# Parallelization of Recurrent Neural Network-Based Equalizer for Coherent Optical Systems via Knowledge Distillation

Sasipim Srivallapanondh, Pedro J. Freire, Bernhard Spinnler, Nelson Costa, Antonio Napoli, Sergei K. Turitsyn, Jaroslaw E. Prilepsky

***Abstract*—The recurrent neural network (RNN)-based equalizers, especially the bidirectional long-short-term memory (biLSTM) structure, have already been proven to outperform the feed-forward NNs in nonlinear mitigation in coherent optical systems. However, the recurrent connections still prevent the computation from being fully parallelizable. To circumvent the non-parallelizability of recurrent-based equalizers, we propose, for the first time, *knowledge distillation* (KD) to recast the biLSTM into a parallelizable feed-forward 1D-convolutional NN structure. In this work, we applied KD to the cross-architecture regression problem, which is still in its infancy. We highlight how the KD helps the student's learning from the teacher in the regression problem. Additionally, we provide a comparative study of the performance of the NN-based equalizers for both the teacher and the students with different NN architectures. The performance comparison was carried out in terms of the Q-factor, inference speed, and computational complexity. The equalization performance was evaluated using both simulated and experimental data. The 1D-CNN outperformed other NN types as a student model with respect to the Q-factor. The proposed 1D-CNN showed a significant reduction in the inference time compared to the biLSTM while maintaining comparable performance in the experimental data and experiencing only a slight degradation in the Q-factor in the simulated data.**

***Index Terms*—Artificial intelligence, machine learning, recurrent neural networks, parallelization, knowledge distillation, nonlinear equalizer, coherent detection.**

## I. Introduction

OPTICAL fiber nonlinearity (e.g., the Kerr effect) poses a significant challenge as it limits the optimal optical launch power, thus the information rate in current coherent transmission systems. The importance of optical nonlinearity increases as transmission bandwidth continues to expand [1]. To mitigate the impact of nonlinearity, various digital signal processing (DSP) techniques have been proposed [2]. Machine learning techniques, especially neural networks (NNs) used in digital communication have been considered over the last two decades [3]. Recently, NNs have emerged as promising tools for optical channel post-equalization due to their universal approximation capability. This capability allows them to effectively approximate the inverse optical channel transfer function and counteract nonlinear distortions. Nonetheless, the main challenges of the NN-based equalizers, like the capability to perform in high-speed processing, and computational complexity (CC), still prevent it from real implementation. Among NN architectures, recurrent-based neural networks (RNNs) have demonstrated superior equalization performance compared to feed-forward NNs when dealing with nonlinear impairments [4]–[6]. However, the RNN structure, characterized by feedback loops, see Fig. 1, presents a challenge in achieving parallelization. Parallelization is essential for low-complexity hardware implementations and high-speed processing in optical networks [7]. To achieve low latency and high throughput, modern high-speed optical networks, rely on effective parallelization techniques, as they increase throughput and reduce computational time [8].

We introduce a novel approach to address the parallelization challenge of RNN-based equalizers, in this case, the bidirectional long-short-term memory (biLSTM) network, through knowledge distillation (KD). Our work focuses on transforming the biLSTM-based equalizer into a feed-forward structure using KD. Generally, KD involves transferring knowledge from a larger model (the teacher) to a more compact model (the student) that requires fewer computations [9]. The previous research on KD has predominantly concentrated on classification tasks involving teacher and student networks with similar topologies. The application of KD to regression tasks and cross-architecture KD, as proposed in our work, has only recently gained attention [10]. The KD in our case aims to be used as a tool to convert the structure of the NN, rather than to reduce the CC. Our approach not only achieves a parallelizable structure but also reduces the inference latency, thereby addressing the challenges associated with implementing biLSTM in low-complexity hardware for high-speed processing. Furthermore, our NN-based equalizers recover multi-symbol output, which helps reduce the CC per recovered symbol [11]. In this work, we:

- Present, for the first time, a successful workaround for the parallelization of RNN-based equalizers using KD. We transfer knowledge from a recurrent teacher model,

This work has received funding from the EU Horizon 2020 program under the Marie Skłodowska-Curie grant agreement No. 956713 (MENTOR). SKT acknowledges the support of the EPSRC project TRANSNET (EP/R035342/1). Bernhard Spinnler, Nelson Costa, Antonio Napoli would like to thank the Horizon Europe project ALLEGRO, GA n. 101092766.

Sasipim Srivallapanondh, Pedro J. Freire, Sergei K. Turitsyn and Jaroslaw E. Prilepsky are with Aston Institute of Photonic Technologies, Aston University, United Kingdom, s.srivallapanondh@aston.ac.uk.

Antonio Napoli, and Bernhard Spinnler are with Infinera R&D, St-Martin-Str. 76, 81541, Munich, Germany. anapoli@infinera.com

Nelson Costa is with Infinera Unipessoal, Lda, Rua da Garagem nº1, 2790-078 Carnaxide, Portugal, ncosta@infinera.com.

Manuscript received Month DD, YYYY; revised Month DD, yyyy.

specifically a biLSTM coupled with a 1D-convolutional NN (1D-CNN), which was proposed as an efficient equalizer model in [12]. The knowledge is transferred to the proposed feed-forward student model, a 1D-CNN architecture, as illustrated in Fig. 2a.
- Compare performance, CC, and latency of the teacher model and the student model when adopting different NN architectures. We consider the bidirectional RNN (biRNN), 1D-CNN, and multilayer perception (MLP).
- Apply KD framework to different transmission scenarios from both simulation and experiments to verify the effectiveness of the KD.

The subsequent sections of this paper are as follows. Sec.II explains in detail the definition and advantages of parallelization both for the NN architecture and for the output symbols. In Sec. III, the KD framework and its background were introduced. After that, Sec. IV describes the numerical and experimental setup, followed by the training process. Sec. V provides comparative analyses and performance evaluations of our approach in various scenarios and various NN structures. Additionally, we discuss the implications and limitations of the proposed approach. The final Sec. VI concludes the article.

## II. Parallelization

### A. Parallelization of the NN architecture

Parallelization of the NN structure should be considered when designing the model. Parallelization can more efficiently leverage hardware usages like multi-core GPUs and CPUs, resulting in the acceleration of the training or inference of the NN. Each NN type offers a different degree of parallelism, depending on its unique architecture. For instance, the structure of NN can have a feed-forward nature like CNNs or a recursive one like RNNs. Fig. 1 illustrates the recurrent structure at the top with a feedback loop, preventing parallelization, while the feed-forward structure at the bottom can process multiple sets of inputs and provide multiple outputs simultaneously.

In this work, we focus on the 1D-CNNs. 1D-CNNs are feed-forward-based networks where the input temporal sequential batches are independently processed, accordingly parallel operations are possible [13], [14]. In signal processing, the convolutional layers share some similarities with Finite Impulse Response (FIR) filters, as they both are implemented based on convolution operation. The output at the current time step depends solely on the current and past inputs. The equation of the 1D-convolutional layer can be formularized as:

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

where $y_i^f$ denotes the output, known as a feature map, of a convolutional layer built by the filter $f$ in the $i$-th input element, $n_k$ is the kernel size, $n_i$ is the size of the input vector, $x^{in}$ represents the raw input data, $k_j^f$ denotes the $j$-th trainable convolution kernel of the filter $f$ and $b^f$ is the bias of the filter $f$. For the FIR filter, the equation can be summarized as:

$$y(n) = \sum_{i=0}^{N} b_i \cdot x(n-i), \tag{2}$$

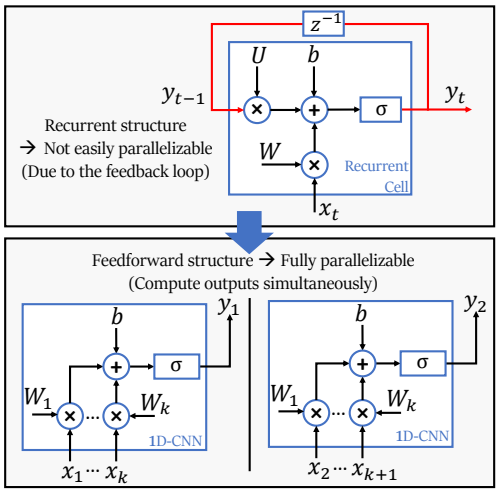



Fig. 1: Illustration of the parallelizability comparison between a recurrent cell (top) that is not easily parallelizable due to the feedback loop and a 1D-CNN (bottom) that can process output simultaneously.

where $y(n)$ represents the output signal, $u(n)$ is the input signal, $N$ denotes the filter order. The FIR filter computation is fully parallelizable as it does not contain any recursive part [15]–[17].

In the case of RNNs, they are useful in learning sequential data. RNN's output of the current stage $y_t$ takes into account the current stage input $x_t$ and the output of the previous stage $y_{t-1}$. The equation of the RNN for a given time step $t$ is as follows:

$$h_t = \phi(Wx_t + Uh_{t-1} + b), \tag{3}$$

where $\phi$ is the nonlinear activation functions, $x_t \in \mathbb{R}^{n_i}$ is the $n_i$-dimensional input vector at time $t$, $h_t \in \mathbb{R}^{n_h}$ is a hidden layer vector of the current state with size $n_h$, $W \in \mathbb{R}^{n_h \times n_i}$ and $U \in \mathbb{R}^{n_h \times n_h}$ represent the trainable weight matrices, and $b$ is the bias vector. Calculating the current stage output as a function of the previous stage output creates a recursive evaluation, which is indeed sequential processing and precludes parallelization. In the signal processing context, RNNs can be compared to Infinite Impulse Response (IIR) filters [18], [19]. This can be observed from the equations Eq. (3) and Eq. (4). The equation of the first-order IIR filter is:

$$y(n) = bx(n) + ay(n-1), \tag{4}$$

where $y(n)$ denotes the output signal, $x(n)$ is the input signal, $b$ represents the feed-forward filter coefficient and $a$ is the feedback filter coefficient. Even though Ref. [20] has unfolded the pipelining processing of the IIR filter to parallel processing to some degree of parallelism, there is still a feedback loop, as shown in Fig. 8 in [20]. Therefore, we can imply that the RNN-based architecture is not fully parallelizable. This implication can be applied to other variations of the RNN-based networks, such as Long Short-Term Memory (LSTM), biLSTM [21], and Gated Recurrent Unit (GRU) networks.

To be more specific, this paper focuses on the biLSTM model architecture. biLSTM is the network consisting of two

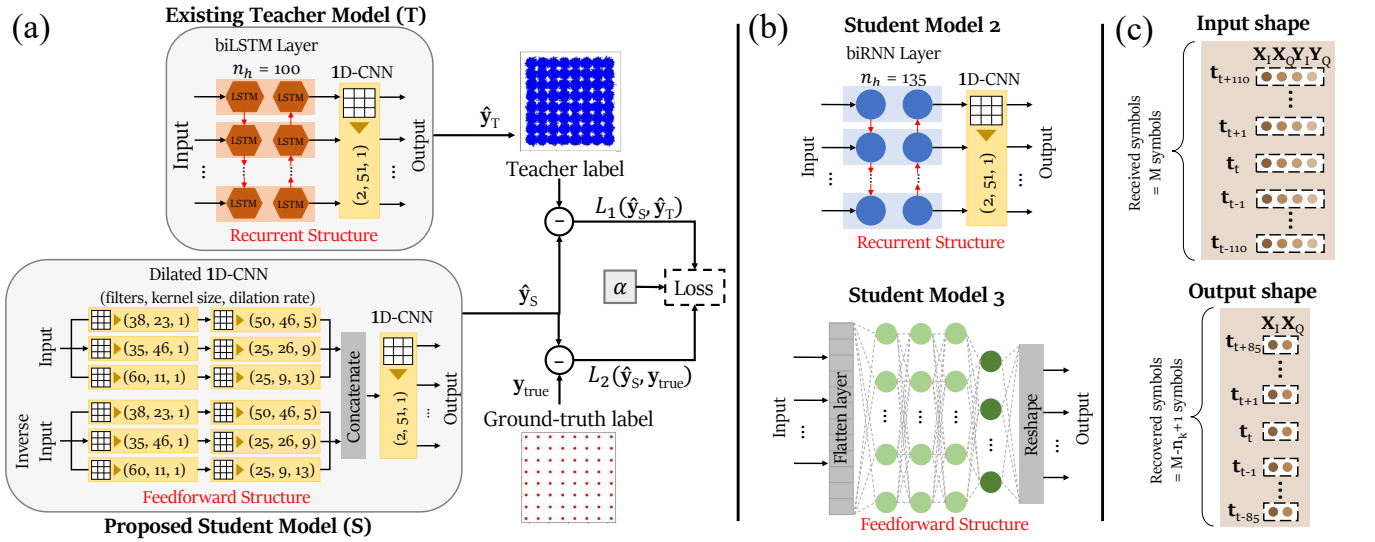



Fig. 2: (a) KD framework with biLSTM+1D-CNN as a teacher model and dilated 1D-CNN as a proposed student model; (b) different student architectures: student model 2 - biRNN model and student model 3 - MLP model; (c) the input of the model contains $M$ real (I) and imaginary (Q) components of both X and Y polarization the received symbols, to recover I and Q parts components of X polarization of $M - n_k + 1$ symbols at the output.

separate LSTM layers: for forward and backward directions [22]. Because of a double recurrent setting, which cannot be fully parallelized, biLSTM is even more computationally expensive than LSTM and RNN. LSTM (see Eq. (5)) and RNN (see Eq. (3)) have essentially identical core properties: sequential processing and retaining past information through past hidden states. Due to the sequential processing of the recurrent setting, the model computation is very expensive due to limited parallelization [14]. The equations of a forward pass of an LSTM cell with a time step $t$ are given as:

$$\begin{aligned} i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i), \\ f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f), \\ o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o), \\ C_t &= f_t \odot C_{t-1} + i_t \odot \phi(W^c x_t + U^c h_{t-1} + b^c), \\ h_t &= o_t \odot \phi(C_t), \end{aligned} \tag{5}$$

where $f_t, i_t, o_t$ are forget gate's, input gate's and output gate's activation vector, respectively. $\phi$ is usually the "tanh" activation function, $\sigma$ is usually the sigmoid activation function. The sizes of each variable are $x_t \in \mathbb{R}^{n_i}$; $f_t, i_t, o_t \in (0,1)^{n_h}$, $C_t \in \mathbb{R}^{n_h}$; and $h_t \in (-1,1)^{n_h}$. The $\odot$ symbol represents the element-wise (Hadamard) multiplication.

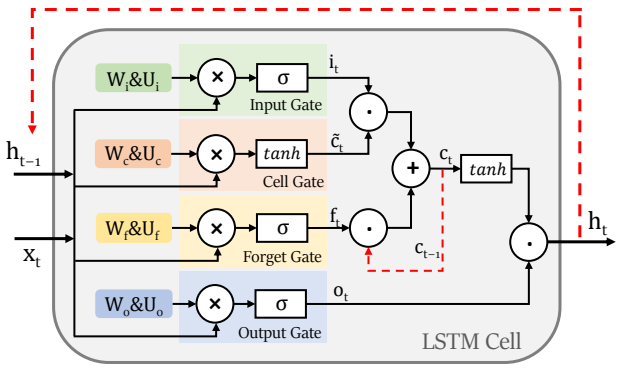



Fig. 3: Architecture of an LSTM cell.

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

### B. Parallelization of recovered symbols

Traditionally, the NN-based equalizers in previous work [4], [23], [24] were designed to recover one symbol at a time, which means that the output of the NN model represents only one recovered symbol at each inference step. However, the single-symbol output NN-based equalizers can be computationally inefficient, as the weights and biases trained to recover one symbol may still be useful for recovering multiple symbols [25]. When taking into account the pre- and post-cursor ISI (inter-symbol interference) and chromatic dispersion, the input window should be wider than the output window [26]. The initial and final input symbols in the window lack the information of their neighbors, resulting in a smaller number of recovered output symbols [12]. Multi-symbol output equalizers draw attention to the research areas, previously proposed by [12], [25], [27].

In this work, our NN-based equalizer recovers multi-symbol output in order to reduce the CC per recovered symbol. The shape of the input and the output is illustrated in Fig. 2c. The last layer of our models (except for the MLP) adopts the 1D-convolutional layer as in Ref. [12]. The 1D-convolutional

layer contains two filters to recover both real (I) and imaginary (Q) components of the output signal. The size of the output window or the number of recovered symbols at each inference step is $M-n_k+1$, where $M$ represents the input window size that NN processes at a time, and $n_k$ is the kernel size in the 1D-convolutional layer. In this work, the padding is set to zero, while the dilation and stride are set to one. The output size can be different depending on the padding, dilation, and stride. Note that the MLP model does not deploy a 1D-convolutional layer but it uses the neurons to recover multi-symbol output instead.

## III. Knowledge Distillation

Generally, KD refers to a model compression technique of transferring knowledge from a complex model, known as a teacher model, to a more compact one, known as a student model, which is less computational expensive to evaluate [9], [28]. KD allows the student model to have a comparable performance with respect to the teacher while requiring less CC. The student model exploits the teacher's predictions to assist its learning. The predictions from the teacher model referred to as "teacher labels" or "soft labels", are used to train the student model together with the ground truth labels to aid the student's learning. Most of the prior work proposed KD [9], [29]–[31] in a classification task. In classification, the ground truth labels are usually one-hot labels. The teacher labels contain useful information about the relative similarity of the incorrect predictions, while the one-hot labels do not provide that sort of information [9], [32]–[34]. For example, the teacher labels, which were the results of the Softmax function, contain probabilities of each class in a multi-class problem, whereas the one-hot label only contains one or zero for each class.

KD in a regression task is still in its infancy, but various papers, e.g., [10], [32], [35] have shown that KD can demonstrate promising results in this context. However, it is still ambiguous in some cases about how the student model can take advantage of the teacher's predictions in the regression task [32]. In our work, we demonstrate in Fig. 2a how the teacher labels contain the noise information in the constellation diagram compared to the ground truth labels. KD in regression also performs as an efficient regularizer to improve generalization. Sec. V shows the weight distribution of the student model trained with KD compared to the one without KD.

The loss paradigm is the joint loss function that takes into account both the loss between the student's and the teacher's predictions and the loss between the student's predictions and ground truths, as described in [10]. The loss function, illustrated in Fig. 2a, can be described as:

$$\mathrm{L_{KD}} = \alpha L(\hat{\boldsymbol{y}}_S, \hat{\boldsymbol{y}}_\mathrm{T}) + (1-\alpha) L(\hat{\boldsymbol{y}}_S, \boldsymbol{y}_\mathrm{true}), \tag{6}$$

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

In addition, KD is commonly employed when the teacher and student models possess similar network topologies, aiming to reduce CC [34]. However, the investigation of cross-architecture KD [10] remains limited. In this work, we apply KD to recast the NN structure in the regression problems, emphasizing the reduction of inference time rather than the CC in terms of the number of real multiplications.

## IV. Data Generation and NN Training

### A. Data Generation

The numerical simulator created the dataset by assuming the transmission of a single-channel (SC) 30 GBd, 64-QAM dual-polarization (DP) channel along $20 \times 50$ km standard single-mode fiber (SSMF) spans. The signal propagation through the fiber was represented by a generalized Manakov equation split-step Fourier method [36]. The SSMF is characterized by the effective nonlinearity coefficient $\gamma$ = 1.2 (W$\cdot$ km)$^{-1}$, chromatic dispersion coefficient $D$ = 16.8 ps/(nm$\cdot$km), and attenuation parameter $\alpha$ = 0.21 dB/km. At the end of each fiber span, optical fiber losses are compensated for by an Erbium-Doped Fiber Amplifier (EDFA) with a 4.5 dB noise figure. Downsampling and CD compensation (CDC) were performed on the receiver end. The CDC was performed in the frequency domain with the transfer function of the CD given by [36]: $G(z,\omega) = \exp\left(-\frac{j\omega^2\beta_2 z}{2}\right)$ where $\omega$ is the angular frequency, $\beta_2$ is the group delay dispersion parameter of the fiber and $z$ is the transmission length. Afterward, the received symbols were normalized and used as inputs of the NN.

In this work, the experimental data were also analyzed to verify the performance of the proposed KD framework. The transmission scenario was SC DP-probabilistic shaped (PS)-64QAM[1] in 34.4 GBd along $9\times110$ km SSMF fiber spans. The experimental setup in Fig. 4 was detailed in Ref. [12]. At the transmitter side, a symbol sequence with a modulation scheme of 64QAM (8bits/4D symbol) with a symbol rate of 34.4 GBd was mapped out of data bits generated by a Marsenner twister generator [37]. After that, the symbol sequence was passed through a digital root-raised cosine (RRC) filter with a 0.1 roll-off factor to limit the channel bandwidth to 37.5 GHz. The filtered digital samples were resampled and fed into a digital-to-analog converter (DAC) operating at 96 GSamples/s. The DAC outputs were then amplified by a four-channel electrical amplifier which drove a dual-polarization in-phase/quadrature Mach–Zehnder modulator (MZM). The modulator modulated a continuous waveform carrier generated by an external cavity laser at a wavelength of $\lambda = 1.55\mu$m. The resulting optical

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

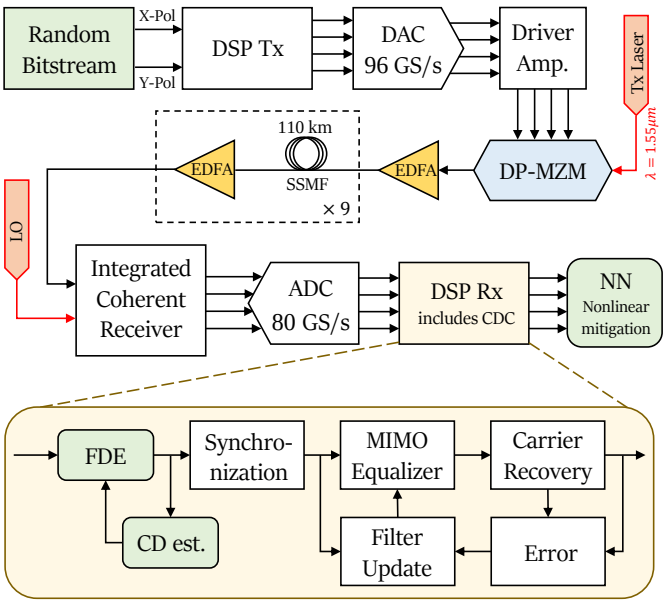



Fig. 4: Experimental setup where the data after the DSP Rx is fed into the NN as the input.

signal was transmitted along 9×110 km spans of SSMF with lumped EDFA amplification with the noise figure ranging from 4.5 to 5 dB. The SSMF had $\alpha$ = 0.21 dB/km, $D$ = 16.8 ps/(nm·km), and $\gamma$ = 1.14 (W· km)$^{-1}$.

At the receiver side, the received optical signal was converted to the electrical domain using an integrated coherent receiver. Then, the resulting signal was sampled at 80 GSamples/s using a digital sampling oscilloscope. The sampled signal was processed offline with the DSP algorithms described in [38]. The CDC was accomplished in two steps. The first step of processing involved compensating for bulk accumulated dispersion using a frequency domain equalizer (FDE) with an FFT size of 12288 samples and an impulse response length of 3072 samples [2]. After that the residual CD and dynamic impairments of channels were mitigated by the adaptive approach, multiple-input–multiple-output (MIMO) equalization with an FFT size of 192 samples and an overlap size of 48 samples. Next, the carrier frequency offset was mitigated. A constant-amplitude zero autocorrelation-based training sequence was located in the received frame, and the equalizer transfer function was estimated from it. Then, the two polarizations of the signal were demultiplexed, and clock frequency and phase offsets were corrected. The carrier phase estimation was performed using pilot symbols. The resulting soft symbols were used as input for an NN equalizer. Finally, the pre-FEC BER was evaluated based on the signal obtained at the output of the NN equalizer. In our case, the NN focused on mitigating the nonlinear effects, and was not designed to replace the regular DSP, instead, the NN was applied as an extra step to the regular DSP.

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

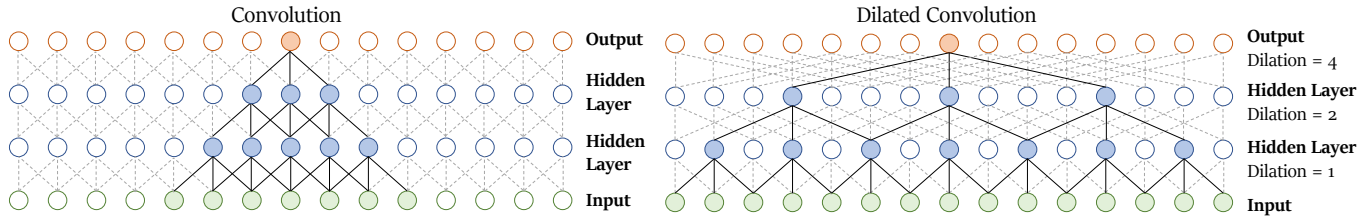
### *B. KD Training to Solve the Parallelization Problem of Recurrent Connection*

The KD framework is deployed to transform the model architecture from the biLSTM+CNN to simpler ones. Fig. 2a shows the KD process with the teacher and student model structure. If we observe the constellation diagram of the teacher labels and the ground-truth labels in Fig. 2a, training the student model with the teacher's predictions that contained the information on noise and some uncertainty, results in not overly confident predictions of the student. This helps reduce overfitting and improve the generalizability of the student model. We aim to compare different types of student models, namely, bidirectional RNN (biRNN), 1D-CNN, and MLP. 1D-CNN and MLP have a feed-forward structure, enabling parallel computation for the previously proposed biLSTM-based equalizer [12] or the teacher model. In contrast, biRNN still has a recurrent structure. However, biRNN architecture is considerably less complex than the biLSTM, thus allowing for faster computation. In this work, we limit the biRNN to only one layer in order to maintain the complexity.

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

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

*1) Teacher Network Architecture:* The teacher model is a biLSTM+CNN model, see Fig. 2a, which was trained




(a) Normal CNN. (b) Dilated CNN.

Fig. 5: Visualization of (a) a stack of convolutional layers; (b) a stack of 'dilated' convolutional layers.

previously in [12]. The biLSTM layer has 100 hidden units ($n_h$), and the CNN layer adopts 2 filters ($n_f$) and $n_k$ = 51 with the linear activation function. The loss function used in this model is MSE, and the optimizer is Adam with a learning rate of $10^{-3}$. The input window ($M$) has the size of 221 input symbols and the model performs a regression task to predict the real and imaginary parts of the recovered symbols, or 171 symbols per one inference step. For the teacher model employed in the experimental setup, the model has 117 hidden units and the output window is 195 symbols. More details can be found in Ref. [12].

*2) Student Network Architecture:* We investigated different types of the student model (1D-CNN, biRNN and MLP)[3] to evaluate the equalization performance, CC and inference speed. All student models in both simulation and experiment in this paper were trained with the loss function: Euclidean distance ($L_2$ distance).

For the proposed 1D-CNN model, the dilated CNN is applied. The dilated convolution is an approach to inflate the kernel by inserting holes between its consecutive elements. Consequently, the network is operated on a coarser scale than with a normal convolution filter with a dilation rate equal to zero [39]–[41]. This approach allows the NN to deal with long-term temporal dependencies, and have larger receptive fields within only a few layers as shown in Fig. 5. The dilated CNN preserves the input resolution throughout the network [42]. The dilated convolutions demonstrated the longer receptive field in a cheaper way than the LSTM [39]. In Ref. [10], the KD student model with dilated CNN architecture shows promising results when the teacher model is the LSTM architecture and the data is in the form of a time-series in the regression task. To mimic biLSTM, which learns the input data in forward and backward directions, the 1D-CNN student model learns both directions of the training data. The backward direction input means the input sequence (forward direction) in reverse time order. The last 1D-CNN layer of both the teacher and the student has the same parameters. The Bayesian Optimizer (BO) [4] is used to optimize the hyper-parameter values of the student model. The estimated optimal values found by BO are depicted in Fig. 2a. Note that (38, 23, 1) means that the 1D-CNN layer operates with 38 filters, a kernel size of 23, and a dilation rate of 1. The alpha value is 0.903. The activation function of the dilated 1D-CNN part is LeakyRelu [43]. Note that the architectures and parameters of the student models in the simulation and experimental setup are the same, apart from the second layer of the 1D-CNN, instead of 25 filters, it has 33 and 34 filters to maintain the output dimensions.

The vanilla RNN is the simplest variant of the recurrent-based models [44]. This RNN student model adopts one layer of biRNN with 135 hidden units followed by a 1D convolutional layer with 2 filters and a kernel size of 51 as the teacher model, see Fig. 2b. We avoid stacking the biRNN layers to maintain the inference speed, therefore, the performance can be limited. The training was carried out with the alpha value of 0.611, optimized by BO.

The last student model type is the MLP, shown in Fig. 2b, consisting of three hidden layers with the hidden units of 401, 510 and 510, respectively, and an output layer of 342 neurons. The output layer is reshaped to match the aforementioned output window shape of the data. The neurons in the hidden layers have the hyperbolic tangent as an activation function, while the output layer deploys a linear function. The alpha value is 0.8.

## V. Results and Discussion

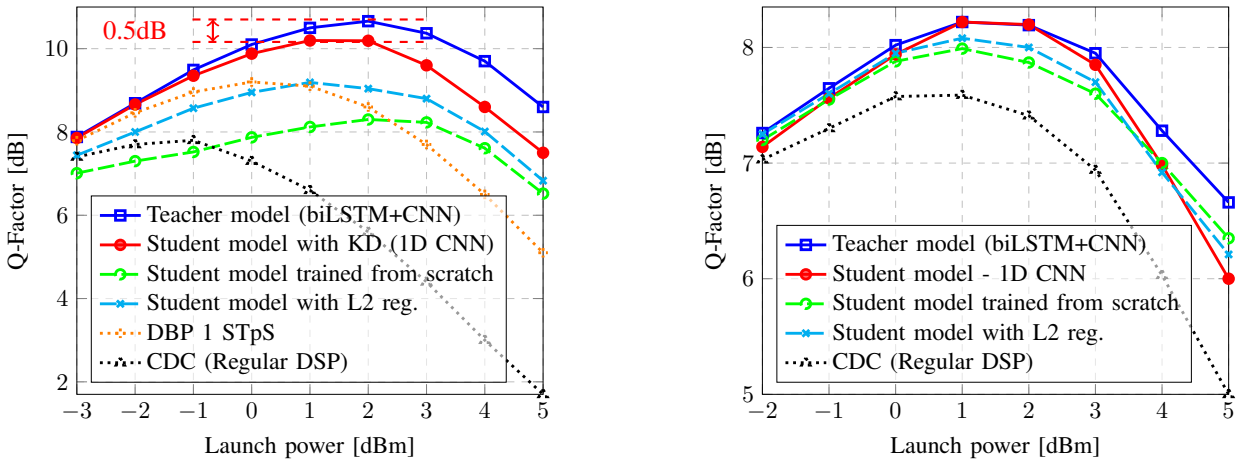
### A. Equalization Performance

In this paper, we showed the equalization performance in terms of Q-factor. The Q-factor was calculated directly from the bit error rate (BER) by:

$$Q = 20\log_{10}\left[\sqrt{2}\mathrm{erfc}^{-1}(2\mathrm{BER})\right]. \tag{7}$$

We compare our proposed student model (1D-CNN) using the KD framework with the teacher model (biLSTM +CNN), the student model trained from scratch (without KD) with exactly the same settings, and the student model trained from scratch with L2 regularizer [45]. The optimum L2 coefficient depends on the launch power. At 2 dBm launch power, the optimum L2 coefficient found by grid search is $10^{-4}$ for simulated data and $5 \times 10^{-6}$ for experimental data. Fig. 6a depicts Q-factor vs. launch power for different types of NN-based equalizers in the simulated data. In all NN-based equalizers, an improvement of the optimum launch power was achieved. We can observe that the Q-factor performance of the feed-forward student model with KD drops by 0.5 dB compared to the recurrent teacher model at its optimal launch

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




(a) SC-DP 30GBd; 64QAM; $20 \times 50$km SSMF link (Sim.). (b) SC-DP 34.4GBd; 64QAM-PS; $9 \times 110$km SSMF link (Exp.).

Fig. 6: Q-factor as a function of the launch power for the NN-based equalizers obtained via KD, compared to the original (teacher) model, CDC in three different transmission scenarios.

power (2 dBm). With KD, the performance of the student model is comparable to that of the teacher model in the linear transmission regime, but the student's performance degrades slightly as the launch power increases. However, when training the student model from scratch without KD, the model suffers from overfitting, resulting in a noticeable degradation of the peak performance by 2.4 dB at its optimal power. Training the student model with the L2 regularizer, which helps enhance the generalization capability, improves the performance of the NN compared to training it from scratch only, but still does not reach a similar performance level as the one achieved when the student model is trained with KD. The performance achieved using digital backpropagation (DBP) 1 step/span (STpS) and chromatic dispersion compensation (CDC) are also shown for reference.

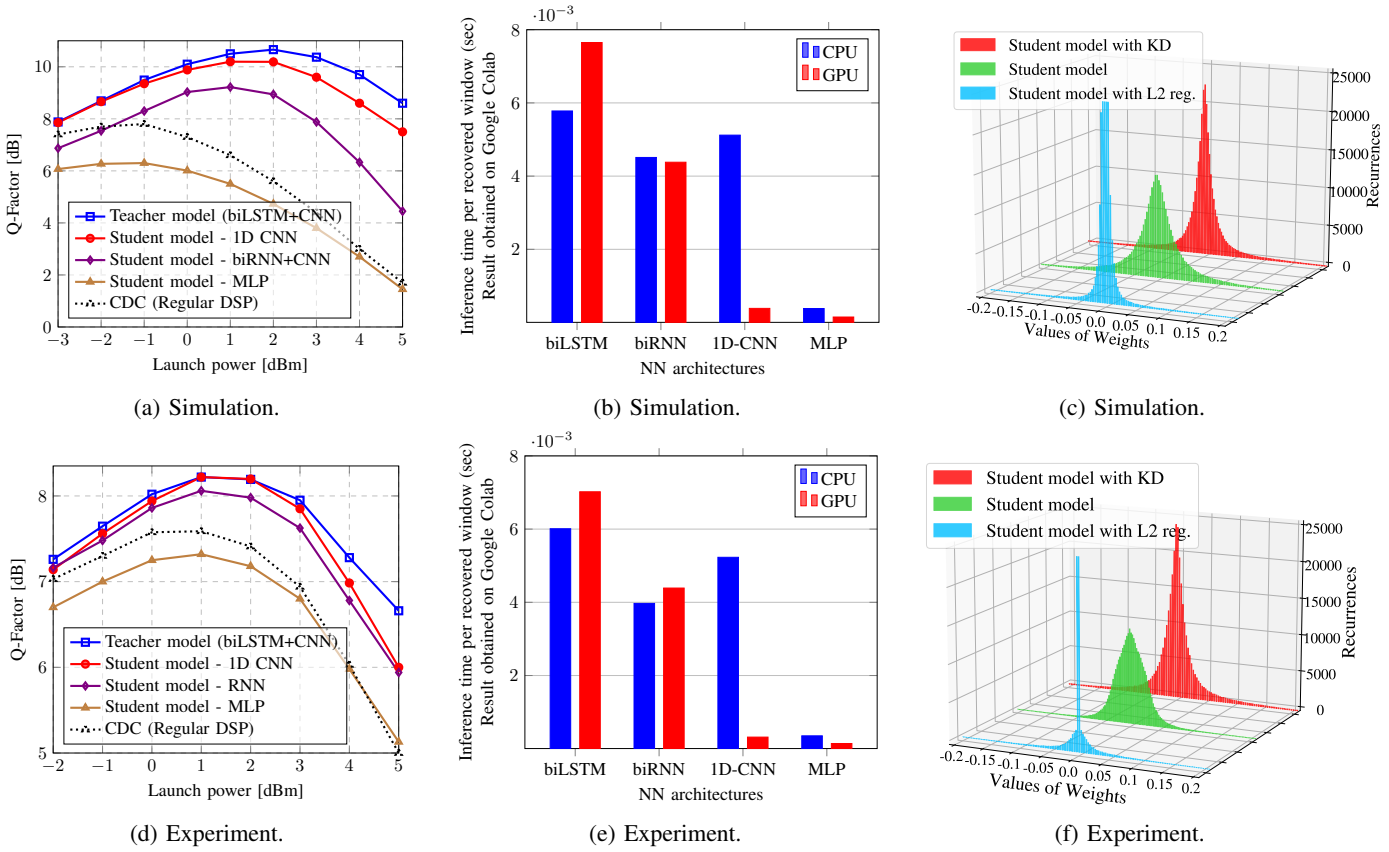
With the experimental setup, the performance of the student model with KD, shown in Fig. 6b, was comparable to the teacher model. We observed no performance drop in the experimental data. However, the student model trained from scratch did not suffer from severe overfitting as in the simulated data but it still could not reach the same Q-factor level as the teacher model or the student model trained with KD. In the case that the student model was trained from scratch with an L2 regularizer, the performance was improved slightly. When the model is not significantly overfitting, the L2 regularizer parameter needs to be carefully selected. In this experimental setup, the weaker regularization parameter was preferred ($5 \times 10^{-6}$), to prevent the regularizer from excessively penalizing the weights, allowing the model to still learn meaningful patterns from the data [46]. This result demonstrated that the proposed student network trained with KD was highly effective in the experimental data and the KD also maintained superior performance compared to the student models trained from scratch. However, it can be observed that at the higher launch power, the performance gap was larger. At the launch power of 5 dBm, the model with KD and the student model with L2 regularize performed worse than the student model trained from scratch. This can be because, during the optimization process, we operated with the dataset with the launch power of 2 dBm, resulting in a sub-optimal performance at 5 dBm and overly constrained weight distribution.

To demonstrate the effectiveness of the proposed 1D-CNN student model, the 1D-CNN is compared against the biRNN and MLP as student models, shown in Fig. 7a and Fig. 7d for the simulated and experimental data, respectively. The performance of the biRNN and MLP students was not comparable to the 1D-CNN model. In the simulation, the biRNN model had around 1.4 dB Q-factor drop from the teacher model but still outperformed the CDC and showed the improvement of the optimum launch power. In the experiment, the biRNN revealed the same behavior as in the simulation. Even the biRNN has a recurrent structure, but as we implemented only one layer to limit the inference latency, the model did not learn well. In addition, we experienced instability during the training, which can result from vanishing/exploding gradients. In the case of the MLP as a student model[4], the performance was poorer than that of the CDC in both simulation and experiment. The MLP was not the most suitable architecture for nonlinear mitigation when considering the performance [47], especially for recovering the multi-symbol output as in our case. This can be explained by considering that the MLP lacks temporal information handling by design. The MLP does not take the sequence of data points into account, which limits their capability to effectively model time series dynamics.

### B. Inference Speed Performance

For this analysis, we tested the inference speed with the simulation data and experimental data. Note that both the teacher and the student models recover 171 symbols per

[4]Note that, in the optimization process, we attempted to increase the number of layers to experience if that will enhance the performance, however, the MLP reached the point where the greater number of hidden layers did not improve the performance .




(a) Simulation. (b) Simulation. (c) Simulation.

(d) Experiment. (e) Experiment. (f) Experiment.

Fig. 7: (a, d) Comparison of different architectures (biRNN+CNN, 1D-CNN and MLP) of the student model to the teacher model, CDC and 1 STpS DBP; (b, e) Inference time of teacher/student models; (c, f) Weight distribution of student model trained with different approaches.

inference step in the simulation, and 195 symbols per inference step in the experiment. The inference time analysis was carried out by using CPU (Intel Xeon Processor 2.20 GHz) and GPU (Tesla T4) on Google Colab [48][5] as the inference engine. In this analysis, the size of the test set was 800, and the batch size was 8. As can be seen from Fig. 7b for the simulation and Fig. 7b for the experiment, the biLSTM-based NNs require the longest inference time in both CPU and GPU as inference engines. The biRNN presented lower inference latency than the biLSTM due to its simpler architecture. The inference time of the recurrent-based NN in the CPU and the GPU did not differ significantly. In contrast, the feed-forward NNs (1D-CNN and MLP) have significantly lower inference latency in the GPU than that of the CPU, especially the 1D-CNN model. 1D-CNN model when the CPU was used as the inference engine experienced similar latency as the recurrent-based NN. However, with the feed-forward nature of the 1D-CNN that allows parallelization, the GPU can demonstrate remarkably faster computation. The MLP's inference time is shorter than the 1D-CNN as it has a simpler structure and operations. The recurrent networks (biLSTM and biRNN) experienced longer inference latency in the GPU compared to the CPU. This can happen due to the not easily parallelizable recurrent structure and the complexity of the model. The GPUs are specialized for efficiently parallel computing, handling complex models with multiple layers and parameters. The GPUs are generally faster when the computation is parallelizable and involves matrix multiplications while in some other types of computations, the GPU can be slower. The parallel computing ability of the GPU can fully exploit the parallelizability of the feed-forward structures of 1D-CNN and MLP. Overall, the proposed 1D-CNN student model provided the most reasonable trade-off between performance and inference speed. The parallelization of the proposed feed-forward equalizer and its savings in latency are key to the real-time hardware implementation of NN-based equalizers.

Table I presents the summary of the performance versus complexity of different NN architectures. The complexity is shown in terms of CC and time complexity. The CC was measured with respect to the number of trainable parameters and the number of real multiplications, while the time complexity is the latency or the inference time on CPU and GPU. The biLSTM model provided the highest Q-factor but only slightly higher than the 1D-CNN in the simulation, while the performance of both biLSTM and 1D-CNN models are comparable in the experiment. Even though the feed-forward structures have a larger number of trainable parameters, the inference latency is still lower, especially when it is processed on the GPU. This result demonstrates that recurrent connections can crucially limit parallelization.

[5]Note that, in this study, we did not consider the GPU-accelerated library of primitives for deep neural networks cuDNN (NVIDIA CUDA®. Deep Neural Network library) for the inference in the GPU.

| Data/ Output Shape | Model Type | Q-factor | No. of Trainable Parameters | No. of Real Multiplications | CPU Inference Time per Window | GPU Inference Time per Window |
|---|---|---|---|---|---|---|
| Simulation 171 Symbols | biLSTM+CNN | **10.66** | 104,402 | $2.2\times10^{7}$ | $5.78\times10^{-3}$ | $7.65\times10^{-3}$ |
| | biRNN+CNN | 9.22 | **65,342** | $8.85\times10^{6}$ | $4.51\times10^{-3}$ | $4.38\times10^{-3}$ |
| | 1D-CNN | 10.19 | 293,390 | $6.37\times10^{7}$ | $5.12\times10^{-3}$ | $3.87\times10^{-4}$ |
| | MLP | 6.3 | 995,277 | $\mathbf{9.93}\times10^{5}$ | $\mathbf{3.83}\times10^{-4}$ | $\mathbf{1.51}\times10^{-4}$ |
| Experiment 195 Symbols | biLSTM+CNN | 8.22 | 126,830 | $2.7\times10^{7}$ | $6.01\times10^{-3}$ | $7.02\times10^{-3}$ |
| | biRNN+CNN | 8.06 | **52,382** | $9.52\times10^{6}$ | $3.97\times10^{-3}$ | $4.39\times10^{-3}$ |
| | 1D-CNN | **8.22** | 309,870 | $6.8\times10^{7}$ | $5.23\times10^{-3}$ | $3.18\times10^{-4}$ |
| | MLP | 7.32 | 1,019,805 | $\mathbf{1.07}\times10^{6}$ | $\mathbf{3.525}\times10^{-4}$ | $\mathbf{1.43}\times10^{-4}$ |

TABLE I: Summary of the performance versus complexity of different types of NN architecture.

### C. Roles of Knowledge Distillation

Now, we study the features associated with the KD-trained model. For this purpose, we also report the weight distribution of the student model trained with different approaches in Fig. 7c for the simulation and Fig. 7f for the experiment. In both figures, compared to the student model trained from scratch, the student model with KD has a more regularized weight distribution: the weights are more concentrated around zero. This characteristic helps reduce the model's variance and overfitting. The optimal value of $\alpha$ in the KD loss function is 0.903, which means that the student model learns 90.3% from the teacher labels and the rest comes from the ground-truth labels. This fact demonstrates the effectiveness of the teacher labels in the student's learning. The teacher constellation/labels depicted in Fig. 2a show that the teacher also provides helpful information on the noise, whereas this information cannot be encoded in the ground-truth labels (which contain only real values). The weight distribution of the student model with KD and the improvement in Q-factor, compared to the training of the 1D-CNN without KD, both support the concept of using teacher labels as efficient regularizers [9].

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

## VI. Conclusion

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

## References

[1] T. Xu, N. A. Shevchenko, Y. Zhang, C. Jin, J. Zhao, and T. Liu, "Information rates in kerr nonlinearity limited optical fiber communication systems," *Optics Express*, vol. 29, no. 11, pp. 17 428–17 439, 2021.

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

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

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

[5] S. Deligiannidis, A. Bogris, C. Mesaritakis, and Y. Kopsinis, "Compensation of fiber nonlinearities in digital coherent systems leveraging long short-term memory neural networks," *Journal of Lightwave Technology*, vol. 38, no. 21, pp. 5991–5999, 2020.

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

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

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

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

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

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

[12] P. J. Freire, A. Napoli, D. A. Ron, B. Spinnler, M. Anderson, W. Schairer, T. Bex, N. Costa, S. K. Turitsyn, and J. E. Prilepsky, "Reducing computational complexity of neural networks in optical channel equalization: From concepts to implementation," *Journal of Lightwave Technology*, pp. 1–26, 2023.

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

[14] R. A. Hamad, M. Kimura, L. Yang, W. L. Woo, and B. Wei, "Dilated causal convolution with multi-head self attention for sensor human activity recognition," *Neural Computing and Applications*, vol. 33, no. 20, pp. 13 705–13 722, 2021.

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

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

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

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

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

[20] K. K. Parhi, "Chapter 10: Pipelined and parallel recursive and adaptive filters," http://people.ece.umn.edu/users/parhi/SLIDES/chap10.pdf, 1999, accessed: 2022–08-03.

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

[22] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[23] B. Karanov, M. Chagnon, V. Aref, F. Ferreira, D. Lavery, P. Bayvel, and L. Schmalen, "Experimental investigation of deep learning for digital signal processing in short reach optical fiber communications," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2020, pp. 1–6.

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

[25] Z. Xu, S. Dong, J. H. Manton, and W. Shieh, "Low-complexity multi-task learning aided neural networks for equalization in short-reach optical interconnects," *Journal of Lightwave Technology*, vol. 40, no. 1, pp. 45–54, 2021.

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

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

[28] M. Takamoto, Y. Morishita, and H. Imaoka, "An efficient method of training small models for regression problems with knowledge distillation," in *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE, 2020, pp. 67–72.

[29] J. Xiang, S. Colburn, A. Majumdar, and E. Shlizerman, "Knowledge distillation circumvents nonlinearity for optical convolutional neural networks," *Applied Optics*, vol. 61, no. 9, pp. 2173–2183, 2022.

[30] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[31] H. Ma, S. Yang, R. Wu, X. Hao, H. Long, and G. He, "Knowledge distillation-based performance transferring for lstm-rnn model acceleration," *Signal, Image and Video Processing*, pp. 1–8, 2022.

[32] M. R. U. Saputra, P. P. De Gusmao, Y. Almalioglu, A. Markham, and N. Trigoni, "Distilling knowledge from a deep pose regressor network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 263–272.

[33] A. K. Menon, A. S. Rawat, S. J. Reddi, S. Kim, and S. Kumar, "Why distillation helps: a statistical perspective," *arXiv preprint arXiv:2005.10419*, 2020.

[34] N. Gautam, V. Kaushik, A. Choudhary, and B. Lall, "Optidistillnet: Learning nonlinear pulse propagation using the student-teacher model," *Optics Express*, vol. 30, no. 23, pp. 42 430–42 439, 2022.

[35] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," *Advances in neural information processing systems*, vol. 30, 2017.

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

[37] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[38] M. Kuschnerov, M. Chouayakh, K. Piyawanno, B. Spinnler, E. De Man, P. Kainzmaier, M. S. Alfiad, A. Napoli, and B. Lankl, "Data-aided versus blind single-carrier coherent receivers," *IEEE Photonics Journal*, vol. 2, no. 3, pp. 387–403, 2010.

[39] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[40] X. Zhen, R. Chakraborty, N. Vogt, B. B. Bendlin, and V. Singh, "Dilated convolutional neural networks for sequential manifold-valued data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10 621–10 631.

[41] S. Gong, Z. Wang, T. Sun, Y. Zhang, C. D. Smith, L. Xu, and J. Liu, "Dilated fcn: Listening longer to hear better," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 254–258.

[42] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions."

[43] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.

[44] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning internal representations by error propagation," 1985.

[45] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 78.

[46] B. J. Kim, H. Choi, H. Jang, D. G. Lee, W. Jeong, and S. W. Kim, "Guidelines for the regularization of gammas in batch normalization for deep residual networks," *arXiv preprint arXiv:2205.07260*, 2022.

[47] P. J. Freire, Y. Osadchuk, B. Spinnler, W. Schairer, A. Napoli, N. Costa, J. E. Prilepsky, and S. K. Turitsyn, "Experimental study of deep neural network equalizers performance in optical links," in *Optical Fiber Communication Conference (OFC) 2021*. Optica Publishing Group, 2021, p. M3H.2. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=OFC-2021-M3H.2

[48] "Google colab," https://colab.research.google.com/, accessed: July 31, 2023.