The RNN BCJR Detector in Time-Varying Channels

Martina Magnaldi and Guido Montorsi
Dep. of Elec. and Telecomm. (DET)
Politecnico di Torino
Torino, Italy
martina.magnaldi@polito.it, guido.montorsi@polito.it

***Abstract*—A Recurrent Neural Network (RNN) detector is obtained by interfacing the RNN BCJR, introduced by the authors, with the channel output through a convolutional linear layer that mimics the presence of a shortening filter. The RNN detector can be trained to implement a channel and modulation-agnostic detector, including the functions of channel shortening, Maximum Likelihood (ML) sequence detection, and symbol or bit Log-Likelihood (LL) computation for the following soft input channel decoder. The RNN detector has a processing complexity that matches that of the corresponding classical receiver. In this paper we explore the effectiveness of its employment in static and time-selective scenarios.**

***Index Terms*—Additive SISO, BCJR algorithm, model-based neural networks, Maximum Likelihood equalization, channel shortening**

I. Introduction

Over the last few years, there has been an increasing research interest in using Deep Learning (DL) across multiple layers of the Open Systems Interconnection (OSI) stack in communication systems. As a reference, [1] includes a survey of the initial deployment of DL on the physical layer signal processing of communication systems. In this specific context, among all the kinds of Neural Networks (NNs), Recurrent Neural Networks (RNNs) have captured significant attention. They are particularly well-suited for sequential inputs, such as data transmissions, by retaining hidden states with past information [2].

First DL applications to physical layer signal processing aimed to substitute traditional Digital Signal Processing (DSP) blocks with off-the-shelf RNNs, originally designed for other tasks. Several studies found that properly trained RNNs could often match or exceed the performance of classical receivers.

It is common knowledge by now that NNs offer flexibility by being data-driven and channel model agnostic, and adaptability across various channel conditions, unlike traditional receivers that rely on specific models and unrealistic assumptions. Indeed, conventional detection algorithms like Viterbi [3] and BCJR (Bahl-Cocke-Jelinek-Raviv) [4] struggle without precise channel knowledge and accurate Channel State Information (CSI), which NNs do not require.

Anyway, these initial attempts of DL applications have raised two main problems.Firstly, the proposed NNs have much higher DSP complexity than classical models, raising concerns about their practicality. Secondly, the resulting network lacks interpretability. To overcome these issues, recent research has introduced "model-based" neural networks for digital receivers, replicating traditional DSP blocks while retaining flexibility. These networks are trainable from data and adaptable to various channel models, with processing complexity similar to classical receivers. We can mention [5], [6], and [7], [8], [9], which integrated NNs for CSI estimation into the Viterbi, BCJR, interference cancellation [10] algorithm and Kalman filter estimator. In [11] is proposed a set of data augmentation techniques tailored to digital communication data, while [12] presents a combination of meta-learning with a model-based inductive bias. An excellent survey of model-based DL approaches is presented in [13], while [14] discusses the key challenges these emerging networks encounter.

Within the same area of study, in a previous conference paper [15], we proposed using the RNN BCJR, a new trainable version of the *additive* BCJR [16], whose structure reflects an underlying trellis diagram. Unlike other proposals like [5] or [6] and [8], the presented RNN BCJR is fully trainable from output LLs, exploiting the smoothness and differentiability of the max* operator, the core operator in the additive BCJR's forward and backward recursions. The RNN BCJR consists of a linear layer to form the edge metrics from the state and input metrics, followed by a SOFTMAX/max* layer to marginalize the edge metrics back to the state and output spaces. It is defined by four trainable matrices describing the additive BCJR equations.

In this work, we focus on the extension of what has been presented in [15]. We propose a complete and trainable RNN detector which includes the RNN BCJR as the core block. This detector is based on the channel shortening approach [17], [18], using a convolutional linear layer to mimic the presence of the shortening filter before the RNN BCJR. This layer, in cascade to the channel, approximates the overall channel impulse response into a truncated one of lenght compatibale with the following BCJR trellis processing. The RNN detector can be trained to implement a channel and modulation agnostic detector, able to execute all the digital baseband processing from analog-to-digital (A/D) converter to the delivery of LL for the following soft-input decoder. The DSP complexity of the RNN detector is equal to that of the corresponding classical receiver, with the additional property of being trainable from the cost function defined on the delivered LLs.

In the result section, we consider a classical scenario with single carrier system. We focus on the application of our RNN detector on a channel characterized by a strong but short

inter-symbol interference (ISI). Initially, we consider the static mode with fixed channel impulse response, demonstrating that when the memory of the RNN BCJR is larger than one, the RNN detector achieves substantial improvement over Minimal Mean Square Error adaptive equalizer based on the stochastic gradient (ASG-MMSE), performing like a Maximum Likelihood Sequence Estimation (MLSE) receiver. In contrast, with a memory of one, indicating no trellis processing, it behaves as MMSE but with the crucial difference that it embeds the LLs computation block.

Next, we proceed to test the detector on the same ISI channel but in a time-varying condition by implementing a training phase with interspersed pilots. We explore the performance across different Doppler frequencies and pilot densities. These experiments highlight the adaptability of RNN detector in slow time-varying channels where minimal pilot density is required and confirm the same behavior observed in the static mode.

The remainder of the paper is organized as follows. In Section II, we introduce the RNN BCJR, whose structure is inherited from the additive version of the BCJR. In section II-A, we derive the fundamental recursions for its training through δ back-propagation. In section III, we present the RNN detector, which uses the RNN BCJR as the core block. In section III-A, we show how to extend the training from the RNN BCJR to the full detector. In Section IV, we provide simulation results showing the effectiveness of the RNN detector on a channel affected by ISI both in static and time-varying modes. Finally, in Section V, we provide conclusions and an outlook on future work.

II. The RNN BCJR

The RNN BCJR has been exhaustively presented in [15]. Here we recall some important aspects that are strictly necessary to understand our work. The BCJR algorithm is generally described in [16], [19]. The additive SISO accepts as input the LLs of the input and output symbols of a Finite State Machine (FSM) and provides as output an updated version of the two. This FSM is defined by a single trellis section. We define a set of states $\mathcal{S}$ of cardinality $|\mathcal{S}|$; a set of input symbols $\mathcal{U}$ of cardinality $|\mathcal{U}|$; a set of output symbols $\mathcal{C}$ of cardinality $|\mathcal{C}|$ and the set of edges $\mathcal{E}$, of cardinality $|\mathcal{E}| = |\mathcal{U}| \times |\mathcal{S}|$. The trellis section is described by four functions $s^S(e)$, $u(e)$, $c(e)$, $s^E(e)$, that associate to each edge e the corresponding starting state, input symbol, output symbol, and ending state.

The *additive* version of the BCJR algorithm is described by the forward and backward recursions and the output computation block. The simpler form that accepts LL λ on output symbols and provides LL of input symbols π reads:

$$\alpha^+(s) = \underset{e:\ s^E(e)=s}{\overset{*}{\max}} \{\alpha(s^S(e)) + \lambda(c(e))\}\,, \tag{1}$$

$$\beta^-(s) = \underset{e:\ s^S(e)=s}{\overset{*}{\max}} \{\beta(s^E(e)) + \lambda(c(e))\}\,, \tag{2}$$

$$\pi(u) = \underset{e:\ u(e)=u}{\overset{*}{\max}} \{\alpha(s^S(e)) + \lambda(c(e)) + \beta(s^E(e))\} \tag{3}$$

where $\alpha(s)$ is the LL of the state s given the past observations, and $\beta(s)$ is the LL of the state s given the future observations. In (1)-(2), to simplify notations, we dropped the time index and denoted $\alpha^+(s)$, $\beta^-(s)$ the state metrics at the next and previous trellis step, respectively.

In RNN literature, given a vector x, the term "SOFTMAX" often refers to the operator $\left(\frac{e^{x(i)}}{\sum_j e^{x(j)}}\right)_{i=1}^{N}$ that returns a vector. However, it has been recognized that this operator should be more properly referred to as the "SOFTARGMAX" operator. In this paper, since the SOFTMAX coincides with the $\max^*$ operator introduced in [16], we will use this term to refer to

$$\text{SOFTMAX}(x) = \log \sum_i \exp(x(i)) = \overset{*}{\max_i} x(i)\,, \tag{4}$$

while the SOFTARGMAX coincides with the gradient of the SOFTMAX : $\nabla\text{SOFTMAX}(x) = \text{SOFTARGMAX}(x)$.

The additive version of the SISO in (1)-(3), interpreted as an RNN, is the concatenation of one linear layer followed by a suitable SOFTMAX layer as represented in fig. 1.

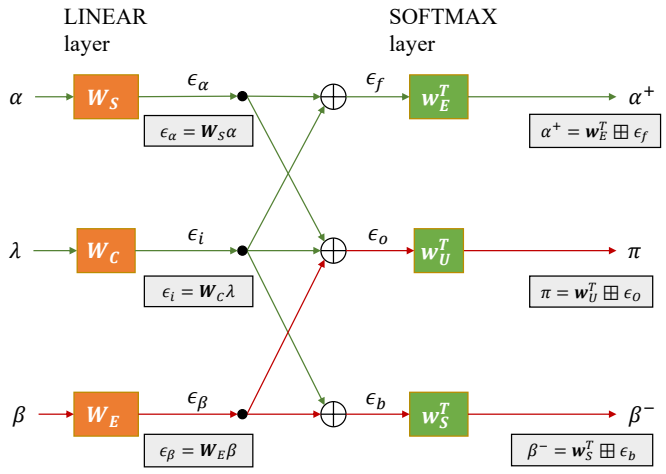

Fig. 1. The RNN-BCJR (predictive step).

Three *linear* layers are used to form the $|\mathcal{E}|$ edge metric components from the state LLs and the input metric λ. $\mathbf{W}_S$, $\mathbf{W}_E$ are $|\mathcal{E}| \times |\mathcal{S}|$ matrices that can embed the structure of "label-free" trellis associated to the functions $s^S(\cdot)$ and $s^E(\cdot)$, respectively. The $|\mathcal{E}| \times |\mathcal{C}|$ matrix $\mathbf{W}_C$, instead can embed the structure of the $c(\cdot)$ function.

Then the three max*/SOFTMAX layers are used to marginalize the edge LLs back to the state or desired output space. We adopt the *matrix* version of the $\max^*$ operator, defined as:

$$\mathbf{C} = \mathbf{A} \boxplus \mathbf{B} : \mathbf{C}(i,j) = \overset{*}{\max_k} \{\mathbf{A}(i,k) + \mathbf{B}(k,j)\}. \tag{5}$$

With the notation introduced in eq. (5), we can write the RNN BCJR recursions and output computation in a compact vector form as follows:

$$\alpha^+ = \mathbf{w}_E^T \boxplus \epsilon_f, \quad \beta^- = \mathbf{w}_S^T \boxplus \epsilon_b, \quad \pi = \mathbf{w}_U^T \boxplus \epsilon_o, \tag{6}$$

where $\mathbf{w}_U$ is a $|\mathcal{E}| \times |\mathcal{U}|$ matrix that can embed the structure of the $u(\cdot)$ function.

In our proposed RNN BCJR, the rows of $\mathbf{W}_S$ and $\mathbf{W}_E$ are related to the rows of $\mathbf{w}_S$ and $\mathbf{w}_E$ with a SOFTARGMAX operation:

$$\mathbf{W}_S(e,s) = \frac{e^{\mathbf{w}_S(e,s)}}{\sum_{s'} e^{\mathbf{w}_S(e,s')}}, \ \forall s,$$
$$\mathbf{W}_S(e) = \mathrm{SOFTARGMAX}(\mathbf{w}_S(e)). \quad (7)$$

With these constraints, the RNN BCJR structure is defined by four trainable weight matrices, namely $\mathbf{w}_S, \mathbf{w}_E$ defining the label-free trellis, $\mathbf{W}_C$ describing the linear layer to compute the input metric and $\mathbf{w}_U$ for the SOFTMAX layer to compute the output LLs.

A. Training the RNN BCJR: the δ back-propagation

Here we provide the recursions for the δ back-propagation used to train the four matrices defining the structure of the RNN BCJR from the gradient of the cost function C of the output LL π:

$$\delta\pi = \nabla C(\pi). \quad (8)$$

We considered as cost function C the conditional entropy with respect to the desired sequence of discrete input symbols:

$$C(\pi) = \log\left(\frac{\sum_k P(Y|k)}{P(Y|U)}\right) = \max_k{}^{*} \pi(k) - \pi(U), \quad (9)$$

where U is the transmitted input symbol. After some manipulations, one can obtain:

$$\delta\pi = \mathrm{SOFTARGMAX}(\pi) - 1_U, \quad (10)$$

where 1_U denotes the one-hot column vector with a one in position U.

The δ back-propagation through the SOFTMAX (max*) operator can be obtained by simple application of the chain rule. After some algebra, one can obtain:

$$\delta\epsilon_f = e^{\mathbf{z}_E}\delta\alpha^+, \qquad \delta\mathbf{w}_E(e,s) = e^{\mathbf{z}_E(e,s)}\delta\alpha^+(s) \quad (11)$$

where the exponents $\mathbf{z}_{(\cdot)}$ of the matrices are obtained as

$$\mathbf{z}_E(e,s) = \mathbf{w}_E(e,s) + \epsilon_f(e) - \alpha^+(s). \quad (12)$$

In back-propagation, sum nodes in the predictive step become repetition nodes and vice-versa, so that $\delta\epsilon_\alpha = \delta\epsilon_f + \delta\epsilon_o$.

Finally, back-propagation through the linear layer follows the standard rule of using the transpose matrix:

$$\delta\alpha = \mathbf{W}_S^T\delta\epsilon_\alpha, \qquad \delta\mathbf{W}_S(e,s) = \delta\epsilon_\alpha(e)\cdot\alpha(s). \quad (13)$$

Moreover, since rows of $\mathbf{W}_S, \mathbf{W}_E$ are related to those of $\mathbf{w}_S, \mathbf{w}_E$ with the SOFTARGMAX operator of eq. (7), we further merge the $\delta\mathbf{W}_S$ ($\delta\mathbf{W}_E$) contributions into $\delta\mathbf{w}_S$ ($\delta\mathbf{w}_E$). After some manipulations, one can find:

$$\delta\mathbf{w}_S(e,s) = \mathbf{W}_S(e,s)\cdot \left(\delta\mathbf{W}_S(e,s) - \sum_{s'}\mathbf{W}_S(e,s')\delta\mathbf{W}_S(e,s')\right). \quad (14)$$

Notice that, during training, back-propagation originates recursions in the opposite directions with respect to the predictive step, so the propagation of $\delta\alpha$ is now anticausal, whereas the propagation of $\delta\beta$ is causal.

To solve the anticausality problem efficiently, the detector's predictive and training phases can be performed with the sliding window with a grouped decision approach. Input streams are processed periodically in blocks of N trellis steps, with a latency of $N+D$. The anticausal recursions, β in the predictive phase, and $\delta\alpha$ in the training phase, are initialized with uniform metrics, and D initial backward steps are performed only for a reliable initialization of the following N backward steps for a total of $N+D$ backward steps. As training requires the cost function to be computed on the detector output, the training phase is performed with an additional latency of $N+D$ steps. In [15] we detailed metric normalization to prevent numerical issues in recursions.

The RNN BCJR is fully trainable from its output $\delta\pi$ obtained from the conditional entropy of output LLs. The delta propagation also delivers the output $\delta\lambda$, which can be propagated back to external blocks. This is in contrast with other similar proposals like [5], [6], [8], where NN is confined to some specific processing units. This allows embedding it in larger RNNs, as shown in the following section.

III. The RNN detector

In this section we present the focus of our work: the implementation of a trainable RNN detector with the RNN BCJR as its core block. The detector structure is based on the channel shortening approach ([17], [18]), where a trellis ML detector (BCJR) is preceded by a filter optimized for the trellis processor memory. The considered receiver scheme is represented in fig. 2.

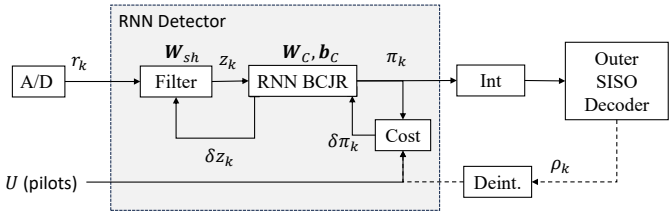

Fig. 2. The RNN Detector.

The discrete sequence of IQ samples r_k provided by the A/D converter feeds the shortening filter, which computes the sequence of observations z_k according to:

$$z_k = \mathbf{W}_{\mathrm{sh}}\mathbf{r}_k, \quad (15)$$

where $\mathbf{r}_k$ is the vector collecting the last L samples of the input sequence $\mathbf{r}_k \triangleq (r_k, r_{k-1}, \ldots, r_{k-L+1})$. $\mathbf{W}_{\mathrm{sh}}$ is a trainable weight matrix defining an additional convolutional linear layer in the RNN detector. The z_k sequence is the input of the RNN BCJR, which provides at its output the LL π_k to the following soft input decoder, e.g., a turbo decoder or an LDPC decoder. Between the detector and the decoder are considered the interleaver (*Int*) and deinterleaver (*Deint*).

The RNN BCJR has a memory of $L_T \geq 1$ symbols and a number of transitions from each state that depends on the modulation cardinality $M = 2^m$ (m is the modulation efficiency). The trellis complexity is then $2^{m \cdot L_T}$.

In section II, we have derived the RNN BCJR for symbol LL input and output. In [15], we showed that the same structure can handle cases with varying input and/or output forms.

In our case, the RNN BCJR performs ML sequence detection on channels affected by ISI. The metrics $\epsilon_i(e)$ are computed from the output of the filter z_k as:

$$\begin{aligned}\epsilon_i(e) =& \frac{1}{2\sigma^2}\left[2\Re(z_k a_k^*) - \gamma_0 \|a_k\|^2 - 2\Re\left(a_k^* \sum_{i=1}^{L_T-1} a_{k-i}\gamma_i\right)\right] \\ &= \frac{1}{\sigma^2}\Re(z_k a_k^*) - \frac{1}{\sigma^2} f(e) = \mathbf{W}_C z_k + \mathbf{b}_C \,. \end{aligned} \tag{16}$$

This last equation shows, as observed also in other papers [5], [8], that input edge metrics for the RNN BCJR can be obtained linearly from the observation z_k if an additional bias $\mathbf{b}_C$ is considered in the input linear layer.

The RNN detector is configured by providing the length of the shortening filter L, the modulation cardinality $M = 2^m$, the memory of the trellis processor L_T, and the type of delivered LLs π_k, which can be either a single LL vector on the constellation set, or m LL of the bits labelling the set for BICM receivers. Including the shortening filter reduces the required trellis memory L_T, which is a significant advantage since higher trellis memory results in exponentially increasing complexity.

A. Training the RNN Detector

During the training phase (see the lower part of fig. 2), the known transmitted symbols or bits U, e.g. pilots, are used to compute the conditional entropy of output LL, and the input $\delta\pi_k$ for training, according to eqs. (9) and (10). The output δz_k computed by the RNN BCJR, as explained in section II-A, are further back propagated to train the coefficient of the additional convolutional linear layer ($\mathbf{W}_{\rm sh}$ in fig. 2). The RNN detector is consequently fully trained from its output cost function, which is the conditional entropy of output LLs.

In this context, the BCJR label-free trellis is known, as we assume for the channel the state evolution of an FIR filter, where the next state is obtained by shifting out from the register the oldest hypotheses and inserting the hypotheses on the current symbol a_k:

$$s_k = (a_{k-1}, \ldots, a_{k-L_T+1}) \rightarrow s_{k+1} = (a_k, \ldots, a_{k-L_T+2}).$$

Furthermore, the output symbol coincides with the least significant symbol of the ending state so that the rows of the output matrix $\mathbf{w}_U$ are also fixed.

In conclusion, the trainable weights of the RNN detector are those relative to the shortening filter $\mathbf{W}_{sh}$ ($4 \cdot L$ real parameters), and those for computing the input metrics $\mathbf{W}_C$ ($2^{mL_T} \cdot 2$ real parameters) with their biases $\mathbf{b}_C$ (2^{mL_T} real parameters).

Observe that the knowledge of the constellation set and binary labelling of the constellation points are not required as they are learnt in the training step, so the resulting detector is channel and modulation agnostic.

B. Interpretation

The proposed RNN-based detector can be seen as a generalization of the classical Minimal Mean Square Error adaptive equalizer based on the stochastic gradient (ASG-MMSE), with two main differences. The first one is the choice of the cost function during training. Instead of minimizing MSE on constellation symbols, the RNN detector minimizes the conditional entropy of the output LLs. This makes the RNN detector constellation and binary labelling agnostic and thus robust to non-linear impairments that may distort the constellation. Furthermore, SNR estimation required for the correct LL computation in ASG-MMSE is included in the training of the weights of $\mathbf{W}_C$ and their biases $\mathbf{b}_C$. The second difference is that the training encompasses both the linear filtering and the trellis detector based on BCJR. The NN's capacity to perform efficiently delta propagation across non-linearities makes it possible to apply the gradient algorithm technique to non-linear devices and other cost functions.

IV. Results

In this section, we provide simulation results that show how the RNN detector can learn to perform as an optimal shortening detector for channel affected by ISI, first in the static mode and then in the time-varying condition.

For testing the detector, we consider a channel characterized by a short but strong ISI. The RNN detector uses a sliding window with grouped decision scheduling. The initialization window is set to $D = 10$, with $N = 40$. The coefficient updating is then performed with periodicity 40. After some optimization, the updating step (learning rate) for backward propagation was set to 0.01. For all tests, the weights of the detector's trainable matrices ($\mathbf{W}_{\rm sh}$, $\mathbf{W}_C$, $\mathbf{b}_C$) are initialized with independent zero-mean Gaussian variables with standard deviation 0.2. The matrices $\mathbf{W}_S$, $\mathbf{W}_E$, $\mathbf{w}_U$ are set according to the label-free trellis structure. The static channel is characterised by a fixed discrete-time complex impulse response, which is:

$$h = (-0.69 + j0.15, -0.71 + j0.046). \tag{17}$$

Performance results are displayed in fig. 3. We use Mutual Information (MI) evaluated on the delivered LL as the performance metric, rather than the commonly used Bit Error Rate (BER). In the considered receiver of fig. 2, the detector generates messages for the outer soft-input and capacity-achieving decoder. While BER is relevant for assessing channel decoder performance, MI is a more suitable metric for the detector, as it is a proxy of the outer code rate required for reliable transmission.

The figure reports the MI of delivered LLs versus the channel SNR in the range [0, 20] dB, for BPSK, 4QAM,

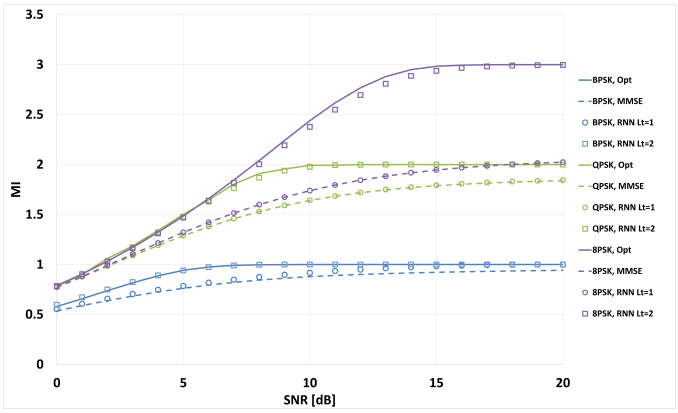

Fig. 3. MI vs SNR for the strong ISI channel. Comparison of ASG-MMSE, optimal ML detector, and RNN detector with L_T = 1 and L_T = 2.

and 8PSK constellations (blue, green and violet plots, respectively), and 4 detectors. Solid lines show the performance of the optimal ML detector based on BCJR with perfect CSI knowledge. The input filter is matched to the channel impulse response, and the memory of trellis processing coincides with the channel memory $L_C = L_T = 2$. The dashed lines report the performance of the ASG-MMSE-based detector. The output LLs are obtained by computing the distance of the MMSE filter output z_k from all constellation points $m(c)$, assumed to be known at the receiver. The distance is then normalized with the estimated residual MSE σ_z^2:

$$\pi_k(c) = -\frac{1}{2\sigma_z^2}|z_k - m(c)|^2, \quad c \in [1, M].$$

Circle and square (discrete) markers represent the performance of the RNN detector with $L = 4$, and L_T = 1 (circles) or L_T = 2 (squares). The RNN detector is unaware of the channel or constellation set but knows the constellation cardinality M.

The RNN detector demonstrates identical performance to the MMSE detector with L_T = 1 and matches the optimal detector with L_T = 2. A slight improvement is observed for the RNN detector compared to the MMSE due to the improved metric and residual variance estimates. Because interference is not Gaussian, there is a significant performance difference between the ideal detector and the MMSE-based detector, particularly for large constellation cardinality. Given that the channel remains constant over time, the RNN detector requires only a brief and limited period for initial training, eliminating the need for further re-training.

Now, we study how the RNN detector behaves in the presence of a slow time-varying channel. The two taps coefficients eq. (17) are substituted by two equal power Gaussian processess with Jake's spectrum [20] and a given Doppler frequency. Since the adaptivity of the RNN detector is based on training, we use interspersed pilots to feed the back-propagation (i.e. training is performed only when pilots are in use). They are required to periodically adapt the detector to the time-varying channel at the expense of the system's data rate.

We want to study the behavior of the RNN detector considering different pilot densities, pilot fields and Doppler frequencies. We define the pilot density (overhead) as the number of pilots sent with respect to the entire transmission that includes both pilots and data while the pilot field is the length of each pilot transmission. Given the initialization parameters, we choose the pilot field to be equal to 20 symbols. We saw from several preliminary experiments that the length of the pilot field has a negligible impact on the performance so we set it to a small value to keep the periodicity small. For each experiment we simulate time series of 10 millions samples. Performances for the channel with strong ISI are shown in fig. 4.

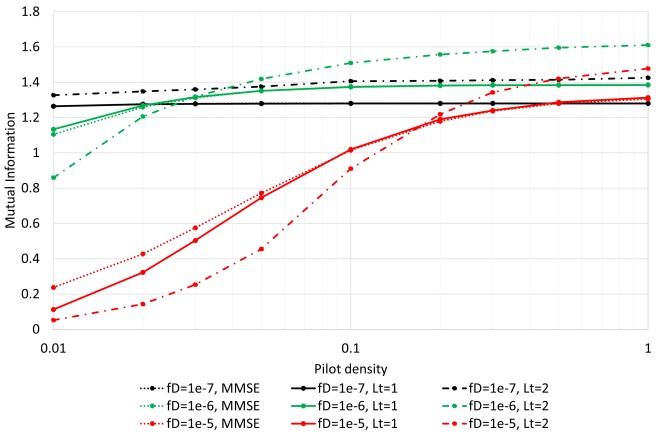

Fig. 4. Mutual Information vs pilot density. Strong ISI channel, SNR = 7 dB, f_D = 1e-7, 1e-6, 1e-5. Comparison of ASG-MMSE and RNN detector with L_T = 1,2.

The figure displays the MI of delivered LLs versus the pilot density, in logarithmic scale, for 4QAM constellation and SNR = 7dB. This specific SNR value is chosen based on the results from the static channel (fig. 3), where the gap between the ASG-MMSE and the RNN detector with L_T = 2 is larger. Three different normalized Doppler frequencies f_D values (1e-7, 1e-6 and 1e-5) and 3 detectors are considered. The dotted lines show the performance of the ASG-MMSE-based detector, while the solid and the dash-dotted lines report the output of the RNN detector for L_T = 1 and L_T = 2, respectively. Considering the black curves, relative to a very slow time-varying condition f_D = 1e-7, the outcomes show that the RNN detector exactly replicates the performance of the MMSE detector when L_T = 1. Rather, with L_T = 2, the RNN detector performs better across all pilot densities. However, as the Doppler frequency increases to e.g. f_D = 1e-5 (red curves), denoting a faster time-varying channel, the findings indicate a different situation. When pilot density is low, the ASG-MMSE outperforms the RNN detector. The RNN detector needs more time to adapt its weights to the time-varying channel, especially when $L_T > 1$ and the complexity of the trellis increases, so a low pilot density results insufficient to guarantee the same performance of the MMSE. Instead, as the overhead increases, the RNN detector with L_T = 1 gets comparable performance to the ASG-MMSE, while the

one with $L_T = 2$ achieves even higher values, as expected from the results obtained in the static condition. To note, asymptotic MI values differ across simulations due to the use of time series. The key takeaway is the trend: as the normalized Doppler frequency increases, more pilots are required for the RNN detector with $L_T = 2$ to surpass the MMSE. To have a more complete analysis, we test our detector also in the TDL-A channel specified in the 5GNR standard. This channel is characterized by many interferers spread in time, in contrast to the one presented before. Delays and power levels are defined according to [21]. The analysis reveals that increasing the trellis processing to $L_T = 2$ adds complexity but provides no significant performance advantage. This is consistent with the fact that in this scenario, where there are many weak interferers, MLSE does not provide significant gain with respect to MMSE.

V. Conclusions

Our main focus was on implementing a trainable RNN detector based on channel shortening approach. The RNN BCJR serves as the core block. Even though the RNN BCJR was fully detailed in [15], the most significant aspects have been reported here to introduce our work. By adding an input convolutional linear layer before the RNN BCJR, we have constructed a trainable RNN detector with a structure inherited from the channel-shortening approach. The RNN detector results to be fully trainable from its output LLs. This fundamental feature is inherited from the RNN BCJR, whose delta propagation output can be back-propagated to train the shortening filter coefficients. This characteristic provides the adaptability needed to deal with unknown channels or potential mismatches. The properly trained RNN detector results to be a channel and modulation agnostic detector able to execute the functions of channel shortening, ML sequence detection, and symbol or bit LL computation for the following soft input channel decoder. It can replicate the performance of a MMSE when the trellis memory L_T is equal to 1 and the performance of the MLSE $L_T > 1$.

Although we have presented the detector's performance in a classical scenario with single carrier systems affected by ISI, its employment can be easily adapted in all scenarios where channel shortening is effective. For example, systems with digital interference from multiple resources, or where the data symbol affects multiple observations. These scenarios include MIMO and multi-user detectors, as well as detectors for systems using different resource allocation strategies, like OFDM affected by ISI and ICI or OTFS systems.

Concerning the time adaptability, the approach used in this work is efficient when the channel's coherence time is much larger than the symbol interval. When this condition does not hold, alternative model-based RNN structures should be investigated, where channel estimation is jointly considered in the predictive step.

Acknowledgement

This work was supported by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3,CUP E13C22001870001,partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

References

[1] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[4] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[5] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3319–3331, 2020.

[6] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Data-Driven Factor Graphs for Deep Symbol Detection," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 2682–2687.

[7] N. Shlezinger, R. Fu, and Y. C. Eldar, "DeepSIC: Deep Soft Interference Cancellation for Multiuser MIMO Detection," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1349–1362, 2021.

[8] J. Yang, Q. Du, and Y. Jiang, "Neural network-assisted receiver design via learning trellis diagram online," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 8075–8085, 2022.

[9] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. van Sloun, and Y. C. Eldar, "KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.

[10] J. Andrews, "Interference cancellation for cellular systems: a contemporary overview," *IEEE Wireless Communications*, vol. 12, no. 2, pp. 19–29, 2005.

[11] T. Raviv and N. Shlezinger, "Data augmentation for deep receivers," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 8259–8274, 2023.

[12] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Online meta-learning for hybrid model-based deep receivers," *IEEE Transactions on Wireless Communications*, vol. 22, no. 10, pp. 6415–6431, 2023.

[13] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *Proceedings of the IEEE*, 2023.

[14] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Adaptive and flexible model-based ai for deep receivers in dynamic channels," *IEEE Wireless Communications*, 2024.

[15] G. Montorsi and B. Ripani, "RNN BCJR: a fully trainable version of the additive BCJR algorithm," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 3696–3701.

[16] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European transactions on telecommunications*, vol. 9, no. 2, pp. 155–172, 1998.

[17] F. Rusek and A. Prlja, "Optimal channel shortening for MIMO and ISI channels," *IEEE transactions on wireless communications*, vol. 11, no. 2, pp. 810–818, 2011.

[18] D. D. Falconer and F. Magee Jr, "Adaptive channel memory truncation for maximum likelihood sequence estimation," *Bell System Technical Journal*, vol. 52, no. 9, pp. 1541–1562, 1973.

[19] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A Soft-Input Soft-Output APP module for iterative decoding of concatenated codes," *IEEE Communications Letters*, vol. 1, no. 1, pp. 22–24, 1997.

[20] M. J. Gans, "A power-spectral theory of propagation in the mobile-radio environment," *IEEE Transactions on Vehicular Technology*, vol. 21, no. 1, pp. 27–38, 1972.

[21] 3GPP, "Study on channel model for frequencies from 0.5 to 100 ghz," 2018.