Extrinsic Neural Network Equalizer for Channels with High Inter-Symbol-Interference

1st Xiang Huang
Dept. of ECE
Univ. of Utah
eric.xiang.huang@utah.edu

2nd Joohyun Cho
Dept. of ECE
Univ. of Utah
joohyun.cho@utah.edu

3rd Kazem Hashemizadeh
Dept. of ECE
Univ. of Utah
s.hashemizadehkolowri@utah.edu

4th Rong-Rong Chen
Dept. of ECE
Univ. of Utah
rchen@ece.utah.edu

ICC 2021 - IEEE International Conference on Communications | 978-1-7281-7122-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICC42927.2021.9500903

***Abstract*—In this paper, we propose a novel extrinsic neural network equalizer (ExNE) for joint iterative equalization and decoding. The proposed ExNE takes the received signal sequence and *a priori* probabilities from the channel decoder as inputs to directly generate output extrinsic probabilities. This approach improves the performance of iterative equalization and decoding by making explicit use of extrinsic information. A three-step, open-loop neural network (NN) training procedure is developed for the ExNE, independent of the choice of channel code. We propose a new NN architecture termed deep concatenated convolutional blocks with skip connections (DCCB-SC) for ExNE which attains excellent performance with only a moderate number of network parameters. For challenging linear and non-linear inter-symbol-interference (ISI) channels considered in this work, the proposed ExNE approaches the performance of the maximum *a posteriori* probability (MAP) equalizer without assuming prior knowledge of the channel model.**

***Index Terms*—turbo equalization, neural network, inter-symbol-interference, iterative decoding, extrinsic information.**

I. Introduction

Recently, the use of neural networks (NN) in communication systems, including NN based equalization and decoding over inter-symbol-interference (ISI) channels, has shown promising results [1]–[8]. Compared to conventional equalizers such as minimum mean square error (MMSE) equalizer or the maximum *a posteriori* probability (MAP) equalizer, a major advantage of the NN equalizer is that it does not require prior knowledge of the channel model. It is applicable to unknown linear and nonlinear channels to combat ISI caused by channel dispersion. In this work, we design a new class of NN equalizer, termed extrinsic neural network equalizer (ExNE), specifically for the application of turbo equalization over ISI channels. Turbo equalization [9] is a powerful technique to improve detection performance via joint iterative equalization and decoding. In this setting, soft information in terms of probabilities, or equivalently, the log-likelihood ratios (LLRs) of transmitted bits are exchanged between the soft-input and soft-output (SISO) equalizer and the decoder over iterations to improve data reliability. It is important to note that extrinsic probabilities, rather than *a posteriori* probabilities (APPs) should be used to ensure good convergence of the iterative processing. However, to the best of our knowledge, there are only very limited studies [10] on NN equalizer designs that use extrinsic probabilities. The proposed ExNE is designed specifically to produce extrinsic probabilities required for turbo equalization. A novel three-step training procedure is developed which utilizes a preliminary APP-based NN equalizer to train the proposed ExNE. The resulting ExNE generates extrinsic probabilities directly.

Our work focuses on the design of a stand alone SISO NN equalizer that can be plugged in a coded system for turbo equalization. Given the operating signal-to-noise power ratio (SNR), the training of the proposed ExNE is independent of the choice of the channel code. In contrast, most works in the literature on NN-based equalizer design is either uncoded [6], considers a one-time equalization and decoding (without turbo equalization) [3], [5], [7], or targets a joint design that combines the tasks of equalization and decoding [2], [4]. In [2], a blind turbo equalizer based on a convolutional neural network (CNN) is designed under an unsupervised setting. In [4], a turbo equalizer/decoder is built jointly based on a recursive neural network (RNN) architecture. While both designs of [2], [4] depend on the specific choice of the channel code, the proposed ExNE does not impose such a constraint, provided that the range of the operating SNR is given. We note that a stand alone turbo equalizer is considered in [10] for fiber-optic nonlinearity compensation. While [10] also considers the use of extrinsic probabilities, our proposed approach is different in that we explicitly remove prior information from NN training to ensure that the generated probabilities are extrinsic.

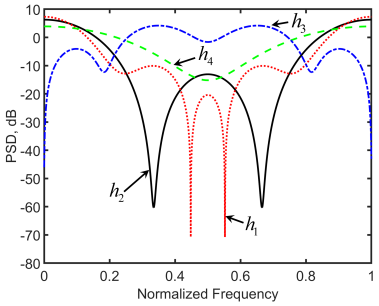


Fig. 1: Channels with high, moderate, or mild ISI.

In this work, we consider challenging channels with high ISI and use them to examine the effectiveness of ExNE. In Fig. 1, we plot the frequency responses of four linear channels considered in this work. The channels are ordered from the most challenging to the least. Channels $\mathbf{h_1}$ and $\mathbf{h_2}$, taken from

This work is supported by NSF Grants 1817154 and 1824558.

[11], have deep and wide spectral nulls. They feature high ISI and thus are the most challenging to equalize. Besides $\mathbf{h_1}$ and $\mathbf{h_2}$, we also consider two representative channels used in existing NN equalizer designs: $\mathbf{h_3}$ and $\mathbf{h_4}$. Channel $\mathbf{h_3}$ from [2] features a moderate ISI due to a narrow spectral null at the edge of the frequency response. In comparison, the commonly used channel $\mathbf{h_4}$ from [3]–[5] has no deep spectral nulls, which indicates a more benign characteristic with mild ISI. Numerical results in Section IV show that the proposed ExNE can approach the performance of the MAP equalizer for these channels without any prior knowledge of the channels.

We summarize main contributions of this work as follows.

- We propose an original extrinsic NN-based equalizer (ExNE) for turbo equalization. A novel three-step training procedure is developed to train the ExNE that directly generates extrinsic LLR for joint iterative equalization and decoding using the received signal and *a priori* LLRs generated from an open-loop simulation.
- We propose a powerful NN architecture, termed deep concatenated convolutional blocks with skip connections (DCCB-SC) for ExNE to increase the reception view of the NN while allowing efficient feature extraction with a moderate network size. This enables ExNE to achieve a performance close to the MAP equalizer over channels with high ISI.
- Different from existing work on NN-based equalizers for turbo equalization, the proposed ExNE is trained using an open-loop simulation, *independent* of the choice of channel codes. A simple adaptive decoder scaling procedure is proposed to scale the range of the decoder LLR over iterations to match that of the training data. This improves the performance of ExNE and provides great flexibility in its application to turbo equalization.
- We provide a comparison of the proposed ExNE with MAP and turbo MMSE equalizers over challenging ISI channels that were not considered in prior work. This demonstrates ExNE as a strong candidate for turbo equalization over unknown channels with high ISI.

II. Background

A. Turbo equalization and decoding

A system block diagram of turbo equalization and decoding is shown in Fig. 2. At the transmitter side, a sequence of information bits $\mathbf{m}$ is encoded and mapped to a sequence of binary phase shift keying (BPSK) symbols $\mathbf{x} = (\mathbf{x_1}, \cdots, \mathbf{x_N})$. Here, N is the number of bits in each codeword. The coded sequence $\mathbf{x}$ is then passed through an ISI channel and the received signal sequence is given by $\mathbf{y} = (\mathbf{y_1}, \cdots, \mathbf{y_N})$. We consider a general ISI channel model as in [2], [4] which has the form of $\mathbf{y} = \mathbf{g}(\mathbf{x} * \mathbf{h}) + \mathbf{w}$, or equivalently,

$$y_n = g(\sum_{k=0}^{L-1} x_{n-k} h_k) + w_n, \quad n = 1, \cdots, N. \tag{1}$$

In (1), $\mathbf{h} = [h_0, h_1, \cdots, h_{L-1}]$ is the channel impulse response, L is the channel length. The noise sequence $\mathbf{w} = [w_1, \cdots, w_N]$ is assumed to be independent identically distributed Gaussian random variables. The scalar function $g(\cdot)$, applied component-wise on the convolution $x * h$, represents the non-linearity of the channel. When $g(u) = u$, the model in (1) reduces to a standard linear ISI channel. Joint turbo

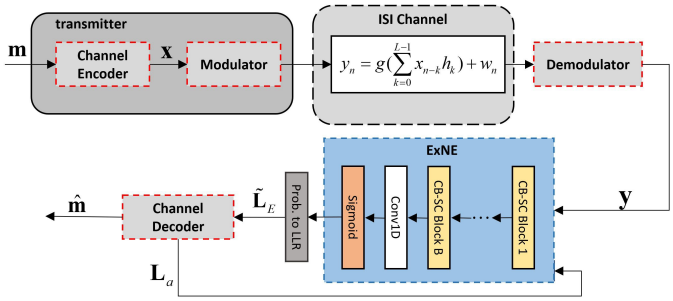


Fig. 2: A system diagram for turbo equalization and decoding.

equalization and decoding is performed at the receiver side. The inputs to ExNE are $\mathbf{y}$ and the *a priori* probabilities of N transmitted bits, produced by the decoder. These are denoted by LLRs $\mathbf{L_a} = (L_a(1), L_a(2), \cdots, L_a(N))$, where $L_a(i) = \log \frac{P(x_i=+1)}{P(x_i=-1)}$. At the beginning of each iteration, the equalizer processes its inputs to suppress the ISI and generates refined estimates of bit conditional probabilities $P(x_i = \pm 1 | \mathbf{y}, \mathbf{L_a})$. Subsequently, these probabilities are passed to the channel decoder for the next iteration of channel decoding. This way, joint equalization and decoding proceeds in an iterative fashion until the receiver converges or a pre-determined number of iterations is reached.

B. MAP equalizer

The MAP equalizer [9] first employs the BCJR algorithm to compute the APPs for each bit $i = 1, \cdots, N$ according to

$$P(x_i = \pm 1 | \mathbf{y}, \mathbf{L}_a) \propto \sum_{\mathbf{x}: x_i = \pm 1} P(\mathbf{y} | \mathbf{x}, \mathbf{L}_a) \prod_{j=1}^{N} P(x_j), \tag{2}$$

where the summation is over all possible sequences $\mathbf{x}$ such that $x_i = +1$ or -1. The *a priori* probabilities $\{P(x_j), j = 1, \cdots, N\}$ are assumed to be independent. Subsequently, it generates extrinsic LLRs $\mathbf{L}_E(i)$ for each bit i by removing the contribution of the *a priori* LLR $L_a(i)$ from (2). This is done simply by subtraction:

$$L_E(i) = \log \frac{P(x_i = +1 | \mathbf{y}, \mathbf{L}_a)}{P(x_i = -1 | \mathbf{y}, \mathbf{L}_a)} - L_a(i). \tag{3}$$

These extrinsic LLRs are then passed to the decoder. The use of extrinsic LLRs in the form of (3) is essential to ensure the success of joint iterative equalization and decoding. This motivates our work to design extrinsic NN equalizer for turbo equalization. We note that for the MAP equalizer, extrinsic LLRs can be obtained by simply subtracting *a priori* LLRs from APPs (see (3)). However, the same approach applied to a NN equalizer does not guarantee that true extrinsic LLRs will be produced. The main reason is that a NN equalizer does not necessarily closely approximate a MAP equalizer (especially for challenging ISI channels), and thus, simply

subtracting *a priori* LLRs cannot remove the effect of the priors completely. This leads to our proposed design that ensures complete removal of the *a priori* LLRs.

III. Proposed Extrinsic Neural Network Equalizer (ExNE)

In this section, we present the proposed ExNE design. We will first describe a novel three-step, open-loop training procedure that allows ExNE to generate extrinsic LLRs. Next, we will discuss details of generating prior LLRs in the open-loop training. This is followed by a description of the proposed DCCB-SC network architecture for ExNE. Finally, we discuss an adaptive decoder scaling procedure to match the range of close-loop decoder LLRs with that of the training data.

A. Three-step, open-loop training in ExNE

The three-step training for ExNE is illustrated in Fig. 3.

Step 1: Train an APP-based NN equalizer (ApNE). We train a preliminary, APP-based NN equalizer, indicated by "ApNE" in Fig. 3. Note that ApNE uses the same DCCB-SC architecture as in the final ExNE architecture shown in Step 3. Here, each training sample (vector) contains the received signal vector $\mathbf{y}$ and a bit-wise *a priori* LLR vector $\mathbf{L_a}$, each of length m. Here, we choose $m \ll N$. . The output is a bit-wise *a posteriori* LLR vector $\mathbf{L_A}$ of the same length. To generate each training sample, a random bit sequence $\mathbf{x}$ of length m is generated first and then used to generate $\mathbf{y}$ according to (1). For a given $\mathbf{x}$, each component $L_a(i), i = 1, \cdots, m$ in $\mathbf{L_a}$ is generated following the method described in Section III-B. Since BPSK modulation is used, we map each BPSK symbol $x_i = +1$ to $b_i = 1$ and $x_i = -1$ to $b_i = 0$. We use the binary cross entropy (BCE) loss function $\mathcal{L}_A = -\frac{1}{m}\sum_{i=1}^{m}[b_i \cdot \log P_A(i) + (1-b_i)\cdot \log(1-P_A(i))]$, where $P_A(i)$ is the probability that $x_i = +1$, to match the output *a posteriori* probability $\mathbf{P_A}$ with true values of the training bits. We then convert $\mathbf{P_A}$ to LLRs $\mathbf{L_A}$ by $L_A(i) = \log \frac{P_A(i)}{1-P_A(i)}$, which are fed to the decoder.

Step 2: **Generate extrinsic training LLRs from ApNE**. As shown in Fig. 3, we run m *modified* sample vectors (all from the same $\mathbf{L}_a$) in parallel, using the ApNE obtained from Step 1. When processing the i-th sample vector, the inputs are $\mathbf{y}$ and the modified *a priori* vector $\widetilde{\mathbf{L}}_a^i$, obtained from $\mathbf{L}_a$ by setting its i-th component to be zero. Specifically, we define

$$\widetilde{L}_a^i(k) = \begin{cases} L_a(k) & \text{if } k \neq i \\ 0 & \text{if } k = i. \end{cases} \tag{4}$$

This way, for each bit i, the output $L_E^i(i)$ of the ApNE is independent of its original prior $L_a(i)$ because $\widetilde{L}_a^i(i)$ is set to be zero at the input. Hence, $L_E^i(i)$ will represent the "extrinsic LLR" of bit i after equalization. At the end of Step 2, we collect LLRs $\{L_E^i(i), i = 1, \cdots, m\}$, one from each output vector and use these as targets for training in Step 3.

Step 3: Train the final ExNE. In this step, we train the final ExNE architecture to directly map $\mathbf{y}$ and $\mathbf{L}_a$ to the extrinsic LLR vector $\widetilde{\mathbf{L}}_\mathbf{E}$. The network is trained so that $\widetilde{\mathbf{L}}_\mathbf{E}$ closely approximates the target $\mathbf{L_E}$ that comes from Step 2. Here, we

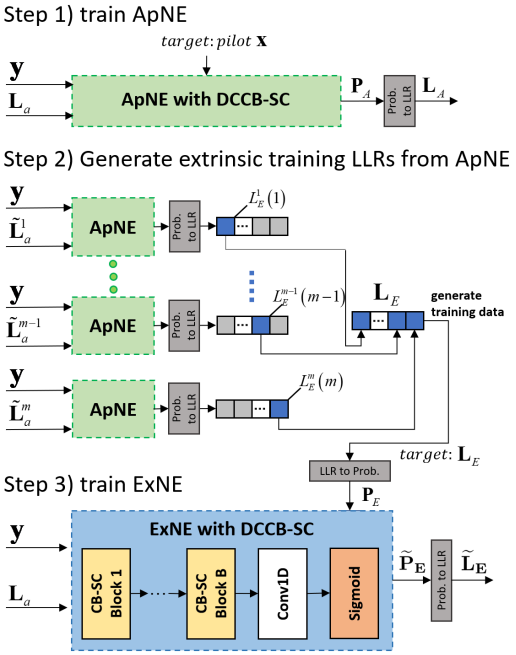


Fig. 3: Three-step, open-loop training of ExNE.

again use the BCE loss function $\mathcal{L}_E = -\frac{1}{m}\sum_{i=1}^{m}\left[P_E(i)\cdot \log \widetilde{P}_E(i) + (1-P_E(i))\cdot \log(1-\widetilde{P}_E(i))\right]$ and convert $\widetilde{\mathbf{P}}_\mathbf{E}$ to $\widetilde{\mathbf{L}}_\mathbf{E}$ in the same way as in Step 1.

B. Generate open-loop a priori LLRs for training samples

Next, we describe how to generate $\mathbf{L_a}$ in Step 1 of open-loop ExNE training. Given $\mathbf{x}$, each $L_a(i), i = 1, \cdots, m$ in $\mathbf{L_a}$ is generated following the open-loop simulation method of the extrinsic information transfer (EXIT) curve [12]. Specifically, we characterize the quality of $\mathbf{L_a}$ using a parameter $I_A \in [0, 1]$, which is the mutual information (MI) between a bit and its *a priori* LLR. Given I_A, we randomly generate $L_a(i)$ from a Gaussian distribution $N(\pm\mu, \sigma^2)$ if $x_i = \pm 1$. Here, we set $\sigma^2 = 2\mu$ and σ^2 is computed from I_A so that the MI between x_i and $L_a(i)$ equals I_A [12]. In summary, to generate each sample vector, we first pick an I_A uniformly from $[0, 1]$ and keep it fixed. Next, we calculate σ^2 for this I_A, and then follow the distribution $N(\pm\sigma^2/2, \sigma^2)$ to generate $\mathbf{L_a}$ for all bits in this training sample. The value of I_A changes from one sample vector to the next. This design exploits the fact that in a close-loop system, the quality of $\mathbf{L_a}$ produced by the channel decoder improves over iterations. Typically, I_A increases from 0, before the start of channel decoding, to a value close to 1, when the decoder converges. Within each codeword, the qualities of LLRs are similar, corresponding to the same I_A. Histograms of $\mathbf{L_a}$ obtained in an open-loop simulation are shown in Fig. 5 (a). The top subfigure shows the combined histogram of $\mathbf{L_a}$ when I_a is randomly chosen from $[0, 1]$. The middle and bottom histograms are for $I_a = 0.2$ and $I_a = 0.8$, respectively. The distribution of $\mathbf{L_a}$ generated using a larger I_a better approximates the distribution of decoder LLRs from later iterations.

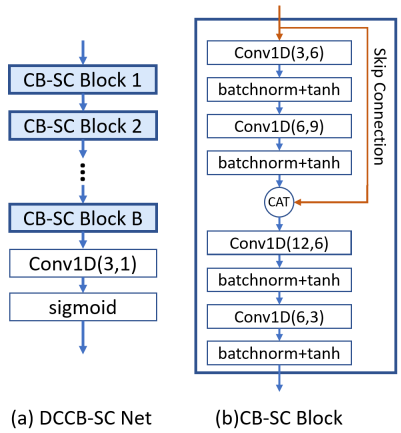


Fig. 4: The proposed DCCB-SC network for ExNE. The "CAT" operator in (b) denotes the concatenation.

C. A DCCB-SC network

In this section, we describe the proposed deep concatenated convolutional blocks with skip connections (DCCB-SC) network for ExNE. As shown in Fig. 4(a), the DCCB-SC consists of B convolutional blocks with skip connections(CB-SC). Let Conv1D(M,N) denote a 1D convolutional layer with M input and N output features. At the last layer of DCCB-SC, a Conv1D(3,1) layer with sigmoid activation is used to map 3 component features to an output probability. In total, the network has $4\times B+1$ convolutional layers. As shown in Fig. 4 (b), each CB-SC block has 4 convolutional layers. The skip connection concatenates 3 network inputs with 9 outputs of 2nd layer to obtain 12 inputs to 3rd layer. In each convolutional layer, kernels of size 1x5 with stride 1 and padding size of 2, batch normalization, and tanh activation are used.

Next, we discuss main features of DCCB-SC and highlight how we adapt this network to turbo equalization.

- *A deep network.* In DCCB-SC, we adopt a deep architecture to increase the receptive field of the network. This is crucial in turbo equalization where the LLR of each transmitted bit depends on the entire sequence of received signal $\mathbf{y}$ and the *a priori* probabilities of all transmitted bits (see Section II-B). Using a deep architecture with concatenated CB-SC blocks allows the network to maintain a small number of features in each block while achieving a sufficiently large receptive field for feature extraction. In our study, we observe that the network depth depends on the level of ISI. Either $B=3$ (for channels with moderate ISI) or $B=4$ (for channels with high ISI) is sufficient to obtain good performance.
- *Skip connection.* As shown in Fig. 4 (b), we use a skip connection (SC) within each CB-SC to combat the vanishing gradient problem and to improve the efficiency of the back propagation during network training. Here, we adopt a concatenated SC [13] rather than a residual connection [2] due to its greater flexibility in building network connection. The concatenated SC also provides feature re-usability by combining features from earlier layers with those of later layers. This allows useful information to be retained from previous layers so that better predictions can be made at later layers.
- *Tanh activation.* In DCCB-SC, we use the tanh activation function because it maintains the symmetry of the data, see Fig. 5(b) for the near symmetric distribution of the LLRs coming out of the ExNE. We find that when other activation functions such as the rectified linear unit (ReLU) is used, this symmetry is no longer maintained, and thus performance is degraded. The use of tanh in a NN equalizer is also proposed in [2], even though the issue of symmetry is not discussed therein.
- *Batch normalization.* We find that adding batch normalization prior to the activation function improves the network training. The batch normalization step helps control the input distributions to the convolutional layers. Note that batch normalization before the sigmoid function in the last layer is not performed.

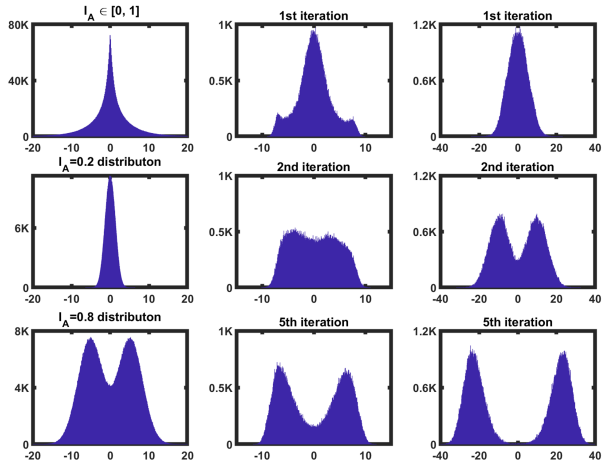
Compared to existing architectures of NN equalizers [2]–[5], the proposed DCCB-SC features the deepest architecture of 13 or 17 layers, while requiring only a moderate network size of 2743 ($B=3$) or 3640 ($B=4$) parameters. This deep architecture, in combination with concatenated SC and tanh activation, demonstrates a strong expressive power and can achieve near optimal performance for channels with high ISI (e.g. $\mathbf{h_1}$, $\mathbf{h_2}$) under turbo equalization. The proposed DCCB-SC utilizes the *a priori* probabilities from the decoder efficiently, without assuming prior knowledge of the channel or the choice of channel code. This is different from the NN equalizer of [2] in which the loss function used in the equalizer training assumes knowledge of coding constraints.

D. Scaling of decoder LLR to match ExNE training

As described in Section III-B, we adopt an open-loop training procedure to generate the *a priori* LLRs $\mathbf{L_a}$. A histogram of the generated L_a used for training is shown in Fig. 5 (a). We see that there exists some ℓ such that with a high probability ($>97\%$), L_a falls in $[-\ell,\ell]$. Since the distribution of L_a used in training does not depend on the choice of channel code, or the specific LLR distribution of the decoder at a given iteration, it is necessary to scale the decoder LLR appropriately so that the resulting LLR after scaling matches the range of $[-\ell,\ell]$ for the training data.

We propose a simple adaptive decoder scaling procedure to scale the range of the decoder LLR over iterations. As the number of iteration increases, the decoder becomes more confident about its estimates, and hence, the mean value of the LLRs will increase over iterations. In our implementation, at the end of each equalization and decoding iteration, we first check the LLRs to determine the range $[-s_i,s_i]$ that contains the LLRs with high probability. Subsequently, if $s_i>\ell$, then these LLRs are scaled by $\frac{\ell}{s_i}$ (no scaling otherwise) so that the scaled LLRs fall into $[-\ell,\ell]$. This adaptive procedure improves the performance of ExNE, especially in the high SNR range. We also note that if one performs clipping, i.e., truncate the decoder LLRs to the desired range of $[-\ell,\ell]$, then the performance is inferior to that of the adaptive scaling. One

explanation is that scaling keeps the relative confidence level of the bits intact, while the clipping operation does not.



(a) Training Data La (b)ExNE output LLR (c) Decoder LLR

Fig. 5: LLR histograms for (a) $\mathbf{L_a}$ generated in open-loop training. (b) ExNE output $\widetilde{\mathbf{L_E}}$ (see step 3 of Fig. 3) (c) decoder LLR $\mathbf{L_a}$. Data from (b) and (c) are collected from a close-loop simulation for channel $\mathbf{h_1}$ using ExNE and a convolutional code.

Fig. 5 (b) and (c) show the histograms of ExNE's output LLR and the decoder's output LLR after the 1st, 2nd, and 5th iteration of joint equalization and decoding, respectively. We see that after the first iteration, the histograms both center at around 0, meaning that neither the ExNE nor the decoder can distinguish the $+1$ and -1 bits reliably. After the 2nd iteration, ExNE's histogram becomes less peaky around 0 and starts to show initial separation of the two Gaussian peaks, corresponding to bit values of $+1$ and -1, respectively. The decoder's histogram shows a better separation of the peaks. By the end of the 5th iteration, the separation of two peaks is clear at the ExNE, while the decoder can separate them almost perfectly. This confirms ExNE can utilize the *a priori* LLR from the decoder efficiently to generate extrinsic LLRs. In this example, decoder LLRs in (c) are scaled by $\frac{\ell}{s_2} = \frac{14}{20}$, $\frac{\ell}{s_5} = \frac{14}{30}$ to match the range of training LLR shown in the top figure of (a) for $[-\ell, \ell]$ with $\ell = 14$.

IV. Numerical Results

We consider 4 linear ISI channels with impulse responses

$$\begin{aligned}
\mathbf{h_1} &= [0.160, 0.227, 0.460, 0.688, 0.460, 0.227, 0.160];\\
\mathbf{h_2} &= [0.227, 0.460, 0.688, 0.460, 0.227];\\
\mathbf{h_3} &= [0.16, 0.545, -0.672, 0.256, 0.095, -0.389];\\
\mathbf{h_4} &= [0.3482, 0.8704, 0.3482].
\end{aligned} \tag{5}$$

Their frequency responses are shown in Fig. 1. Note that ISI is high for $\mathbf{h_1}$ and $\mathbf{h_2}$, moderate for $\mathbf{h_3}$, and mild for $\mathbf{h_4}$. We also consider 2 nonlinear channels of the form (1), given by

$$\mathbf{h} = \mathbf{h}_1, g = g_1(u) = u + 0.2u^2 - 0.1u^3, \tag{6}$$

$$\mathbf{h} = \mathbf{h_4}, g = g_2(u) = u + 0.2u^2 - 0.1u^3 + 0.5\cos(\pi u). \tag{7}$$

Note that [2] considers a nonlinear channel $(\mathbf{h}_3, g_1)$. Here, we consider a more challenging combination $(\mathbf{h}_1, g_1)$ in (6). The nonlinear channel $(\mathbf{h}_4, g_2)$ in (7) is from [3], [5], [7].

For each channel defined above, for a given signal-to-noise power ratio (SNR), we perform the three-step training procedure described in III-A and III-B to train the ExNE, using the DCCB-SC network. The ExNE is implemented under a PyTorch framework. Detailed parameters for network training is shown in Table 1.

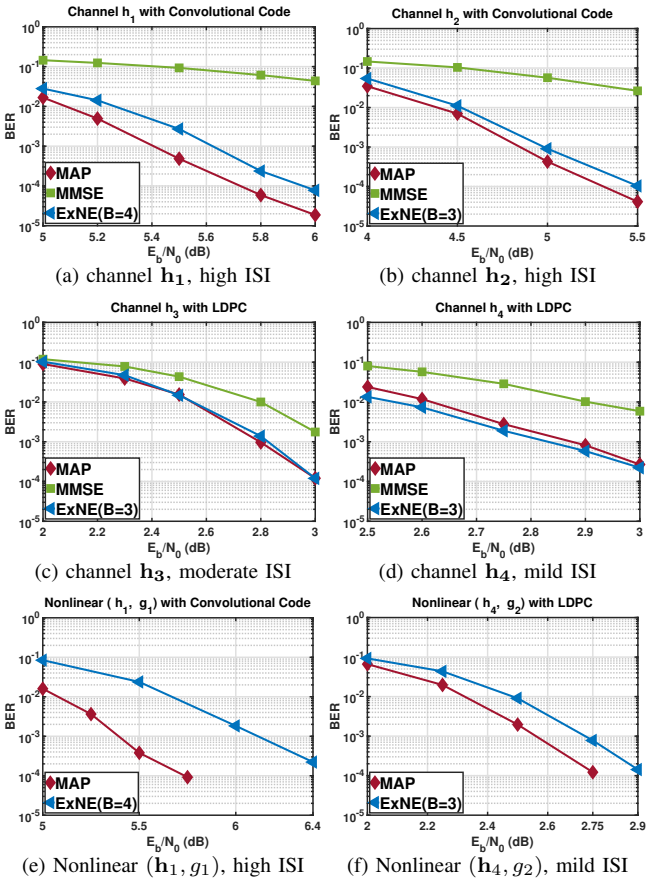
TABLE I: Hyper-parameters used in network training

# of training samples per SNR $= 10^4$	kernel size $= 5$
length of training sample vector $m = 286$	mini-batch size $= 128$
initial learning rate $= 0.001$	epoch number $= 200$
# of layers $= 4B + 1$	Optimizer= Adam
# parameters in DCCB-SC: 3640 for $B = 4$, 2734 for $B = 3$	

In the simulations, we use either a convolutional code with generating polynomial $[1 + D^2, 1 + D + D^2]$ or a regular $(3, 6)$ low-density parity-check (LDPC) code constructed by the Progressive-Edge-Growth (PEG) algorithm. Both codes are rate $R = 1/2$ and have a length of 3072 bits. For $\mathbf{h_3}$ and $\mathbf{h_4}$ with moderate or mild ISI, we use the LDPC code because it outperforms the convolutional code. In contrast, for high ISI channels $\mathbf{h_1}$ and $\mathbf{h_2}$, we use convolutional code because it performs better than LDPC code. This is consistent with observations from [14] that the difference in code performance is likely attributed to the drastically different channel characteristics compared to the AWGN channel.

Simulation results for 4 linear channels and 2 nonlinear channels are shown in Fig. 6. For each channel, we compare the performance of the ExNE with the MAP equalizer and the turbo MMSE equalizer [9]. Out of these three equalizers, the ExNE does not require any prior knowledge of the channel model, while the MAP and the MMSE equalizers assume full knowledge of the channel. In all figures, we plot the bit-error-rate (BER) as a function of E_b/N_0 in dB, given by $\frac{E_b}{N_0}(\text{dB}) = 10\log_{10}\left(\|g(\mathbf{x} * \mathbf{h})(\mathbf{n})\|/R\|w_n\|\right)$, where $R = 1/2$ is the coding rate. The results shown in Fig. 6 are after 6 iterations of joint equalization and decoding. Main observations from Fig. 6 are summarized as follows:

- The ExNE achieves excellent performance over all 4 linear channels. At BER $= 10^{-4}$, the gap between ExNE (unknown channel) and MAP (known channel) is only about 0.2 dB and 0.17 dB for high ISI channels $\mathbf{h_1}$ and $\mathbf{h_2}$, respectively. The ExNE performs virtually the same as MAP for channel $\mathbf{h_3}$ and even slightly better than MAP for channel $\mathbf{h_4}$ at lower E_b/N_0. It also outperforms the MMSE equalizer over all 4 channels and the performance gap is large for $\mathbf{h_1}$ and $\mathbf{h_2}$.
- For the highly challenging nonlinear channel $(\mathbf{h_1}, \mathbf{g_1})$, we note that the performance gap between ExNE and MAP increases to about 0.76 dB at BER $= 10^{-4}$. In comparison, the gap is only 0.15 dB for the less challenging combination of $(\mathbf{h_4}, \mathbf{g_2})$. The turbo MMSE equalizer is not applicable to nonlinear channels.
- The proposed DCCB-SC network works well with the depth of $B = 3$ or 4 for all channels considered. The number of parameters remains moderate 3640 ($B = 4$)



(a) channel $\mathbf{h_1}$, high ISI (b) channel $\mathbf{h_2}$, high ISI

(c) channel $\mathbf{h_3}$, moderate ISI (d) channel $\mathbf{h_4}$, mild ISI

(e) Nonlinear $(\mathbf{h}_1, g_1)$, high ISI (f) Nonlinear $(\mathbf{h}_4, g_2)$, mild ISI

Fig. 6: BER comparisons over linear and nonlinear channels.

for most challenging channels $\mathbf{h_1}$ and $(\mathbf{h}_1, g_1)$. For other channels, only 2734 parameters are needed ($B = 3$).

- The proposed ExNE works effectively with both convolutional codes and LDPC codes. This demonstrates the effectiveness of the proposed open-loop training that is independent of the channel coding constraint.
- Fig. 6 (d) shows that the MAP equalizer performs slightly worse than the ExNE after 6 iterations. A closer examination (results not shown here) reveals that MAP is better than ExNE after the 1st iteration, but its performance becomes inferior to ExNE in later iterations. This may be attributed to (2) which assumes accurate and independent prior $\{P(x_j), j = 1, \cdots, N\}$. This assumption holds during the 1st iteration with uniform priors, but may become less accurate in later iterations.

For the above channels and code choices, we have not found an APP-based NN equalizer whose performance is comparable to the ExNE in the low operating E_b/N_0 range shown in Fig. 6. Note that the E_b/N_0 range in Fig. 6 (c), (d), (f) is noticeably lower than those of [2]–[5] for channels $\mathbf{h_3}$, $\mathbf{h_4}$, and $(\mathbf{h_4}, \mathbf{g_2})$. Besides differences in the system setup and choice of channel code, we believe that proper use of extrinsic information in the ExNE is a key factor that allows us to achieve the low E_b/N_0 performance. Thus, in Fig. 6, we limit our comparisons to only the MAP and the MMSE equalizer.

V. Conclusion

In this work, we conducted a new study of the NN-based equalizer specifically for turbo equalization. We developed a novel ExNE equalizer that performs closely to the MAP equalizer over a variety of linear and non-linear ISI channels without knowledge of the channel model. The proposed ExNE features an open-loop, three-step training that is independent of the choice of channel code. This offers great flexibility for the application of ExNE in turbo equalization. We proposed a deep network, DCCB-SC, which enables ExNE to achieve a near optimal performance with only a moderate number of network parameters. We studied challenging ISI channels that were not considered previously for NN-based equalizers. The effectiveness of ExNE over these high ISI channels makes it a promising candidate for turbo equalization. Future work includes extensions of the proposed ExNE to higher-order modulations and to ISI channels with longer memory.

References

[1] S. Cammerer, F. A. Aoudia, S. Dörner, M. Stark, J. Hoydis, and S. Ten Brink, "Trainable communication systems: Concepts and prototype," *IEEE Transactions on Communications*, vol. 68, no. 9, pp. 5489–5503, 2020.

[2] A. Caciularu and D. Burshtein, "Unsupervised linear and nonlinear channel equalization and decoding using variational autoencoders," *IEEE Transactions on Cognitive Communications and Networking*, 2020.

[3] W. Xu, Z. Zhong, Y. Be'ery, X. You, and C. Zhang, "Joint neural network equalizer and decoder," in *2018 15th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, 2018, pp. 1–5.

[4] Y. Hu, L. Zhao, and Y. Hu, "Joint channel equalization and decoding with one recurrent neural network," in *2019 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2019, pp. 1–4.

[5] H. Ye and G. Y. Li, "Initial results on deep learning for joint channel equalization and decoding," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2017, pp. 1–5.

[6] N. Farsad and A. Goldsmith, "Sliding bidirectional recurrent neural networks for sequence detection in communication systems," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2331–2335.

[7] C. Teng, H. Ou, and A. A. Wu, "Neural network-based equalizer by utilizing coding gain in advance," in *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2019, pp. 1–5.

[8] B. Liu, S. Li, Y. Xie, and J. Yuan, "Deep learning assisted sum-product detection algorithm for faster-than-nyquist signaling," in *2019 IEEE Information Theory Workshop (ITW)*. IEEE, 2019, pp. 1–5.

[9] M. Tuchler, R. Koetter, and A. C. Singer, "Turbo equalization: principles and new results," *IEEE transactions on communications*, vol. 50, no. 5, pp. 754–767, 2002.

[10] T. Koike-Akino, Y. Wang, D. S. Millar, K. Kojima, and K. Parsons, "Neural turbo equalization: Deep learning for fiber-optic nonlinearity compensation," *Journal of Lightwave Technology*, vol. 38, no. 11, pp. 3059–3066, 2020.

[11] R.-H. Peng, R.-R. Chen, and B. Farhang-Boroujeny, "Markov chain monte carlo detectors for channels with intersymbol interference," *IEEE transactions on signal processing*, vol. 58, no. 4, pp. 2206–2217, 2009.

[12] S. Ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE transactions on communications*, vol. 52, no. 4, pp. 670–678, 2004.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[14] M. M. Mashauri, "Spatially coupled codes in turbo equalization," 2019.