
[14] R. T. Jones *et al.*, “Deep learning of geometric constellation shaping including fiber nonlinearities,” in *Proc. of ECOC*, 2018, pp. 1-3.
[15] B. Karanov *et al.*, “End-to-end optimized transmission over dispersive intensity-modulated channels using bidirectional recurrent neural networks,” *Opt. Express*, vol. 27, no. 14, pp. 19650-19663, 2019.
[16] B. Karanov *et al.*, “Optical fiber communication systems based on end-to-end deep learning,” in *Proc. of IPC*, 2020, pp. 1-2.
[17] B. Karanov *et al.*, “Deep learning for communication over dispersive nonlinear channels: performance and comparison with classical digital signal processing,” in *Proc. of Allerton Conference*, 2019, pp. 192-199.
[18] Z. Wang, “Super-FEC codes for 40/100 Gbps networking,” *IEEE Commun. Lett.*, vol. 16, no. 12, pp. 2056–2059, 2012.
[19] M. Chagnon, “Optical communications for short reach,” *J. Lightw. Technol.*, vol. 37, no. 8, pp. 1779-1795, 2019.
[20] T. Eriksson, H. Bülow, and A. Leven, “Applying neural networks in optical communication systems: possible pitfalls,” *IEEE Photon. Technol. Lett.*, vol. 29, no. 23, pp. 2091-2094, 2017.
[21] B. Karanov *et al.*, “Concept and experimental demonstration of optical IM/DD end-to-end system optimization using a generative model,” in *Proc. of OFC*, 2020, paper Th2A.48.
[22] N. Stojanovic *et al.*, “Volterra and Wiener equalizers for short-reach 100G PAM-4 Applications,” *J. Lightw. Technol.*, vol. 35, no. 21, pp. 4583-4594, 2017.